EVALITA
Evaluation of NLP and Speech Tools for Italian

Proceedings of the Final Workshop

12-13 December 2018, Naples

Editors:

Tommaso Caselli
Nicole Novielli
Viviana Patti
Paolo Rosso

aA accademia university press

aA

AILC

EVALITA Evaluation of NLP and Speech Tools for Italian

Proceedings of the Final Workshop 12-13 December 2018, Naples

Tommaso Caselli, Nicole Novielli, Viviana Patti and Paolo Rosso (dir.)

DOI: 10.4000/books.aaccademia.4421
Publisher: Accademia University Press
Place of publication: Torino
Year of publication: 2018
Published on OpenEdition Books: 5 June 2019
Serie: Collana dell'Associazione Italiana di Linguistica Computazionale
Electronic ISBN: 9788831978699

http://books.openedition.org

Printed version
Date of publication: 1 December 2018
ISBN: 9788831978422
Number of pages: 281

Electronic reference
CASELLI, Tommaso (ed.) ; et al. *EVALITA Evaluation of NLP and Speech Tools for Italian: Proceedings of the Final Workshop 12-13 December 2018, Naples.* New edition [online]. Torino: Accademia University Press, 2018 (generated 12 juin 2019). Available on the Internet: <http://books.openedition.org/aaccademia/4421>. ISBN: 9788831978699. DOI: 10.4000/books.aaccademia.4421.

Accademia University Press
via Carlo Alberto 55
I-10123 Torino
info@aAccademia.it

isbn 978-88-31978-42-2
www.aAccademia.it/EVALITA_2018

Accademia University Press è un marchio registrato di proprietà
di LEXIS Compagnia Editoriale in Torino srl

Preface to the Evalita 2018 Proceedings*

Welcome to EVALITA 2018! EVALITA is the evaluation campaign of Natural Language Processing and Speech Tools for the Italian language. This year we celebrate 10 years of EVALITA and shared tasks covering the analysis of both written and spoken language with the aim of enhancing the development and dissemination of resources and technologies for Italian. EVALITA is an initiative of the Italian Association for Computational Linguistics (AILC, http://www.ai-lc.it/). It is supported by the Italian Association for Artificial Intelligence (AI*IA, http://www.aixia.it/) and by the Italian Association of Speech Science (AISV, http://www.aisv.it/).

This volume collects the reports of the tasks organisers and of the participants to all of the EVALITA 2018 tasks. This year we helped to organize 10 tasks structured in four tracks: i.) *Affect, Creativity and Style*: Aspect-based Sentiment Analysis (ABSITA), Italian Emoji Prediction (ITAMoji), Irony Detection in Twitter (IronITA), Cross-Genre Gender Prediction (GxG); ii.) *Dialogue Systems*: itaLIan Speech acT labEliNg (iLISTEN), Italian DIALogue systems evaluation (IDIAL); iii.) *Hate Speech*: Automatic Misogyny Identification (AMI); Hate Speech Detection (HaSpeeDe); iv.) *Semantics4AI*: Solving language games (NLP4FUN), Spoken Utterances Guiding Chefs Assistant Robots (SUGAR).

The volume opens with an overview to the campaign, in which we describe the tasks in more detail, provide figures on the participants, and, especially, highlight the innovations introduced in this year's edition. The abstract of Saif M. Mohammad's keynote with title *"The Search for Emotions, Creativity, and Fairness in Language"* is also included in this introductory part of the volume.

The final workshop was held in Turin on the 12th and 13th of December 2018 as a co-located event of the Fifth Italian Conference on Computational Linguistics (CLiC-it 2018, http://clic2018.di.unito.it/en/home-2/). The workshop has been an occasion for organizers and participants, from both academic institutions and companies, to disseminate their work and results and to share ideas through oral and poster presentations. This year the program also includes a panel on *"The Future of Shared Tasks: data, evaluation, and technology transfer"* as a moment of reflection after 10 years of EVALITA, also in light of the new regulations on privacy (i.e., the EU General Data Protection Regulation (GDPR)). A major innovation of this year is the best system award across all tasks. This wants to be a further incentive to involve students (both, undergraduates and Ph.D. students) and to push the boundaries of the state of the art further by "daring to experiment" rather than just winning.

We thank all the people and institutions involved in the organisation of the tasks, and all the participants, who contributed to the success of the event. A special thank is due to AILC. Thanks are also due to AI*IA and AISV for endorsing EVALITA, to FBK and Manuela Speranza for making the web platform available once more for this edition (http://www.evalita.it), to our sponsors: CELI (https://www.celi.it/), Google Research, ELRA (http://elra.info/en/), and to Agenzia per l'Italia Digitale (AGID, https://www.agid.gov.it) for its endorsement. Last but not least, we heartily thank our invited speaker, Saif M. Mohammad from National Research Council Canada, for agreeing to share his expertise on key topics of EVALITA 2018 and to participate in our panel discussion, and all the other panelists: Franco Cutugno, (Università degli Studi di Napoli Federico II), Caterina Flick (Lawyer and coordinator of the Web & Media Committee of the *Fédération Internationale des Femmes des Carrières Juridiques*) and Malvina Nissim (Rijksuniversiteit Groningen).

November 2018

Tommaso Caselli
Nicole Novielli
Viviana Patti
Paolo Rosso

* Originally published online by CEUR Workshop Proceedings (CEUR-WS.org ISSN 1613-0073)

Chairs

Tommaso Caselli, Rijksuniveristeit Groningen, The Netherlands
Nicole Novielli, Università degli Studi di Bari “A. Moro”, Italy
Viviana Patti, Università degli Studi di Torino, Italy
Paolo Rosso, PRHLT Research Center, Universitat Politècnica de València, Spain

Steering Committee

Maria Anzovino, Università degli Studi di Milano-Bicocca, Italy
Francesco Barbieri, Universitat Pompeu Fabra, Spain
Pierpaolo Basile, Università degli Studi di Bari “A. Moro”, Italy
Valerio Basile, Università degli Studi di Torino, Italy
Cristina Bosco, Università degli Studi di Torino, Italy
Francesca Chiusaroli, Università degli Studi di Macerata, Italy
Alessandra Cignarella, Università degli Studi di Torino, Italy
Danilo Croce, Università degli Studi di Roma “Tor Vergata”, Italy
Francesco Cutugno, Università degli Studi di Napoli “Federico II”, Italy
Marco De Gemmis, Università degli Studi di Bari “A. Moro”, Italy
Felice Dell’Orletta, Istituto di Linguistica Computazionale “A. Zampolli”, CNR Pisa, Italy
Maria Di Maro, Università degli Studi di Napoli “Federico II”, Italy
Sara Falcone, Fondazione Bruno Kessler, Italy
Elisabetta Fersini, Università degli Studi di Milano-Bicocca, Italy
Simona Frenda, Università degli Studi di Torino, Italy
Marco Guerini, Fondazione Bruno Kessler, Italy
Bernardo Magnini, Fondazione Bruno Kessler, Italy
Malvina Nissim, Rijksuniveristeit Groningen, The Netherlands
Nicole Novielli, Università degli Studi di Bari “A. Moro”, Italy
Antonio Origlia, Università degli Studi di Napoli “Federico II”, Italy
Endang Wahyu Pamungkas, Università degli Studi di Torino, Italy
Viviana Patti, Università degli Studi di Torino, Italy
Fabio Poletto, Acmos, Torino, Italy
Marco Polignano, Università degli Studi di Bari “A. Moro”, Italy
Francesco Ronzano, Universitat Pompeu Fabra and Hospital del Mar Medical Research Center, Spain
Paolo Rosso, PRHLT Research Center, Universitat Politcnica de Valncia, Spain
Manuela Sanguinetti, Università degli Studi di Torino, Italy
Giovanni Semeraro, Università degli Studi di Bari “A. Moro”, Italy
Lucia Siciliani, Università degli Studi di Bari “A. Moro”, Italy
Maurizio Tesconi, Istituto di Informatica e Telematica, CNR Pisa, Italy
Claudia Tortora, Università degli Studi di Napoli “LOrientale”, Italy

Website

Manuela Speranza, Fondazione Bruno Kessler, Italy

Contents

PART I

INTRODUCTION TO EVALITA 2018 AND TASK OVERVIEWS

Evalita 2018: Overview on the 6th Evaluation Campaign of Natural Language Processing and Speech Tools for Italian

Tommaso Caselli
Rijksuniversiteit Groningen
Groningen, The Netherlands
t.caselli@gmail.com

Nicole Novielli
Dipartimento di Informatica
Universit degli Studi di Bari Aldo Moro, Italy
nicole.novielli@uniba.it

Viviana Patti
Dipartimento di Informatica
Universit degli Studi di Torino, Italy
patti@di.unito.it

Paolo Rosso
PRHLT Research Center
Universitat Politcnica de Valncia, Spain
prosso@dsic.upv.es

1 Introduction

EVALITA[1] is the evaluation campaign of Natural Language Processing and Speech Tools for Italian. Since 2007, the general objective of EVALITA is to promote the development and dissemination of language resources and technologies for Italian, providing a shared framework where different systems and approaches can be evaluated in a consistent manner.

EVALITA is an initiative of the Italian Association for Computational Linguistics (AILC)[2] and it is endorsed by the Italian Association for Artificial Intelligence (AI*IA)[3] and the Italian Association for Speech Sciences (AISV)[4].

2 Tasks and Challenge

For the 2018 edition, ten tasks are organized along the following tracks:

Affect, Creativity and Style

- *ABSITA - Aspect-based Sentiment Analysis*. The task is organized as a cascade of two subtasks consisting in automatically annotating sentences from hotel reviews with respect to the identified aspects (Aspect Category Detection (ACD) subtask) and the polarity associated to each one of them (Aspect Category Polarity (ACP) subtask) (Basile et al., 2018a);

- *ITAMoji - Italian Emoji Prediction*. The goal of this task is to develop a system for predicting the most likely emoji associated to a tweet. For simplicity purposes, tweets including only one emoji are considered (Ronzano et al., 2018);

- *IronITA - Irony Detection in Twitter*. The task aims at automatically identifying ironic tweets along two different subtasks. Specifically, the irony detection subtask (Task A) is a binary classification task where the systems are required to predict whether a tweet is ironic or not, while the second subtask (Task B) focuses on the identification of the different types of irony, with special attention to sarcasm recognition (Cignarella et al., 2018);

- *GxG - Cross-Genre Gender Prediction*. This task addresses the problem of gender prediction across different textual genres. Specifically, given a collection of texts from a specific genre, the gender of the author has to be predicted as either female or male. A dataset from different genres is distributed to the participants and gender prediction has to be done either (i) using a model which has been trained on the same genre, or (ii) using a model which has been trained on anything but that genre (Dell'Orletta and Nissim, 2018).

[1] http://www.evalita.it
[2] http://www.ai-lc.it
[3] http://www.aixia.it
[4] http://www.aisv.it

Dialogue Systems

- *iLISTEN - itaLIan Speech acT labEliNg*. This task consists in automatically annotating dialogue turns with speech act labels, i.e. with the communicative intention of the speaker, such as statement, request for information, agreement, opinion expression, general answer (Basile and Novielli, 2018).

- *IDIAL - Italian DIALogue systems evaluation*. The task develops and applies evaluation protocols for the quality assessment of dialogue systems for the Italian language. The target of the evaluation are existing task-oriented dialogue systems, both from industry and academia (Cutugno et al., 2018).

Hate Speech

- *AMI - Automatic Misogyny Identification*. This task focuses on the automatic identification of misogynous content both in English and in Italian languages in Twitter. More specifically, it is a two-fold task. It includes: (i) a Misogyny Identification subtask consisting in a binary classification of tweets as being either misogynous or not; (ii) a Misogynistic Behaviour and Target Classification subtask aimed at classifying tweets according to different finer-grained types of misogynistic behaviour detected, such as sexual harassment or discredit, and the target of the message (individuals or group of people). (Fersini et al., 2018a);

- *HaSpeeDe - Hate Speech Detection*. This task is organized into three sub-tasks, concerning: (i) the identification of hate speech on Facebook (HaSpeeDe-FB), (ii) the identification of hate speech on Twitter (HaSpeeDe-TW), and (iii) the cross-dataset setting concerning the assessment of the performance of the hate speech recognition system developed, i.e., when trained on Facebook data and evaluated on Twitter data, and vice versa (Bosco et al., 2018).

Semantics4AI

- *NLP4FUN - Solving language games*. This task consists in designing a solver for "The Guillotine" game, inspired by an Italian TV show. The game involves a single player, who is given a set of five words - the clues - each linked in some way to a specific word that represents the unique solution of the game. Words are unrelated to each other, but each of them has a hidden association with the solution. Once the clues are given, the player has to provide the unique word representing the solution. The participant systems are required to build an artificial player able to solve the game (Basile et al., 2018b).

- *SUGAR - Spoken Utterances Guiding Chef's Assistant Robots*. This task goal is to develop a voice-controlled robotic agent to act as a cooking assistant. To this aim, a train corpus of spoken commands is collected and annotated using a 3D virtual environment that simulates a real kitchen where users can interact with the robot. The task specifically focuses on a set of commands, whose semantics is defined according to the various possible combination of actions, items (i.e. ingredients), tools and different modifiers (Di Maro et al., 2018).

3 Fostering Reproducibility and Cross-community Engagement

Open access to resources and research artifacts, such as data, tools, and dictionaries, is deemed crucial for the advancement of the state of the art in scientific research. Accessibility of resources and experimental protocols enable both full and partial replication of studies in order to further validate their findings, towards building of new knowledge based on solid empirical evidence. To foster reproducibility and encourage follow-up studies leveraging the resources built within EVALITA 2018, we introduced two novelties this year. First of all, we intend to distribute all datasets used as benchmark for the tasks of this edition. To this aim, we have set up a repository on Github[5], in line with the good practices already applied by the organizers of the previous edition[6]. Also, the datasets for all the tasks will be

[5]The dataset of EVALITA 2018 made available by the task organizers can be found at: `https://github.com/evalita2018/data`

[6]The datasets of EVALITA 2016 can be found at: `https://github.com/evalita2016/data`

hosted and distributed by the European Language and Resources Association (ELRA). In addition, we decided to further encourage the sharing of resources by making availability of the systems an eligibility requirement for the best system award (see Section 4).

In the same spirit, we encouraged cross-community involvement in both task organization and participation. We welcomed the initiative of the organizers of AMI, the Automatic Misogyny Identification task (Fersini et al., 2018a), focusing on both English and Italian tweets. This task has been proposed first at IberEval 2018 for Spanish and English (Fersini et al., 2018b), and then re-proposed at Evalita for Italian, and again for English with a new dataset for training and testing. The ITAmoji shared task was also a re-proposal for the Italian language of the *Multilingual Emoji Prediction Task* at International Workshop on Semantic Evaluation (SemEVAL 2018) (Barbieri et al., 2018), which focused on English and Spanish. Here the re-proposal of the task at Evalita was driven by twofold aim to widen the setting for cross-language comparisons for emoji prediction in Twitter and to experiment with novel metrics to better assess the quality of the automatic predictions, also proposing a comparison with human performances on the same task.

In the 2016 edition task organisers were encouraged to collaborate on the creation of a shared test set across tasks (Basile et al., 2017). We were happy to observe that also this year this practice was maintained. In particular, a portion of the dataset of IronITA (Cignarella et al., 2018), the task on irony detection in Twitter, partially overlaps with the dataset of the hate speech detection task (HaSpeeDe) (Bosco et al., 2018). The intersection includes tweets related to three social groups deemed as potential target for hate speech online: immigrants, Muslims and Roma. Also, the sentiment corpora with multi-layer annotations developed in last years by the EVALITA community, which included also morpho-syntactic and entity linking annotations, were exploited by some ABSITA (Basile et al., 2018a) and IronITA (Cignarella et al., 2018) participants to address the finer-grained sentiment related tasks proposed this year under the *Affect, Creativity and Style track*.

4 Award: Best System Across-tasks

For the first time, this year we decided to award the best system across-task, especially that of young researchers. The award was introduced with the aim of fostering student participation to the evaluation campaign and to the workshop, and received a funding from *Google Research*, *CELI*[7], and from the *European Language and Resources Association* (ELRA)[8].

Criteria for eligibility, are (i) the availability of the system as open source software by the end of the evaluation period, when the results are due to the task organizers, and (ii) the presence of at least one PhD candidate, a master or a bachelor student among the authors of the final report describing the system. The systems will be evaluated based on:

- *novelty*, to be declined as novelty of the approach with respect to the state of the art (e.g. a new model or algorithm), or novelty of features (for discrete classifiers);
- *originality*, to be declined as identification of new linguistic resources employed to solve the task (for instance, using WordNet should not be considered as a new resource), or identification of linguistically motivated features; or implementation of theoretical framework grounded in linguistics;
- *critical insight*, to be declined as a deep error analysis that highlights limits of the current system and pave direction to future challenges; technical soundness and methodological rigor.

We collected 7 system nominations from the organizers of 5 tasks belonging to the *Affect, Creativity and Style track* and to the *Hate Speech track*. 14 students were involved in the development of the systems which received a mentions: 7 PhD students and and 7 master students. Most students are enrolled in Italian universities, but 5 of them. The award recipient(s) will be announced during the final EVALITA workshop, co-located with CliC-it 2018, the Fifth Italian Conference on Computational Linguistics[9].

[7] https://www.celi.it/
[8] http://elra.info/en/
[9] http://clic2018.di.unito.it/it/home/

5 Participation

The tasks and the challenge of EVALITA 2018 attracted the interest of a large number of researchers from academia and industry, for a total of 237 single preliminary registrations. Overall, 50 teams composed of 115 individuals from 13 different countries participated to one or more tasks, submitting a total of 34 system descriptions.

Table 1: Registered and actual participants, with overall number of teams and submitted runs.

Track	**Task**	**Participants**		**Teams**	**Submitted runs**
		Registered	*Actual*		
Affect, Creativity, and Style	ABSITA	30	11	7	20
	ITAMOJI	28	11	5	12
	IronITA	30	14	7	24
	GxG	15	9	3	50
Dialogue Systems	iLISTEN	16	4	2	2
	IDIAL	12	12	3	N/A
Hate Speech	AMI	39	16	10	73
	HaSpeeDe	40	32	9	55
Semantics4AI	NLP4FUN	17	4	2	3
	SUGAR	10	2	2	3
Total		237	115	50	242

A breakdown of the figures per task is shown in Table 1. With respect to the 2016 edition, we collected a significantly higher number of both preliminary registrations (237 registrations vs. 96 collected in 2016), teams (50 vs. 34 in 2016), and participants (115 [10] vs. 60 in 2016), that can be interpreted as a signal that we succeeded in reaching a wider audience of researchers interested in participating in the campaign as well as a further indication of the growth of the NLP community at large. This result could be also positively affected by the novelties introduced this year to involve cross-community participation, represented by the ITAMoji and AMI tasks. Indeed, of the 50 teams that submitted at least one run, 12 include researchers from foreign institutions. In addition to this, this year all tasks have received at least one submission.

A further aspect of the success for this edition can be due to the tasks themselves, especially the "Affect, Creativity and Style" and the "Hate Speech" tracks. Although these two tracks cover 60% of all tasks, they have collected the participation of 82% of the teams (41 teams). This is clearly a sign of growing interest in the NLP community at large in the study and analysis of new text types such as those produced in Social Media platforms and (on-line) user-generated content, also reflecting the outcome of the 2016 survey (Sprugnoli et al., 2016).

Finally, we consider the new protocol for the submission of participants' runs, consisting in three non-overlapping evaluation windows, as a further factor that may have positively impact the participation. Indeed, from the 2016 survey, it emerges that the main reasons for not participating in the evaluation either refer to personal issues or preferences ("I gave priority to other EVALITA tasks") also due to the difficulty of participating in the evaluation step of all tasks simultaneously, as the evaluation period was perceived as too short to enable participation to more than one task (Sprugnoli et al., 2016). Although appreciated by the EVALITA participants, this is not a major cause of the increased participation: out of 50 teams, only 6 have participated in more than one task.

Finally, it is compelling to open a reflection on the distinction between constrained and unconstrained submissions and participation to the tasks. Half of the tasks, namely ABSITA, ITAMOji, IronITA, and AMI, paid attention to this distinction and the other half did not take it into account. In early evaluation campaigns, the distinction used to be very relevant as it aimed at distinguishing the contribution of features or the learning approach from external sources of information, mainly intended as lexical

[10] Please note that the unique participants that also submitted a report are 68. This drop is mainly due to the participation to more than one task, resulting in the submission of only one report from the same team.

resources. In recent years, the spread and extensive use of pre-trained word embedding representations, especially as a strategy to initialize Neural Network architectures, challenges this distinction at its very heart. Furthermore, this distinction is also challenged by the development of multi-task learning architectures. A multi-task system could definitely represent an instance of an unconstrained system, although it exploits data from a different task, rather than a lexical resource or additional data annotated with the same information as that in the main task. As a contribution to the discussion on this topic, we think that proponents of tasks that aim at differentiating between constrained and unconstrained runs must specify what are the actual boundaries, in terms of extra training data, auxiliary tasks, use of word embeddings and lexical resources.

6 Final Remarks

For this edition of EVALITA we introduced novelties towards supporting reproducibility and cross-community engagement, towards advancement of methodology and techniques for natural language and speech processing tasks beyond the performance improvement, which is typically used as a metrics to assess state of the art approaches in benchmarking and shared task organization. In particular, the decision to award the best-system across tasks is inspired by this vision and aim at emphasizing the value of critical reflection and insightful discussion beyond the metric-based evaluation of participating systems.

In line with the suggestion provided by the organizers of the previous edition in 2016 (Basile et al., 2016; Sprugnoli et al., 2016), we introduced a novel organization of the evaluation period based on non-overlapping windows, in order to help those who want to participate in more than one task. This year EVALITA has reached a new milestone concerning the participation of industry. Overall, we have registered a total of 9 industrial participants: 7 directly participated to tasks, 6 of them submitted a paper, and 2 were involved as "targets" of an evaluations exercise (Cutugno et al., 2018).

Finally, a new trend that has emerged this year is the presence of tasks, GxG and HaSpeeDe, that aimed at testing the robustness of systems across text genres, further challenging the participants to develop their system. This "extra challenge" aspect is a new trend in EVALITA that started with the 2016 SENTIPOLC task (Barbieri et al., 2016), where the text genre was not changed but the test data was partially created using tweets that do not exactly match the selection procedure used for the creation of the training set.

Acknowledgments

We would like to thank our sponsors CELI[11], Google Research and the European Language and Resources Association (ELRA)[12] for their support to the event and to the best-system across task award. A further thank goes to ELRA for its offer and support in hosting the task datasets and systems' results. We also thanks Agenzia per l'Italia Digitale (AGID)[13] for its endorsement.

References

Francesco Barbieri, Basile Valerio, Croce Danilo, Nissim Malvina, Novielli Nicole, and Patti Viviana. 2016. Overview of the Evalita 2016 SENTIment POLarity Classification Task. In Pierpaolo Basile, Franco Cutugno, Malvina Nissim, Viviana Patti, and Rachele Sprugnoli, editors, *Proceedings of the 5th Evaluation Campaign of Natural Language Processing and Speech Tools for Italian (EVALITA 2016)*, Turin, Italy. CEUR.org.

Francesco Barbieri, Jose Camacho-Collados, Francesco Ronzano, Luis Espinosa Anke, Miguel Ballesteros, Valerio Basile, Viviana Patti, and Horacio Saggion. 2018. Semeval 2018 task 2: Multilingual emoji prediction. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 24–33. Association for Computational Linguistics.

Pierpaolo Basile and Nicole Novielli. 2018. Overview of the Evalita 2018 itaLIan Speech acT labEliNg (iLISTEN) Task. In Tommaso Caselli, Nicole Novielli, Viviana Patti, and Paolo Rosso, editors, *Proceedings of the 6th*

[11] https://www.celi.it/
[12] http://elra.info/en/
[13] https://www.agid.gov.it

evaluation campaign of Natural Language Processing and Speech tools for Italian (EVALITA'18), Turin, Italy. CEUR.org.

Pierpaolo Basile, Franco Cutugno, Malvina Nissim, Viviana Patti, and Rachele Sprugnoli. 2016. EVALITA 2016: Overview of the 5th Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. In Pierpaolo Basile, Franco Cutugno, Malvina Nissim, Viviana Patti, and Rachele Sprugnoli, editors, *Proceedings of the 5th Evaluation Campaign of Natural Language Processing and Speech Tools for Italian (EVALITA 2016)*, Turin, Italy. CEUR.org.

Pierpaolo Basile, Malvina Nissim, Rachele Sprugnoli, Viviana Patti, and Francesco Cutugno. 2017. Evalita goes social: Tasks, data, and community at the 2016 edition. *Italian Journal of Computational Linguistics*, 3(1).

Pierpaolo Basile, Valerio Basile, Danilo Croce, and Marco Polignano. 2018a. Overview of the EVALITA 2018 Aspect-based Sentiment Analysis task (ABSITA). In Tommaso Caselli, Nicole Novielli, Viviana Patti, and Paolo Rosso, editors, *Proceedings of the 6th evaluation campaign of Natural Language Processing and Speech tools for Italian (EVALITA'18)*, Turin, Italy. CEUR.org.

Pierpaolo Basile, Marco de Gemmis, Lucia Siciliani, and Giovanni Semeraro. 2018b. Overview of the EVALITA 2018 Solving language games (NLP4FUN) Task. In Tommaso Caselli, Nicole Novielli, Viviana Patti, and Paolo Rosso, editors, *Proceedings of the 6th evaluation campaign of Natural Language Processing and Speech tools for Italian (EVALITA'18)* , Turin, Italy. CEUR.org.

Cristina Bosco, Felice Dell'Orletta, Fabio Poletto, Manuela Sanguinetti, and Maurizio Tesconi. 2018. Overview of the EVALITA 2018 Hate Speech Detection Task. In Tommaso Caselli, Nicole Novielli, Viviana Patti, and Paolo Rosso, editors, *Proceedings of the 6th evaluation campaign of Natural Language Processing and Speech tools for Italian (EVALITA'18)* , Turin, Italy. CEUR.org.

Alessandra Teresa Cignarella, Simona Frenda, Valerio Basile, Cristina Bosco, Viviana Patti, and Paolo Rosso. 2018. Overview of the EVALITA 2018 Task on Irony Detection in Italian Tweets (IronITA). In Tommaso Caselli, Nicole Novielli, Viviana Patti, and Paolo Rosso, editors, *Proceedings of the 6th evaluation campaign of Natural Language Processing and Speech tools for Italian (EVALITA'18)* , Turin, Italy. CEUR.org.

Francesco Cutugno, Maria Di Maro, Sara Falcone, Marco Guerini, Bernardo Magnini, and Antonio Origlia. 2018. Overview of the EVALITA 2018 Evaluation of Italian DIALogue systems (IDIAL) Task. In Tommaso Caselli, Nicole Novielli, Viviana Patti, and Paolo Rosso, editors, *Proceedings of the 6th evaluation campaign of Natural Language Processing and Speech tools for Italian (EVALITA'18)* , Turin, Italy. CEUR.org.

Felice Dell'Orletta and Malvina Nissim. 2018. Overview of the EVALITA 2018 Cross-Genre Gender Prediction (GxG) Task. In Tommaso Caselli, Nicole Novielli, Viviana Patti, and Paolo Rosso, editors, *Proceedings of the 6th evaluation campaign of Natural Language Processing and Speech tools for Italian (EVALITA'18)* , Turin, Italy. CEUR.org.

Maria Di Maro, Antonio Origlia, and Francesco Cutugno. 2018. Overview of the EVALITA 2018 Spoken Utterances Guiding Chef's Assistant Robots (SUGAR) Task. In Tommaso Caselli, Nicole Novielli, Viviana Patti, and Paolo Rosso, editors, *Proceedings of the 6th evaluation campaign of Natural Language Processing and Speech tools for Italian (EVALITA'18)* , Turin, Italy. CEUR.org.

Elisabetta Fersini, Debora Nozza, and Paolo Rosso. 2018a. Overview of the Evalita 2018 Task on Automatic Misogyny Identification (AMI). In Tommaso Caselli, Nicole Novielli, Viviana Patti, and Paolo Rosso, editors, *Proceedings of the 6th evaluation campaign of Natural Language Processing and Speech tools for Italian (EVALITA'18)* , Turin, Italy. CEUR.org.

Elisabetta Fersini, Paolo Rosso, and Maria Anzovino. 2018b. Overview of the Task on Automatic Misogyny Identification at IberEval 2018. In Paolo Rosso, Julio Gonzalo, Raquel Martínez, Soto Montalvo, and Jorge Carrillo de Albornoz, editors, *Proceedings of the Third Workshop on Evaluation of Human Language Technologies for Iberian Languages (IberEval 2018) co-located with 34th Conference of the Spanish Society for Natural Language Processing (SEPLN 2018), Sevilla, Spain, September 18th, 2018.*, volume 2150 of *CEUR Workshop Proceedings*, pages 214–228. CEUR-WS.org.

Francesco Ronzano, Francesco Barbieri, Endang Wahyu Pamungkas, Viviana Patti, and Francesca Chiusaroli. 2018. Overview of the EVALITA 2018 Italian Emoji Prediction (ITAMoji) Task. In Tommaso Caselli, Nicole Novielli, Viviana Patti, and Paolo Rosso, editors, *Proceedings of the 6th evaluation campaign of Natural Language Processing and Speech tools for Italian (EVALITA'18)* , Turin, Italy. CEUR.org.

Rachele Sprugnoli, Viviana Patti, and Cutugno Franco. 2016. Raising Interest and Collecting Suggestions on the EVALITA Evaluation Campaign. In Pierpaolo Basile, Franco Cutugno, Malvina Nissim, Viviana Patti, and Rachele Sprugnoli, editors, *Proceedings of the 5th Evaluation Campaign of Natural Language Processing and Speech Tools for Italian (EVALITA 2016)*, Turin, Italy. CEUR.org.

The Search for Emotions, Creativity, and Fairness in Language

Saif M. Mohammad
National Research Council, Canada
Saif.Mohammad@nrc-cnrc.gc.ca

Emotions are central to human experience, creativity, and behavior. They are crucial for organizing meaning and reasoning about the world we live in. They are ubiquitous and everyday, yet complex and nuanced. In this talk, I will describe our work on the search for emotions in language – by humans (through data annotation projects) and by machines (in automatic emotion detection systems).

I will outline ways in which emotions can be represented, challenges in obtaining reliable annotations, and approaches that lead to high-quality annotations. The lexicons thus created have entries for tens of thousands of terms. They provide fine-grained scores for basic emotions as well as for valence, arousal, and dominance (argued by some to be the core dimensions of meaning). They have wide-ranging applications in natural language processing, psychology, social sciences, digital humanities, and computational creativity. I will highlight some of the applications we have explored in literary analysis and automatic text-based music generation. I will also discuss new sentiment analysis tasks such as inferring fine-grained emotion intensity and stance from tweets, as well as detecting emotions evoked by art. I will conclude with work on quantifying biases in the way language is used and the impact of such biases on automatic emotion detection systems. From social media to home assistants, from privacy concerns to neuro-cognitive persuasion, never has natural language processing been more influential, more fraught with controversy, and more entrenched in everyday life. Thus as a community, we are uniquely positioned to make substantial impact by building applications that are not only compelling and creative but also facilitators of social equity and fairness.

Overview of the EVALITA 2018 Aspect-based Sentiment Analysis task (ABSITA)

Pierpaolo Basile
University of Bari Aldo Moro
pierpaolo.basile@uniba.it

Valerio Basile
University of Turin
basile@di.uniroma1.it

Danilo Croce
University of Rome "Tor Vergata"
croce@info.uniroma2.it

Marco Polignano
University of Bari Aldo Moro
marco.polignano@uniba.it

Abstract

English. ABSITA is the Aspect-based Sentiment Analysis task at EVALITA 2018 (Caselli et al., 2018). This task aimed to foster research in the field of aspect-based sentiment analysis within the Italian language: the goal is to identify the aspects of given target entities and the sentiment expressed for each aspect. Two subtasks are defined, namely Aspect Category Detection (ACD) and Aspect Category Polarity (ACP). In total, 20 runs were submitted by 7 teams comprising 11 total individual participants. The best system achieved a micro F1-score of 0.810 for ACD and 0.767 for ACP.

Italiano. *ABSITA è l'esercizio di valutazione di aspect-based sentiment analysis di EVALITA 2018 (Caselli et al., 2018). Il compito ha l'obiettivo di promuovere la ricerca nel campo della sentiment analysis per lingua italiana: ai partecipanti è stato richiesto di identificare gli aspetti rilevanti per le entitá fornite come input e la sentiment espressa per ognuno di essi. In particolare abbiamo definito come sottotask l'Aspect Category Detection (ACD) e l'Aspect Category Polarity (ACP). In totale, sono state presentate 20 soluzioni di 7 team composti in totale da 11 singoli partecipanti. Il miglior sistema ha ottenuto un punteggio di micro F1 di 0,810 per ACD e 0,767 per ACP.*

1 Introduction

In recent years, many websites started offering a high level interaction with users, who are no more a passive audience, but can actively produce new content. For instance, platforms like Amazon[1] or TripAdvisor[2] allow people to express their opinions on products, such as food, electronic items, clothes, and services, such as hotels and restaurants.

In such a social context, *Sentiment Analysis* (SA) is the task of automatically extract subjective opinions from a text. In its most basic form, a SA system takes in input a text written in natural language and assign it a label indicating whether the text is expressing a positive or negative sentiment, or neither (neutral, or objective, text). However, reviews are often quite detailed in expressing the reviewer's opinion on several aspects of the target entity. *Aspect-based Sentiment Analysis* (ABSA) is an evolution of Sentiment Analysis that aims at capturing the aspect-level opinions expressed in natural language texts (Liu, 2007).

At the international level, ABSA was introduced as a shared task at SemEval, the most prominent evaluation campaign in the Natural Language Processing field, in 2014 (SE-ABSA14), providing a benchmark dataset of reviews in English (Pontiki et al., 2014). Datasets of computer laptops and restaurant reviews were annotated with aspect terms (both fine-grained, e.g. "hard disk", "pizza", and coarse-grained, e.g., "food") and their polarity (positive or negative).

The task was repeated in SemEval 2015 (SE-ABSA15) and 2016 (SE-ABSA16), aiming to facilitate more in-depth research by providing a new ABSA framework to investigate the relations between the identified constituents of the expressed opinions and growing up to include languages other than English and different domains (Pontiki et al., 2015; Pontiki et al., 2016).

ABSITA (Aspect-based Sentiment Analysis on Italian) aims at providing a similar evaluation with respect to texts in Italian. In a nutshell, partic-

[1] http://www.amazon.com
[2] http://www.tripadvisor.com

ipants are asked to detect within sentences (expressing opinions about accommodation services) some of the aspects considered by the writer. These aspects belongs to a close set ranging from the cleanliness of the room to the price of the accommodation. Moreover, for each detected aspect, participants are asked to detect a specific polarity class, expressing appreciation or criticism towards it.

During the organization of the task, we collected a dataset composed of more than 9,000 sentences and we annotated them with aspects and polarity labels. During the task, 20 runs were submitted by 7 teams comprising 11 individual participants.

In the rest of the paper Section 2 provides a detailed definition of the task. Section 3 describes the dataset made available in the evaluation campaign, while Section 4 reports the official evaluation measures. In Section 5 and 6, the results obtained by the participants are reported and discussed, respectively. Finally, Section 7 derives the conclusions.

2 Definition of the task

In ABSITA, Aspect-based Sentiment Analysis is decomposed as a cascade of two subtasks: **Aspect Category Detection** (ACD) and **Aspect Category Polarity** (ACP). For example, let us consider the sentence describing an hotel:

I servizi igienici sono puliti e il personale cordiale e disponibile. (*Toilets are clean but the staff is not friendly nor helpful.*)

In the ACD task, one or more "aspect categories" evoked in a sentence are identified, e.g. the `pulizia` (`cleanliness`) and `staff` categories in sentence 2. In the **Aspect Category Polarity** (ACP) task, the polarity of each expressed category is recognized, e.g. a `positive` category polarity is expressed concerning the `pulizia` category while it is `negative` if considering the `staff` category.

In our evaluation framework, the set of aspect categories is known and given to the participants, so the ACD task can be seen as a multi-class, non-exclusive classification task where each input text has to be classified as evoking or not each aspect category. The participant systems are asked to return a binary vector where each dimension corresponds to an aspect category and the values 0 (`false`) and 1 (`true`) indicate whether each aspect has been detected in the text. Table 1 shows examples of annotation for the ACD task.

For the ACP task, the input is the review text paired with the set of aspects identified in the text within the ACD subtask, and the goal is to assign polarity labels to each of the aspect category. Two binary polarity labels are expected for each aspect: `POS` an `NEG`, indicating a positive and negative sentiment expressed towards a specific aspect, respectively. Note that the two labels are not mutually exclusive: in addition to the annotation of *positive* aspects (`POS:true`, `NEG:false`) and *negative* aspects (`POS:false`, `NEG:true`), there can be aspects with no polarity, or *neutral* polarity (`POS:false`, `NEG:false`). This is also the default polarity annotation for the aspects that are not detected in a text. Finally, the polarity of an aspect can be *mixed* (`POS:true`, `NEG:true`), in cases where both sentiments are expressed towards a certain aspect in a text. Table 2 summarizes the possible annotations with examples.
The participants could choose to submit only the results of the ACD subtask, or both tasks. In the latter case, the output of the ACD task is used as input for the ACP. As a constraint on the results submitted for the ACP task, the polarity of an aspect for a given sentence can be different than (`POS:false`, `NEG:false`) only if the aspect is detected in the ACD step.

3 Dataset

The data source chosen for creating the ABSITA datasets is the popular website *booking.com*[3]. The platform allows users to share their opinions about hotels visited through a positive/negative textual review and a fine-grain rating system that can be used for assigning a score to each different aspect: cleanliness, comfort, facilities, staff, value for money, free/paid WiFi, location. Therefore, the website provides a large number of reviews in many languages.

We extracted the textual reviews in Italian, labeled on the website with one of the eighth considered aspects. The dataset contains reviews left by users for hotels situated in several main Italian cities such as Rome, Milan, Naples, Turin, Bari, and more. We split the reviews into groups of sentences which describe the positive and the negative characteristics of the selected hotel. The reviews have been collected between the 16th and

[3] `https://www.booking.com`

Sentence	CLEANLINESS	STAFF	COMFORT	LOCATION
I servizi igienici sono puliti e il personale cordiale e disponibile	1	1	0	0
La posizione è molto comoda per il treno e la metro.	0	0	0	1
Ottima la disponibilitá del personale, e la struttura della stanza	0	1	1	0

Table 1: Examples of categories detection ACD.

Sentence	Aspect	POS	NEG
Il bagno andrebbe ristrutturato	CLEANLINESS	0	0
Camera pulita e spaziosa.	CLEANLINESS	1	0
Pulizia della camera non eccelsa.	CLEANLINESS	0	1
Il bagno era pulito ma lasciava un po' a desiderare	CLEANLINESS	1	1

Table 2: Examples of polarity annotations with respect to the *cleanliness* aspect.

the 17th of April 2018 using Scrapy[4], a Python web crawler. We collect in total 4,121 distinct reviews in Italian language.

The reviews have been manually checked to verify the annotation of the aspects provided by booking.com, and to add missing links between sentences and aspects. We started by annotating a small portion of the whole dataset split by sentences (250 randomly chosen sentences) using four annotators (the task organizers) in order to check the agreement of the annotation. For the ACD task, we asked the annotators to answer straightforward questions in the form of "Is aspect X mentioned in the sentence Y?" (Tab. 1).

The set of italian aspects is the direct translation of those booking.com: PULIZIA (cleanliness), COMFORT, SERVIZI (amenities), STAFF, QUALITA-PREZZO (value), WIFI (wireless Internet connection) and POSIZIONE (location). Similarly, for the ACP subtask, the annotation is performed at sentence level, but with the set of aspects already provided by the ACD annotation, and checkboxes to indicate positive and negative polarity of each aspect (Tab. 2). The result of the pilot annotation has been used to compute an inter-annotator agreement measure, in order to understand if it was possible to allow annotators to work independently each other on a different set of sentences. We found agreement ranging from 82.8% to 100% with an average value of 94.4% obtained counting the number of sentences annotated with the same label by all the annotators.

In order to complete the annotation, we assigned different 1,000 reviews to each annotator (about 2,500 sentences on average). We split the dataset among the annotators so that each of them received a uniformly balanced distribution of positive and negative aspects, based on the scores provided by the original review platform. Incomplete, irrelevant, and incomprehensible sentences have been discarded from the dataset during the annotation. At the end of the annotation process, we obtained the gold standard dataset with the associations among sentence, sentiment and aspect. The entire annotation process took a few weeks to complete. The positive and negative polarities are annotated independently, thus for each aspect the four sentiment combination discussed in Section 2 are possible: *positive*, *negative* , *neutral* and *mixed*. The resulting classes are: *cleanliness_positive, cleanliness_negative, comfort_positive, comfort_negative, amenities_positive, amenities_negative, staff_positive, staff_negative, value_positive, value_negative, wifi_positive, wifi_negative, location_positive, location_negative, other_positive, other_negative*. For each aspect, the sentiment is encoded in two classes:

- *negative* = (**_positive = 0, *_negative = 1*)
- *positive* = (**_positive = 1, *_negative = 0*)
- *neutral* = (**_positive = 0, *_negative = 0*)
- *mixed* = (**_positive = 1, *_negative = 1*)

Please note that the special topic, OTHER has been added for completeness, to annotate sentences with opinions on aspects not among the seven considered by the task. The aspect OTHER is provided additionally and it is not part of the evaluation of results provided for the task.

We released the data in Comma-separated Value format (CSV) with UTF-8 encoding and semicolon as separator. The first attribute is the id of the review. Note that in booking.com the order of positive and negative sentences is strictly defined and this can make too easy the task. To overcome

[4] https://scrapy.org

Dataset	Description	#Sentences
Trial set	Trial dataset containing a small set of features used for checking the format of the file format	*30* *0.34% of Total*
Training set	The dataset contains sentences provided for training. They have been selected using a random stratification of the whole dataset.	*6,337* *69.75% of Total*
Test set	The dataset contains sentences provided for testing. They contains sentences without the annotations of aspects.	*2,718* *29.91% of Total*

Table 3: List of datasets released for the ABSITA task at EVALITA 2018.

Dataset	clean_pos	comf_pos	amen_pos	staff_pos	value_pos	wifi_pos	loca_pos
Trial set	2	8	6	3	1	1	5
Training set	504	978	948	937	169	43	1,184
Test set	193	474	388	411	94	18	526

Dataset	clean_neg	comf_neg	amen_neg	staff_neg	value_neg	wifi_neg	loca_neg
Trial set	1	2	3	1	1	0	1
Training set	383	1,433	920	283	251	86	163
Test set	196	666	426	131	126	52	103

Table 4: Distribution of the sentences in the datasets among the aspects and polarities.

this issue, we randomly assign for each sentence a new position in the review. As a consequence, the final positional id showed in the data file do not reflect the real order of the sentences in the review. The text of the sentence is provided at the end of the line and delimited by ". It is preceded by three binary values for each aspect indicating respectively: the presence in the sentence ($aspectX\_presence$:0/1), the positive polarity for that aspect ($aspectX\_pos$:0/1) and finally the negative polarity ($aspectX\_neg$:0/1). Fig. 1 shows an example of the annotated dataset in the proposed format.

The list of the datasets released for the task is provided in Tab. 3 and the distribution of the sentences among aspects and polarity is provided in Tab. 4. The subdivision adopted for it is respectively 0.34%, 69.75%, 29,91% for trial, training and test data. The datasets can be freely downloaded from `http://sag.art.uniroma2.it/absita/` and reused in non-commercial projects and researches. After the submission deadline, we also distributed the gold standard test set and evaluation script.

4 Evaluation measures and baselines

We evaluate the ACD and ACP subtasks separately by comparing the classifications provided by the participant systems to the gold standard annotations of the test set. For the ACD task, we compute Precision, Recall and F_1-score defined as: $F1_a = \frac{2P_aR_a}{P_a+R_a}$, where Precision ($P_a$) and Recall ($R_a$) are defined as: $P_a = \frac{|S_a \cap G_a|}{|S_a|}; R_a = \frac{|S_a \cap G_a|}{|G_a|}$. Here S_a is the set of aspect category annotations that a system returned for all the test sentences, and G_a is the set of the gold (correct) aspect category annotations. For instance, if a review is labeled in the gold standard with the two aspects $G_a = \{\text{CLEANLINESS}, \text{STAFF}\}$, and the system predicts the two aspects $S_a = \{\text{CLEANLINESS}, \text{COMFORT}\}$, we have that $|S_a \cap G_a| = 1$, $|G_a| = 2$ and $|S_a| = 2$ so that $P_a = \frac{1}{2}$, $R_a = \frac{1}{2}$ and $F1_a = \frac{1}{2}$. For the ACD task the baseline will be computed by considering a system which assigns the most frequent aspect category (estimated over the training set) to each sentence.

For the ACP task we evaluate the entire chain, thus considering both the aspect categories detected in the sentences together with their corresponding polarity, in the form of $(aspect, polarity)$ pairs. We again compute Precision, Recall and F_1-score now defined as $F1_p = \frac{2P_pR_p}{P_p+R_p}$. Precision ($P_p$) and Recall ($R_p$) are defined as $P_p = \frac{|S_p \cap G_p|}{|S_p|}; R_p = \frac{|S_p \cap G_p|}{|G_p|}$, where S_p is the set of $(aspect, polarity)$ pairs that a system returned for all the test sentences, and G_a is the set of the gold (correct) pairs annotations. For instance, if a review is labeled in the gold standard with the pairs $G_p = \{(\text{CLEANLINESS}, POS), (\text{STAFF}, POS)\}$, and the system predicts the three pairs $S_p = \{(\text{CLEANLINESS}, POS), (\text{CLEANLINESS}, NEG), (\text{COMFORT}, POS)\}$, we have that $|S_p \cap G_p| = 1$, $|G_p| = 2$ and $|S_p| = 3$ so that $P_a = \frac{1}{3}$, $R_a = \frac{1}{2}$ and $F1_a = 0.28$.

For the ACP task, the baseline is computed by considering a system which assigns the most fre-

```
sentence_id; aspect1_presence; aspect1_pos; aspect1_neg; ...; sentence
201606240;0;0;0;0;0;0;0;0;0;0;0;0;1;1;0;0;0;0;1;1;0;"Considerato il prezzo e per una sola notte,va    ..."
201606241;1;0;1;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;"Almeno i servizi igienici andrebbero rivisti e   ..."
201606242;0;0;0;1;0;1;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;"La struttura purtroppo \'e vecchia e ci vorrebbero ..."
```

Figure 1: Sample of the annotated dataset in CSV format.

System	Micro-P	Micro-R:	Micro-F1
ItaliaNLP_1	0.8397	**0.7837**	**0.8108**
gw2017_1	0.8713	0.7504	0.8063
gw2017_2	0.8697	0.7481	0.8043
X2Check_gs	0.8626	0.7519	0.8035
UNIPV	0.8819	0.7378	0.8035
X2Check_w	**0.8980**	0.6937	0.7827
ItaliaNLP_2	0.8658	0.6970	0.7723
SeleneBianco	0.7902	0.7181	0.7524
VENSES_1	0.6232	0.6093	0.6162
VENSES_2	0.6164	0.6134	0.6149
ilc_2	0.5443	0.5418	0.5431
ilc_1	0.6213	0.4330	0.5104
mfc baseline	0.4111	0.2866	0.3377

Table 5: Results of the submissions for the ACD subtask.

System	Micro-P	Micro-R:	Micro-F1
ItaliaNLP_1	0.8264	0.7161	**0.7673**
UNIPV	**0.8612**	0.6562	0.7449
gw2017_2	0.7472	**0.7186**	0.7326
gw2017_1	0.7387	0.7206	0.7295
ItaliaNLP_2	0.8735	0.5649	0.6861
SeleneBianco	0.6869	0.5409	0.6052
ilc_2	0.4123	0.3125	0.3555
ilc_1	0.5452	0.2511	0.3439
mfc baseline	0.2451	0.1681	0.1994

Table 6: Results of the submissions for the ACP subtask.

quent $(aspect, polarity)$ pair (estimated over the training set) to each sentence.

We produced separate rankings for the tasks, based on the F_1 scores. Participants who submitted only the result of the ACD task appear in the first ranking only.

5 Results

We received submissions from several teams that participated in past editions of EVALITA, in particular to the SENTIPOLC (Sentiment Polarity Classification (Barbieri et al., 2016)) and NEEL-it (Named Entity Recognition (Basile et al., 2016)), but also some new entries in the community. In total, 20 runs were submitted by 7 teams comprising 11 individual participants. The task allowed participant teams to send up to 2 submissions from each team. In particular, 12 runs were submitted to ACD task and 8 runs to the ACP task.

We also provide the result of a baseline system that assigns to each instance the most frequent class in each task, i.e., the aspect (COMFORT) and polarity (*positive*) for that aspect, according to the frequency of classes in the training set. The results of the submissions for the two tasks, and the baseline (namely mfc baseline), are reported in Tab. 5 and Tab. 6. Of the seven teams who participated to the ACD task, five teams also participated to the ACP task.

The results obtained by the teams largely outperform the baseline demonstrating the efficacy of the solutions proposed and the affordability of all two tasks. The results obtained for the ACD task (Tab. 5) show a small range of variability, at least in the first part of the ranking (the top results are concentrated around a F1 score value of 0.80). On the contrary, the values of precision and recall show higher variability, indicating significant difference among the proposed approaches.

6 Discussion

The teams of the ABSITA challenge have been invited to describe their solution in a technical report and to fill in a questionnaire, in order to gain an insight on their approaches and to support their replicability. Five systems (*ItaliaNLP*, *gw2017*, *X2Check*, *UNIPV*, *SeleneBianco*) are based on supervised machine learning, that is, all the systems for which we have access to the implementation details, with the exception of *VENSES*, which is a rule-based unsupervised system. Among the system that use supervised approaches, three systems (*ItaliaNLP*, *gw2017*, *UNIPV*) employ deep learning (in particular LTSM networks, often in their bi-directional variant).

All runs submitted can be considered "constrained runs", that is, the systems were trained on the provided data set only.

Besides additional training data, some systems employ different kind of external resources. Among these, pre-trained word embeddings are used as word representations by *UNIPV* (Fast-

text[5]) and *gw2017* (word embeddings provided by the SpaCy framework[6]). The system of *ItaliaNLP* employs word embedding created from the ItWaC corpus (Baroni et al., 2009) and corpus extracted from Booking.com.

Some of the systems are ABSA extensions built on top of custom or pre-existing NLP pipelines. This is the case for *ItaliaNLP*, *VENSES* and *X2Check*. Other systems make use of off-the-shelf NLP tools for preprocessing the data, such as SpaCy (*gw2017*, *UNIPV*) and Freeling[7] (*SeleneBianco*).

Finally, additional resources used by the systems often include domain-specific or affective lexicons. *ItaliaNLP* employed the MPQA affective lexicon (Wilson et al., 2005), and further developed an affective lexicon from a large corpus of tweets by distant supervision. The *UNIPV* system makes use of the affective lexicon for Italian developed in the framework of the OpeNER project[8].

In the ACD task, the precision of the second ranked system (*gw2017*) is significantly higher than that of the first system (*ItaliaNLP*), although the latter ranks at the top because of a higher recall. This unbalance between precision and recall is mainly due to the high number of aspect that can be assigned at the same time to a sentence: a system returning too many aspects is exposed to low precision but higher recall, while a more conservative system would achieve the opposite situation. Further details about the systems developed for the task can be found in the technical reports of the partecipants: ItaliaNLP (Cimino et al., 2018), *UNIPV* (Nicola, 2018), *VENSES* (Delmonte, 2018), *X2Check* (Di Rosa and Durante, 2018), *gw2017* (Bennici and Portocarrero, 2018)

7 Conclusion

The large availability of user-generated contents over the Web that characterizes the current tendencies of virtually sharing opinions with others has promoted the diffusion of platforms able to analyze and reuse them for personalized services. A challenging task is the analysis of the users' opinions about a product, service or topic of discussion. In particular, the ABSA (Aspect-based Sentiment Analysis) task concerns the association of a polarity (positive, negative, neutral/objective) to the piece of the sentence that refers to an aspect of interest. In ABSITA, we proposes to automatically extract users' opinions about aspects in hotel rewievs. The complexity of the task has been successfully faced by the solutions submitted to the task. Systems that used supervised machine learning approaches, based on semantic and morphosyntactic features representation of textual contents, demonstrate encouraging performances in the task. Good results have also been obtained using rule-based systems, even though they suffer from generalization issues and need to be tailored on the set of sentences to classify. The decision to use additional resources as additional lexicons in conjunction with semantic word embeddings have been demonstrated to be successful. More details about the implementation of the systems that participated in the task can be found in their specific reports. In conclusion, we consider the ABSITA 2018 task a success and an improvement of state of the art for the ABSA task in the Italian language.

[5] https://github.com/facebookresearch/fastText/blob/master/pretrained-vectors.md

[6] https://spacy.io/

[7] http://nlp.lsi.upc.edu/freeling/node/

[8] https://github.com/opener-project/\\VU-sentiment-lexicon

References

Francesco Barbieri, Valerio Basile, Danilo Croce, Malvina Nissim, Nicole Novielli, and Viviana Patti. 2016. Overview of the Evalita 2016 SENTIment POLarity Classification Task. In *Proceedings of Third Italian Conference on Computational Linguistics (CLiC-it 2016) & Fifth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2016)*, Naples, Italy, December.

Marco Baroni, Silvia Bernardini, Adriano Ferraresi, and Eros Zanchetta. 2009. The wacky wide web: a collection of very large linguistically processed web-crawled corpora. *Language Resources and Evaluation*, 43:209–226.

P. Basile, A. Caputo, A.L. Gentile, and G. Rizzo. 2016. Overview of the evalita 2016 named entity recognition and linking in italian tweets (neel-it) task. In *5th Evaluation Campaign of Natural Language Processing and Speech Tools for Italian (EVALITA 2016)*, Napoli, Italia, 12/2016.

Mauro Bennici and Xileny Seijas Portocarrero. 2018. Ensemble for aspect-based sentiment analysis. In Tommaso Caselli, Nicole Novielli, Viviana Patti, and Paolo Rosso, editors, *Proceedings of the 6th evaluation campaign of Natural Language Processing and Speech tools for Italian (EVALITA'18)*, Turin, Italy. CEUR.org.

Tommaso Caselli, Nicole Novielli, Viviana Patti, and Paolo Rosso. 2018. Evalita 2018: Overview of the 6th evaluation campaign of natural language processing and speech tools for italian. In Tommaso Caselli, Nicole Novielli, Viviana Patti, and Paolo Rosso, editors, *Proceedings of Sixth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2018)*, Turin, Italy. CEUR.org.

Andrea Cimino, Lorenzo De Mattei, and Felice Dell'Orletta. 2018. Multi-task Learning in Deep Neural Networks at EVALITA 2018. In Tommaso Caselli, Nicole Novielli, Viviana Patti, and Paolo Rosso, editors, *Proceedings of the 6th evaluation campaign of Natural Language Processing and Speech tools for Italian (EVALITA'18)*, Turin, Italy. CEUR.org.

Rodolfo Delmonte. 2018. Itvenses - a symbolic system for aspect-based sentiment analysis. In Tommaso Caselli, Nicole Novielli, Viviana Patti, and Paolo Rosso, editors, *Proceedings of the 6th evaluation campaign of Natural Language Processing and Speech tools for Italian (EVALITA'18)*, Turin, Italy. CEUR.org.

Emanuele Di Rosa and Alberto Durante. 2018. Aspect-based sentiment analysis: X2check at absita 2018. In Tommaso Caselli, Nicole Novielli, Viviana Patti, and Paolo Rosso, editors, *Proceedings of the 6th evaluation campaign of Natural Language Processing and Speech tools for Italian (EVALITA'18)*, Turin, Italy. CEUR.org.

Bing Liu. 2007. *Web data mining*. Springer.

Giancarlo Nicola. 2018. Bidirectional attentional lstm for aspect based sentiment analysis on italian. In Tommaso Caselli, Nicole Novielli, Viviana Patti, and Paolo Rosso, editors, *Proceedings of the 6th evaluation campaign of Natural Language Processing and Speech tools for Italian (EVALITA'18)*, Turin, Italy. CEUR.org.

Maria Pontiki, Dimitris Galanis, John Pavlopoulos, Harris Papageorgiou, Ion Androutsopoulos, and Suresh Manandhar. 2014. Semeval-2014 task 4: Aspect based sentiment analysis. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 27–35, Dublin, Ireland, August. Association for Computational Linguistics and Dublin City University.

Maria Pontiki, Dimitris Galanis, Haris Papageorgiou, Suresh Manandhar, and Ion Androutsopoulos. 2015. Semeval-2015 task 12: Aspect based sentiment analysis. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 486–495, Denver, Colorado, June. Association for Computational Linguistics.

Maria Pontiki, Dimitrios Galanis, Haris Papageorgiou, Ion Androutsopoulos, Suresh Manandhar, Mohammad AL-Smadi, Mahmoud Al-Ayyoub, Yanyan Zhao, Bing Qin, Orphée De Clercq, Véronique Hoste, Marianna Apidianaki, Xavier Tannier, Natalia Loukachevitch, Evgeny Kotelnikov, Nuria Bel, Salud María Jiménez-Zafra, and Gülşen Eryiğit. 2016. SemEval-2016 task 5: Aspect based sentiment analysis. In *Proceedings of the 10th International Workshop on Semantic Evaluation*, SemEval '16, San Diego, California, June. Association for Computational Linguistics.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the conference on human language technology and empirical methods in natural language processing*, pages 347–354. Association for Computational Linguistics.

Overview of the EVALITA 2018 Italian Emoji Prediction (ITAMoji) Task

Francesco Ronzano
Universitat Pompeu Fabra, Spain
Hospital del Mar Medical Research Center
Barcelona, Spain
francesco.ronzano@upf.edu

Francesco Barbieri
Universitat Pompeu Fabra
Barcelona, Spain
francesco.barbieri@upf.edu

Endang Wahyu Pamungkas,Viviana Patti
Department of Computer Science
University of Turin, Italy
{pamungka,patti}@di.unito.it

Francesca Chiusaroli
Department of Humanities
Università di Macerata, Italy
f.chiusaroli@unimc.it

Abstract

English. The Italian Emoji Prediction task (ITAmoji) is proposed at EVALITA 2018 evaluation campaign for the first time, after the success of the twin Multilingual Emoji Prediction Task, organized in the context of SemEval-2018 in order to challenge the research community to automatically model the semantics of emojis in Twitter. Participants were invited to submit systems designed to predict, given an Italian tweet, its most likely associated emoji, selected in a wide and heterogeneous emoji space. Twelve runs were submitted at ITAmoji by five teams. We present the data sets, the evaluation methodology including different metrics and the approaches of the participating systems. We also present a comparison between the performance of automatic systems and humans solving the same task. Data and further information about this task can be found at: https://sites.google.com/view/itamoji/.

Italiano. *Il task italiano per la predizione degli emoji in Twitter (ITAmoji) viene proposto nell'ambito della campagna di valutazione di Evalita 2018 per la prima volta, dopo il successo del task gemello, il* Multilingual Emoji Prediction Task*, proposto a Semeval-2018 per stimolare la comunità di ricerca a costruire modelli computazionali della semantica delle emoji in Twitter. I partecipanti sono stati invitati a costruire sistemi disegnati per predire l'emoji piú probabile dato un tweet in italiano, selezionandola in uno spazio ampio e eterogeneo di emoji. In ITAmoji sono stati valutati i risultati di dodici sistemi di predizione di emoji messi a punto da cinque gruppi di lavoro. Presentiamo qui i dataset, la metodologia di valutazione (che include diverse metriche) e gli approcci dei sistemi che hanno partecipato. Presentiamo inoltre una riflessione sui risultati ottenuti in tale task da sistemi automatici e umani.*

1 Introduction

During the last decade the use of emoji has increasingly pervaded social media platforms by providing users with a rich set of pictograms useful to visually complement and enrich the expressiveness of short text messages. Nowadays this novel, visual way of communication represents a *de facto* standard in a wide range of social media platforms including fully-fledged portals for user-generated contents like Twitter, Facebook and Instagram as well as instant-messaging services like WhatsApp. As a consequence, the possibility to effectively interpret and model the semantics of emojis has become an essential task to deal with when we analyze social media contents.

Even if over the last few years the study of this new form of language has been receiving a growing attention, at present the body of investigations that deal with emojis is still scarce, especially when we consider their characterization from a Natural Language Processing (NLP) perspective. While there are notable exceptions which study the semantics of emojis and their usage (Barbieri et al., 2016a; Barbieri et al., 2018b; Aoki and Uchida, 2011; Eisner et al., 2016; Ljubešić and Fišer, 2016), reflecting also on their informative behaviour (Donato and Paggio, 2017; Donato and Paggio, 2018), or their sentiment (Novak et al., 2015), the interplay between text-based mes-

sages and emojis remains still explored only by a small number of studies. Among these investigations there is the analysis of emoji predictability by (Barbieri et al., 2017), which proposed a neural model to predict the most likely emoji to appear in a text message (tweet). The task resulted to be hard, as emojis encode multiple meanings (Barbieri et al., 2016b). Related to this, in the context of the International Workshop on Semantic Evaluation (SemEVAL 2018), the Multilingual Emoji Prediction Task (Barbieri et al., 2018a) has been organized in order to challenge the research community to automatically model the semantics of emojis occurring in English and Spanish Twitter messages. The task was very successful, with 49 teams participating in the English subtask and 22 in the Spanish subtask. This motivated us to propose the shared task also for the Italian language in the context of the Evalita 2018 evaluation campaign (Caselli et al., 2018), with the twofold aim to widen the setting for cross-language comparisons for emoji prediction in Twitter and to experiment with novel metrics to better assess the quality of the automatic predictions.

In general, exciting and highly relevant avenues for research are still to explore with respect to emoji understanding, since emojis represent often an essential component of social media texts: ignoring or misinterpreting them may lead to misunderstandings in comprehending the intended meaning of a message (Miller et al., 2016). The ambiguity of emojis raises also interesting questions in application domains, think for instance to a human-computer interaction setting: how can we teach an artificial agent to correctly interpret and recognize emojis' use in spontaneous conversation? The main motivation behind this question is that an AI system able to predict emojis could contribute notably to better natural language understanding (Novak et al., 2015) and thus to other Natural Language Processing tasks such as generating emoji-enriched social media content, enhancing emotion/sentiment analysis systems, improving retrieval of social network material, and ultimately improving user profiling.

In the following, we describe the main elements of the shared task (Section 3), after proposing a brief summary about previous projects reflecting on the semantics of emojis in Italian (Section 2). Then, we cover the data collection, curation and release process (Section 4). In Section 5 we detail the evaluation metrics, we describe the participants results and we propose a first comparison with performances of humans solving the same task. We conclude the paper with some reflections on the outcomes of the proposed task.

2 Emojis and Italian

We can observe a growing interest on the semantics of emojis in relation with Italian. In particular, some recent interesting projects have been carried out in the last years, which address the issue in a translation framework, investigating the possibility to translate from Italian literary texts into the universal visual language of emoji (Chiusaroli, 2015; Monti et al., 2016). In particular, the *Emojitaliano project* was launched as a translation project of the Italian novel *Pinocchio* in emoji (Chiusaroli, 2017) on Twitter. An original approach based on crowdsourcing was adopted, by involving for the translation task the Twitter community named as *Scritture Brevi*.

The Twitter community #scritturebrevi The community (#scritturebrevi, @FChiusaroli, 10,151 followers in November 2018) had previously been involved in experiments of creative writing, also in emojis: with the hashtag #inemoticon, on Twitter, experiments of mixed translation - words and emojis – have been carried out, experiencing the semantic versatility of emojis, and their values in rebus writings. Translating the whole *Pinocchio* book was a more complex and engaging task, especially for its focus on developing a common code base, in terms of glossary and grammar, which is absolute new with respect to previous projects. The translation of Pinocchio started on February 2016. Everyday, for 28 weeks, sentences taken from Pinocchio were tweeted, and the followers were invited to suggest their translations to emoji; at the end of each day, the official version of the translation was validated and published. An online tool *Emojiitalianobot* has been developed in order to support the community to memorize the semantic values assigned to each emoji during the collective translation process. Since its first beginning on Twitter, the project was an instant success, becoming a viral web phenomenon thanks to the *Scritture brevi* community. Therefore, it was a natural choice to involve the same Twitter community to reflect on the semantics of emoji from a different perspective, i.e. the one we

propose in the context of the ITAmoji shared task, thus helping us to understand how humans are good at predicting emojis (see Section 5.5.2).

3 Task Description

We invited participants to submit systems designed to predict, given a tweet in Italian, its most likely associated emoji, only based on the text of the tweet. As for the experimental setting, for simplicity purposes, we considered tweets including only one emoji (eventually repeated). After removing the emoji from the tweet, we asked users to predict it. We challenged systems to predict

> Innamorato sempre di più 😍 [URL]

Figure 1: Example of tweet with an emoji at the end, considered in the emoji prediction task.

emojis among a wide and heterogeneous emoji space. In particular, we selected the tweets that included one of the twenty five emojis that occur most frequently in the Twitter data we collected (see Table 1). Therefore, the task can be seen as a multi-class classification task where systems should predict one of 25 possible emojis from the text of a given tweet. Each participant was allowed to submit up to three system runs. Participants were allowed to use additional data to train the systems such as lexicons and pre-trained word embeddings. In order to have the possibility to perform a finer grained evaluation of results, we encouraged participants to submit, for each tweet, not only the most likely emoji predicted but also the complete *rank* from the most likely to the less likely emoji to be associated to the text of the tweet.

4 Task Data

The data for this task were retrieved from Twitter by experimenting with two different approaches: (i) gathering Twitter stream on (geolocalized) Italian tweets from October 2015 to February 2018; and (ii) retrieving tweets from the followers of the most popular Italian newspaper's accounts. We randomly selected $275,000$ tweets from these collections by choosing tweets that contained one and only one emoji over 25 most frequent emojis listed in Table 1. We split our data into two sets consisting of $250,000$ *training* samples and $25,000$ *test* samples.

Emoji	% Tweet in Train and Test set
❤️	20.27
😂	19.86
😍	9.45
😉	5.35
😊	5.13
😁	4.11
😀	3.54
😘	3.33
😎	2.80
👍	2.57
🤣	2.18
🤔	2.16
💙	2.03
😜	1.94
😱	1.78
💪	1.67
😋	1.55
😅	1.52
😭	1.49
🔝	1.39
💕	1.37
☀️	1.28
💋	1.12
✨	1.07
🌹	1.06

Table 1: The distribution (percentage) for each emoji in the train and test set

5 Evaluation

In this section we present the evaluation setting for the ITAmoji shared task.

5.1 Metrics

The evaluation of the emoji prediction systems has been based on the classic *precision* and *recall* metrics over each emoji. The final ranking of the participating teams of ITAmoji 2018 relies on the **Macro F1** score computed with respect to the most likely emoji predicted, given the text of each tweet of the test set, in line with the proposal in the twin task at Semeval 2018 for English and Spanish (Barbieri et al., 2018a). In this way we intend to

encourage systems to perform well overall, which would inherently mean a better sensitivity to the use of emojis in general, rather than for instance overfitting a model to do well in the three or four most common emojis of the test data.

In general, the identification of a coherent and effective approach to compare the performance of distinct emoji prediction systems is not an easy task. We have often the clear impression that the semantics of some sets of emojis can be similar, therefore it would be interesting to have a way to compare and evaluate at a finer grained level the emoji prediction quality of two distinct systems, when they both fail in predicting the right emoji to associate to a tweet. In such cases, indeed, it can be important to distinguish between the system that identifies the right prediction among the most likely emojis to be associated to that tweet and the one that characterizes the right prediction as an emoji that is unlikely to be associated to that tweet. In order to catch this aspect, we gave ITAmoji participants the possibility to submit as emoji predictions, the *ordered ranking* of the 25 emojis considered in ITAmoji. Systems providing the ranked list of emoji predictions were also compared by considering the following additional *emoji-rank-based metrics*: **Accuracy@5/10/15/20** and **Coverage Error**. All the submissions we received provided the ranked list of 25 emojis as predictions: as a consequence it was possible to compute the emoji-rank-based metrics considered for all of them.

A detailed description of all the evaluation metrics we considered to compare the quality of emoji prediction approaches is given below. The following three **standard metrics** are computed by considering only the emoji predicted as the most likely one to be associated to the text of a tweet:

- **Macro F1**: compute the F1 score for each label (emoji), and find their un-weighted mean (exploited to determine the final ranking of the participating teams);
- **Micro F1**: compute the F1 score globally by counting the total true positives, false negatives and false positives across all label (emojis);
- **Weighted F1**: compute the F1 score for each label (emoji), and find their average, weighted by support (the number of true instances for each label);

Regarding the **emoji-rank-based metrics**, we considered:

- **Coverage error**: compute how far we need to go through the ranked scores of labels (emojis) to cover all true labels;
- **Accuracy@n**: is the accuracy value computed by considering as right predictions the ones in which the right label (emoji) is among the top N most likely ones.

5.2 Baseline

In order to compare the performance of the ITAmoji participating systems with baseline approaches, we considered three different baselines:
- Majority baseline: for each text of a tweet we predict the ordered list of 25 most-likely emojis sorted by their frequency in the training set, that is, we always predict as first choice the red heart, and as last choice the rose emoji.
- Weighted random baseline: for each text of a tweet we predict the ordered list of the 25 most-likely emojis where the first prediction is randomly selected taking in consideration the label-frequency in the training set (in order to keep the same labels distribution) and the rest of the predictions (from the second to the last one) are generated by considering the rest of emojis sorted by label-frequency.
- FastText baseline: for each text of a tweet we predict the ordered list of the 25 most-likely emojis by relying on fasttext with basic parameters[1] and pretrained embeddings with 300 dimensions (Barbieri et al., 2016a).

5.3 Participating Systems and Results

We received 12 submissions in total from 5 different teams. The main approaches and features of participating teams are described below.
FBK_FLEXED_BICEPS (Andrei et al., 2018) This system exploit recurrent neural network architecture Bidirectional Long Short Term Memory (Bi-LSTM), together with user based features to deal with this task. They concatenate the output of Bi-LSTM network that take word sequence as input with the user history distribution in using emoji. Finally, the softmax activation is used to get the probability distribution of the 25 emoji labels.

[1] https://fasttext.cc/

GW2017 (Mauro and Xileny, 2018) This system based on ensemble of two models, Bi-LSTM and LightGBM[2]. The first model uses two different word2vec models based on the time creation, while the second model exploits several surfaces feature extracted from tweet text (e.g., number of words, number of characters).

CIML-UNIPI (Daniele et al., 2018) This system is based on ensemble composed of 13 models (12 basen on TreeESNs and one on LSTM over characters. Models based on TreeESN are built by varying the number of reservoir units, activation function, readout and parser.

sentim (Jacob, 2018) This system relies on a convolutional neural network (CNN) architecture which uses character embedding as input. 9 layers of residual dilated convolutions with skip connections are applied, followed by a ReLU activation to increase nonlinearity.

UNIBA (Lucia and Daniela, 2018) This system is built by using ensemble classifier based on WEKA[3] and scikit-learn[4]. Several features are exploited by using micro-blogging based feature, sentiment based feature, and semantic based feature.

Table 2 shows the official results of ITAmoji 2018 task, ordered by decreasing Macro F1. The best performing system was proposed by the *FBK_FLEXED_BICEPS* team, which achieves 0.365312 in Macro F1. Overall, we can see that systems which exploit neural network architecture obtained good performances in this task, especially when relying on Bi-LSTM model. Table 3 shows the performance of ITAmoji systems with respect to emoji-rank-based metrics.

5.4 Analysis

From Table 2 we can notice that the ranking order of the 5 system runs that obtained the best Macro F1 is substantially preserved when we consider Micro F1 or Weighted F1. Anyway, with respect to Macro F1, when we consider Micro F1 the differences among the scores obtained by the top-performing systems tend to be substantially smaller: for instance the Macro F1 of the best system is greater by a factor of 1.64 with respect to the fifth system, while the Micro F1 of the best system is greater by a factor of 1.18 with respect to the fifth system (ranked by Micro F1). This fact can be motivated by the trend, when we consider Micro F1, to favour systems that tend to overfit their prediction model to do well in the most common emojis of the test data with respect to systems with good performances over all emojis: this fact confirms our choice to select Macro F1 as the official metric to rank ITAmoji 2018 participating systems.

From Table 3 we can see how the order to the top-5 best performing systems in terms of Macro F1 is substantially preserved when we consider the emoji-rank-based metrics Coverage Error and Accuracy@5 (except for the switch between the fourth and fifth best performing approach).

If we consider the performance of our three baseline systems (described in Section 5.2) we can notice from Table 2 that, as expected, FastText is the best performing baseline approach: a FastText embedding based prediction system would have ranked as eight by Macro F1 in ITAmoji 2018.

Table 6 shows the highest F1 score for each emoji / label across all ITAmoji 2018 team submissions. We can notice that even if specific emojis like 😋, 💋, 🌹, or ☀ are characterized by a small percentage of training samples (about 1%), prediction systems manage to obtain high Macro F1 scores. In contrast, when we consider emojis like 🤔 or 👍, even if there are more training samples available with respect to the previous set of emojis (more than 2%), we observe that the prediction systems do not manage to get high Macro F1 scores. This fact can be explained by the variability of the context of use that characterizes the latter set of emojis that makes it difficult for system to learn to predict.

To conclude our analysis, we have to notice that the three runs that obtained the highest Macro F1 scores, to predict the emojis exploited, besides the text of a tweet, the way the author of that tweet used emojis in previous tweets. This fact highlights that the choice of an emoji strongly depends on the preferences and writing style of each individual, both representing relevant inputs to model in order to improve emoji prediction quality.

5.5 Emoji prediction by humans

In this section we present a preliminary discussion of the results of two experiments designed in order to evaluate how humans perform when they are requested to identify the most likely emoji(s) to associate to the text of an Italian tweet. The

[2] https://github.com/Microsoft/LightGBM
[3] https://www.cs.waikato.ac.nz/ml/weka/
[4] http://scikit-learn.org/stable/

Rank	Team	Run Name	Macro F1	Micro F1	Weighted F1
1	FBK_FLEXED_BICEPS	base_ud_1f	36.53	47.67	46.98
2	FBK_FLEXED_BICEPS	base_ud_10f	35.63	47.62	46.58
3	FBK_FLEXED_BICEPS	base_tr_10f	29.21	42.35	39.57
4	GW2017	gw2017_p	23.29	40.09	37.81
5	GW2017	gw2017_e	22.21	42.19	36.90
6	CIML-UNIPI	run1	19.24	29.12	31.48
7	CIML-UNIPI	run2	18.80	37.63	34.101
-	**FastText baseline**		11.96	28.72	27.02
8	sentim	Sentim_Test_Run_3	10.62	29.43	23.24
9	sentim	Sentim_Test_Run_2	10.23	31.27	23.11
-	**Weighted random baseline**		3.94	10.36	10.36
10	GW2017	gw2017_pe	3.75	11.95	10.97
11	UNIBA	itamoji_uniba_run1	3.19	27.38	15.61
12	sentim	Sentim_Test_Run_1	1.95	6.48	3.99
-	**Majority baseline**		1.35	20.28	6.84

Table 2: Official Results of ITAmoji Shared Task: evaluation metrics computed by considering only the emoji predicted as the most likely one to be associated to the text of a tweet. Teams runs are ranked by Macro F1. The table shows also the performance of the three baselines considered in ITAmoji, ranked with respect to their Macro F1.

Rank	Team	Run Name	Coverage Error	Accuracy@5 / 10 / 15 / 20
1	FBK_FLEXED_BICEPS	base_ud_1f	3.47	81.67 / 92.14 / 96.86 / 99.10
2	FBK_FLEXED_BICEPS	base_ud_10f	3.49	81.53 / 91.94 / 96.82 / 99.17
3	FBK_FLEXED_BICEPS	base_tr_10f	4.35	74.54 / 87.50 / 94.34 / 98.00
4	GW2017	gw2017_p	5.66	67.18 / 81.49 / 89.42 / 92.99
5	GW2017	gw2017_e	4.60	71.30 / 85.90 / 94.30 / 98.25
6	CIML-UNIPI	run1	5.43	64.60 / 83.02 / 93.00 / 98.01
7	CIML-UNIPI	run2	5.11	68.46 / 83.86 / 92.38 / 97.28
-	**FastText baseline**		7.23	59.07 / 74.22 / 82.58 / 88.89
8	sentim	Sentim_Test_Run_3	6.41	58.53 / 76.93 / 88.52 / 95.74
9	sentim	Sentim_Test_Run_2	6.33	57.60 / 77.17 / 89.70 / 96.41
-	**Weighted random baseline**		6.92	59.06 / 76.11 / 86.42 / 94.10
10	GW2017	gw2017_pe	13.49	27.93 / 43.04 / 56.00 / 66.27
11	UNIBA	itamoji_uniba_run1	6.70	58.78 / 75.97 / 86.36 / 93.53
12	sentim	Sentim_Test_Run_1	12.45	29.20 / 48.78 / 64.38 / 74.04
-	**Majority baseline**		6.63	60.07 / 76.43 / 86.51 / 94.12

Table 3: Official Results of ITAmoji Shared Task: emoji-rank-based metrics (Coverage error and Accuracy@n). Teams runs ranked by Macro F1. The table shows also the performance of the three baselines considered in ITAmoji, ranked with respect to their Macro F1.

final purpose here is to explore if humans are better than automated systems in the emoji prediction task from text, or viceversa. In an attempt to consider an uniform set of emojis in our experimental settings, in both human emoji prediction experiments described in the rest of this section we decided to focus only on the 15 emojis shown in Table 4. This group of emojis includes all the yellow-face emojis considered in the ITAmoji task (Table 1).

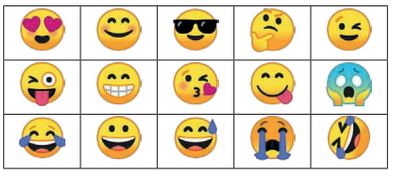


Table 4: The set of 15 face emoji considered in the human annotation experiments.

5.5.1 Figure 8 human annotation

We selected 1,005 tweets with one face-emojis from the ITAmoji test set and set up a collaborative annotation task in Figure Eight (F8)[5] by asking annotators to chose the first, second and third most likely face emoji they would associate to the text of each tweet [6]. The set of 1,005 tweets to annotate was perfectly balanced across the 15 face emojis considered. A total of 64 annotators from the F8

[5] https://www.figure-eight.com/

[6] Instructions provided to annotators (in Italian) here: http://bit.ly/itaMoji

platform provided 6,150 evaluations by spotting the 3 most likely face emojis to associate to the text of a tweet.

The Macro F1 of F8 annotators is 24.74. On the same set of 1,005 tweets, the emoji prediction performance of human annotators was better than 9 out of 12 systems submitted to ITAmoji. However, the the best performing system submitted to ITAmoji obtained a Macro F1 of 40.48 on those tweets, suggesting that computational models can perform better than humans in this task.

5.5.2 Twitter human annotation

Thanks to the support and collaboration of the #scritturebrevi Twitter community, we replicated the human annotation experiment carried out in F8 in a "crowdcourcing in the wild" setting. From the end of July to the beginning of September 2018, we posted 485 tweets on the Scritture Brevi Twitter account (*@FChiusaroli*), most of them selected from the same portion of the ITAmoji test set considered in our F8 experiment (see Section 5.5.1). Members of the Scritture Brevi Twitter community were called to participate to a sort of Twitter crowdsourcing game with slogan **#ITAmoji che passione** and hashtag #ITAmoji. Every day a set of tweets without emoji was posted on the Scritture Brevi Twitter account, and *ITAmojiers* had to post as a *reply* the most likely face emoj they would associate to the text of the posted tweet[7]. The game became viral. We managed to involve more than one hundred users with an average number of valid predictions/replies per tweet equal to 5.4. When the **#ITAmoji che passione** game ended, we were able to identify for each tweet posted on #scritturebrevi (485 tweets in total) the most-likely face-emoji that the Twitter community would associate. In general, the emoji prediction performance of people from Scritture Brevi Twitter community was better than 8 out of 12 systems submitted to ITAmoji (always on the same set of 485 tweets annotated by that community).

5.5.3 Comparing human and automated emoji predictions

In the two experiments just described, we asked humans to identify the face emoji(s) they would associate to the text of a tweet by exploiting different approaches to collect data: a controlled collaborative annotation environment in the case of F8 (Section 5.5.1) and a "crowdcourcing in the wild" setting in the case of the Scritture Brevi Twitter community (Section 5.5.2). In Table 5 we compare the emoji prediction performance of human annotators (from both F8 and Scritture Brevi Twitter community) with the performance of the emoji prediction systems submitted to ITAmoji. To perform this comparison we consider the set of 428 tweets of the ITAmoji test set annotated by F8 and the Scritture Brevi Twitter community.

We can notice that human predictions, both from F8 and Scritture Brevi, outperforms most of the automated systems. Moreover, F8 predictions obtain a Macro F1 (24.46) higher than Scritture Brevi Twitter community (22.94). This trend may be related to the fact that F8, in contrast to the #scritturebrevi Twitter community, represents a controlled annotation environment.

6 Conclusion

Considered the widespread diffusion of emojis as visual devices useful to provide an additional layer of meaning to social media messages, on one hand, and the unquestionable role of Twitter as one of the most important social media platforms, on the other, we proposed this year at Evalita 2018 ITAmoji, the Italian Emoji Prediction task.
Results of automated systems are in line with ones obtained in the twin shared task proposed for English and Spanish at Semeval 2018 (Barbieri et al., 2018a). The introduction of new experimental emoji-rank based metrics in ITAmoji allowed us to perform a finer-grained evaluation of the systems' emoji prediction quality. Moreover, comparing performances of humans and systems in the emoji prediction task confirms also in an Italian setting the outcomes of a similar experiment proposed for English (Barbieri et al., 2017), suggesting that computational models are able to better capture the underlying semantics of emojis.

[7]The announce of the "#ITAmoji che passione" game was published on the Scritture Brevi's blog and linked to every posted tweet: `https://www.scritturebrevi.it/2018/07/16/itamoji-che-passione/`

References

Catalin Coman Andrei, Nechaev Yaroslav, and Zara Giacomo. 2018. Predicting emoji exploiting multimodal data: Fbk participation in itamoji task. In Tommaso Caselli, Nicole Novielli, Viviana Patti, and Paolo Rosso, editors, *Proceedings of 6th Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2018)*, Turin, Italy. CEUR.org.

Team	Run Name	Macro F1	Micro F1	Weighted F1
FBK_FLEXED_BICEPS	base_ud_1f	35.70	34.81	35.94
FBK_FLEXED_BICEPS	base_tr_10f	35.03	34.81	35.36
FBK_FLEXED_BICEPS	base_ud_10f	34.73	34.11	34.83
Figure Eight predictions		24.46	26.40	24.57
CIML-UNIPI	run1	24.03	25.00	23.65
Scritture Brevi predictions		22.94	24.06	22.99
GW2017	gw2017_p	20.40	23.13	19.97
GW2017	gw2017_e	20.33	22.66	19.83
CIML-UNIPI	run2	19.45	21.26	18.80
sentim	Sentim_Test_Run_2	12.17	15.19	11.59
sentim	Sentim_Test_Run_3	11.07	14.49	10.82
GW2017	gw2017_pe	5.01	7.48	5.02
UNIBA	itamoji_uniba_run1	2.95	7.47	2.84
sentim	Sentim_Test_Run_1	2.74	4.90	2.83

Table 5: Performance of human (Scritture Brevi and Figure 8) and automated emoji prediction approaches, compared by considering the set of 428 tweets with face-emoji that are part of the ITAmoji test set and have been annotated by both Figure 8 platform and #scritturebrevi community. Emoji prediction approaches are ranked by decreasing Macro F1.

Emoji	Label	Macro F1	Num. Samples	% Samples
❤️	red heart	75.74	5069	20.28
😂	face with tears of joy	57.08	4966	19.86
💋	kiss mark	51.71	279	1.12
😋	face savoring food	48.34	387	1.55
🌹	rose	46.83	265	1.06
☀️	sun	44.69	319	1.28
😍	smiling face with heart eyes	42.93	2363	9.45
😘	face blowing a kiss	41.61	834	3.34
💙	blue heart	39.26	506	2.02
😊	smiling face with smiling eyes	38.92	1282	5.13
😀	grinning face	37.74	885	3.54
😉	winking face	34.98	1338	5.35
😁	beaming face with smiling eyes	34.47	1028	4.11
✨	sparkles	32.31	266	1.06
🤣	rolling on the floor laughing	31.79	546	2.18
👍	thumbs up	31.55	642	2.57
😎	smiling face with sunglasses	30.89	700	2.80
💪	flexed biceps	30.75	417	1.67
🤔	thinking face	29.06	541	2.16
💕	two hearts	27.48	341	1.36
😭	loudly crying face	25.62	373	1.49
🔝	top arrow	24.03	347	1.39
😅	grinning face with sweat	23.94	379	1.52
😜	winking face with tongue	23.66	483	1.93
😱	face screaming in fear	22.56	444	1.78

Table 6: Best F1 score for each emoji / label across all ITAmoji 2018 teams. The fourth and fifth columns respectively show, for each emoji, the number and percentage of test samples present in the test dataset.

Sho Aoki and Osamu Uchida. 2011. A method for automatically generating the emotional vectors of emoticons using weblog articles. In *Proc. 10th WSEAS Int. Conf. on Applied Computer and Applied Computational Science, Stevens Point, Wisconsin, USA*, pages 132–136.

Francesco Barbieri, German Kruszewski, Francesco Ronzano, and Horacio Saggion. 2016a. How cosmopolitan are emojis?: Exploring emojis usage and meaning over different languages with distributional semantics. In *Proceedings of the 2016 ACM on Multimedia Conference*, pages 531–535. ACM.

Francesco Barbieri, Francesco Ronzano, and Horacio Saggion. 2016b. What does this emoji mean? a vector space skip-gram model for Twitter emojis. In *Proc. of LREC 2016*.

Francesco Barbieri, Miguel Ballesteros, and Horacio Saggion. 2017. Are emojis predictable? In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 105–111, Valencia, Spain, April. ACL.

Francesco Barbieri, Jose Camacho-Collados, Francesco Ronzano, Luis Espinosa Anke, Miguel Ballesteros, Valerio Basile, Viviana Patti, and Horacio Saggion. 2018a. Semeval 2018 task 2: Multilingual emoji prediction. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 24–33. ACL.

Francesco Barbieri, Luis Marujo, William Brendel, Pradeep Karuturim, and Horacio Saggion. 2018b. Exploring Emoji Usage and Prediction Through a Temporal Variation Lens. In *1st International Workshop on Emoji Understanding and Applications in Social Media (at ICWSM 2018)*.

Tommaso Caselli, Nicole Novielli, Viviana Patti, and Paolo Rosso. 2018. Evalita 2018: Overview of the 6th evaluation campaign of natural language processing and speech tools for italian. In Tommaso Caselli, Nicole Novielli, Viviana Patti, and Paolo Rosso, editors, *Proceedings of Sixth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2018)*, Turin, Italy. CEUR.org.

Francesca Chiusaroli. 2015. La scrittura in emoji tra dizionario e traduzione. In *Proceedings of 2nd Italian Conference on Computational Linguistics (CLiC-it 2015), Trento, Italy, December 3-4, 2015.* Aacademia University Press.

F. Chiusaroli. 2017. *Pinocchio in emojitaliano*. Apice Libri.

Di Sarli Daniele, Gallicchio Claudio, and Micheli Alessio. 2018. Itamoji 2018: Emoji prediction via tree echo state networks. In Tommaso Caselli, Nicole Novielli, Viviana Patti, and Paolo Rosso, editors, *Proceedings of 6th Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2018)*, Turin, Italy. CEUR.org.

Giulia Donato and Patrizia Paggio. 2017. Investigating redundancy in emoji use: Study on a Twitter based corpus. In *Proceedings of the 8th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 118–126.

Giulia Donato and Patrizia Paggio. 2018. Classifying the Informative Behaviour of Emoji in Microblogs. In *Proc. of the 11th International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan, May 7-12, 2018. ELRA.

Ben Eisner, Tim Rocktäschel, Isabelle Augenstein, Matko Bošnjak, and Sebastian Riedel. 2016. emoji2vec: Learning emoji representations from their description. *arXiv preprint arXiv:1609.08359*.

Anderson Jacob. 2018. Fully convolutional networks for text classification. In Tommaso Caselli, Nicole Novielli, Viviana Patti, and Paolo Rosso, editors, *Proceedings of 6th Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2018)*, Turin, Italy. CEUR.org.

Nikola Ljubešić and Darja Fišer. 2016. A global analysis of emoji usage. In *Proceedings of the 10th Web as Corpus Workshop*, pages 82–89. Association for Computational Linguistics.

Siciliani Lucia and Girardi Daniela. 2018. The uniba system at the evalita 2018 italian emoji prediction task. In Tommaso Caselli, Nicole Novielli, Viviana Patti, and Paolo Rosso, editors, *Proceedings of 6th Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2018)*, Turin, Italy. CEUR.org.

Bennici Mauro and Seijas Portocarrero Xileny. 2018. The validity of word vectors over the time for the evalita 2018 emoji prediction task (itamoji). In Tommaso Caselli, Nicole Novielli, Viviana Patti, and Paolo Rosso, editors, *Proceedings of 6th Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2018)*, Turin, Italy. CEUR.org.

Hannah Miller, Jacob Thebault-Spieker, Shuo Chang, Isaac Johnson, Loren Terveen, and Brent Hecht. 2016. “Blissfully happy" or “ready to fight": Varying interpretations of emoji. *Proc. of ICWSM'16*.

Johanna Monti, Federico Sangati, Francesca Chiusaroli, Martin Benjamin, and Sina Mansour. 2016. Emojitalianobot and emojiworldbot - new online tools and digital environments for translation into emoji. In *Proc. CLiC-it 2016, Napoli, Italy, December 5-7, 2016.*, volume 1749 of *CEUR Workshop Proceedings*.

Petra Kralj Novak, Jasmina Smailović, Borut Sluban, and Igor Mozetič. 2015. Sentiment of emojis. *PloS one*, 10(12):e0144296.

Overview of the EVALITA 2018 Task on Irony Detection in Italian Tweets (IronITA)

Alessandra Teresa Cignarella
Simona Frenda
Dipartimento di Informatica
Università degli Studi di Torino, Italy
PRHLT Research Center
Universitat Politècnica de València, Spain
{cigna,frenda}@di.unito.it

Valerio Basile, Cristina Bosco
Viviana Patti
Dipartimento di Informatica
Università degli Studi di Torino, Italy
{basile,bosco,patti}@di.unito.it

Paolo Rosso
PRHLT Research Center
Universitat Politècnica de València, Spain
prosso@dsic.upv.es

Abstract

English. IronITA is a new shared task in the EVALITA 2018 evaluation campaign, focused on the automatic classification of irony in Italian texts from Twitter. It includes two tasks: 1) irony detection and 2) detection of different types of irony, with a special focus on sarcasm identification. We received 17 submissions for the first task and 7 submissions for the second task from 7 teams.

Italiano. *IronITA è un nuovo esercizio di valutazione della campagna di valutazione EVALITA 2018, specificamente dedicato alla classificazione automatica dell'ironia presente in testi estratti da Twitter. Comprende due task: 1) riconoscimento dell'ironia e 2) riconoscimento di diversi tipi di ironia, con particolare attenzione all'identificazione del sarcasmo. Abbiamo ricevuto 17 sottomissioni per il primo task e 7 per il secondo, da parte di 7 gruppi partecipanti.*

1 Introduction

Irony is a figurative language device that conveys the opposite of literal meaning, profiling intentionally a secondary or extended meaning. Users on the web usually tend to use irony like a creative device to express their thoughts in short-texts like tweets, reviews, posts or commentaries. But irony, as well as other figurative language devices, for example metaphors, is very difficult to deal with automatically. For its traits of recalling another meaning or obfuscating the real communicative intention, it hinders correct sentiment analysis of texts and, therefore, correct opinion mining. Indeed, the presence of ironic devices in a text can work as an unexpected "polarity reverser" (one says something "good" to mean something "bad"), thus undermining systems' accuracy.

Considering the majority of state-of-the-art studies in computational linguistics, *irony* is often used as an umbrella-term which includes satire, sarcasm and parody due to fuzzy boundaries among them (Marchetti et al., 2007). However, some linguistic studies focused on *sarcasm*, a particular type of verbal irony defined in Gibbs (2000) as "a sharp or cutting ironic expression with the intent to convey scorn or insult". Other scholars concentrated on cognitive aspects related on how such figurative expressions are processed in the brain, focusing on key aspects influencing processing (see for instance the "defaultness" hypothesis presented in Giora et al. (2018)).

The importance to detect irony and sarcasm is also very relevant for reaching better predictions in Sentiment Analysis, for instance, what are the real opinion and orientation of users about a specific subject (product, service, topic, issue, person, organization, or event).

IronITA is organized in continuity with previous shared tasks of the past years within the context of the EVALITA evaluation campaign (see for instance the irony detection subtask proposed at SENTIPOLC in the 2014 and 2016 editions (Basile et al., 2014; Barbieri et al., 2016)). It is also inspired by the recent experience within the SemEval2018-Task3 *Irony detection in English tweets* (Van Hee et al., 2018). The shared task we propose for Italian is specifically dedicated to

irony detection taking into account both the classical binary classification task (irony vs not irony), and a related subtask, which gives to participants the possibility to reason on different types of irony. Differently from SemEval2018-Task3, we indeed ask the participants to distinguish sarcasm as a specific type of irony. This is motivated by the growing interest for detecting sarcasm, which is characterized by sharp tones and aggressive intention (Gibbs, 2000; Joshi et al., 2017; Sulis et al., 2016) often present in interesting domains such as politics and hate speech (Sanguinetti et al., 2018).

2 Task Description

The task consists in automatically annotating messages from Twitter for irony and sarcasm. It is organized in a main task (Task A) centered on irony, and a second task (Task B) centered on sarcasm, whose results will be separately evaluated. Participation was allowed to both the tasks (Task A and Task B) or to Task A only.

Task A: Irony Detection. Task A consists in a two-class (or binary) classification where systems have to predict whether a tweet is ironic or not.

Task B: Different types of irony with special focus on sarcasm identification. Sarcasm has been recognized in Bowes and Katz (2011) with a specific target to attack (Attardo, 2007; Dynel, 2014), more offensive and delivered with a cutting tone (rarely ambiguous). According to Lee and Katz (1998) hearers perceive aggressiveness as the feature that distinguishes sarcasm. Provided a definition of sarcasm as a specific type of irony, Task B consists in a multi-class classification where systems have to predict one out of the three following labels: **i) sarcasm**, **ii) irony not categorized as sarcasm** (i.e. other kinds of verbal irony or descriptions of situational irony which do not show the characteristics of sarcasm), and **iii) not-irony**.

The proposed tasks encourage the investigation of this linguistic devices. Moreover, providing a dataset from social media (Twitter), we focus on texts especially hard to be dealt with, because of their shortness and because they will be analyzed out of the context where they were generated.

The participants are allowed to submit either "constrained" or "unconstrained" runs (or both, within the submission limits). The constrained runs have to be produced by systems whose only training data is the dataset provided by the task organizers. On the other hand, the participant teams are encouraged to train their systems on additional annotated data and submit the resulting unconstrained runs.

We implemented two straightforward baseline systems for the task. *baseline-mfc* (Most Frequent Class) assigns to each instance the majority class of the respective task, namely `not-ironic` for task A and `not-sarcastic` for task B. *baseline-random* assigns uniformly random values to the instances. Note that for task A, a class is assigned randomly to every instance, while for task B the classes are assigned randomly only to eligible tweets who are marked `ironic`.

3 Training and Test Data

3.1 Composition of the datasets

The data released for the shared task come from different source datasets, namely: Hate Speech Corpus (HSC) (Sanguinetti et al., 2018) and the TWITTIRÒ corpus (Cignarella et al., 2018), composed of tweets from LaBuonaScuola corpus (TW-BS) (Stranisci et al., 2016), Sentipolc corpus (TW-SENTIPOLC), Spinoza corpus (TW-SPINO) (Barbieri et al., 2016).

In the test data we have the same sources, and in addition some tweets from the TWITA collection, that were annotated by the organizers of the SENTIPOLC 2016 shared task, but were not exploited during the 2016 campaign (Barbieri et al., 2016).

3.2 Annotation of the datasets

The annotation process involved four Italian native speakers and focused only on the finer-grained annotation of sarcasm in the ironic tweets, since the presence of irony was already annotated in the source datasets. It began by splitting in two halves the dataset and assigning the annotation task for each portion to a different couple of annotators. In the following step, the final inter-annotator agreement (IAA) has been calculated on all the dataset. Then, in order to achieve an agreement on a larger portion of data, the effort of the annotators has been focused only on the detected cases of disagreement. In particular, the couple previously involved in the annotation of the first half of the corpus produced a new annotation for the tweets in disagreement of the second portion of the dataset, while the couple involved in the annotation of the second half of the corpus did the same on the first

	TRAINING SET				TEST SET				
	IRONIC	NOT-IRO	SARC	NOT-SARC	IRONIC	NOT-IRO	SARC	NOT-SARC	TOTAL
TW-BS	467	646	173	294	111	161	51	60	
TW-SPINO	342	0	126	216	73	0	32	41	2,886
TW-SENTIPOLC	461	625	143	318	0	0	0	0	
HSC	753	683	471	282	185	119	106	79	1,740
TWITA	0	0	0	0	67	156	28	39	223
TOTAL	**3,977**				**872**				**4,849**

Table 1: Distribution of tweets according to the topic

portion of the dataset. After that, the cases where the disagreement persists have been discarded as too ambiguous to be classified (131 tweets).

The final IAA calculated with Fleiss' kappa is $\kappa = 0.56$ for the tweets belonging to the TWITTIRÒ corpus and $\kappa = 0.52$ for the data from the HSC corpus and it is considered *moderate*[1] and satisfying for the purpose of the shared task.

In this process the annotators relied on a specific definition of "sarcasm", and followed detailed guidelines[2]. In particular we defined **sarcasm** as *a kind of sharp, explicit and sometimes aggressive irony, aimed at hitting a specific target to hurt or criticize without excluding the possibility of having fun* (Du Marsais et al., 1981; Gibbs, 2000). The factors we have taken into account for the annotation are, the presence of:

1. a clear **target**,
2. an obvious **intention** to hurt or criticize,
3. **negativity** (weak or strong).

We have also tried to differentiate our concept of "sarcasm" from that of "satire", often present in tweets. For us, satire aims to ridicule the target as well as criticize it. Differently from sarcasm, satire is solely focused on a more negative type of criticism and moved by a personal and angry emotional charge.

A single training set has been provided for both tasks A and B, which includes 3,977 tweets. Following, a single test set has been distributed for both tasks A and B, which includes 872 tweets, hence creating an $82\% - 18\%$ balance between training and test data. Table 1 shows the distribution of ironic and sarcastic tweets among the different source/topic datasets cited in Section 3.1.

Additionally the IronITA datasets overlap with the data released for *HaSpeeDe*, the task of Hate Speech Detection (Bosco et al., 2018). In the training set we count 781 overlapping tweets, while in the test set we count an overlap of just 96 tweets.

3.3 Data Release

The data were released in the following format[3]:

```
idtwitter text irony sarcasm topic
```

where `idtwitter` is the Twitter ID of the message, `text` is the content of the message, `irony` is 1 or 0 (respectively for ironic and not ironic tweets), `sarcasm` is 1 or 0 (respectively for sarcastic and not sarcastic tweets), and `topic` refers to the source corpus from where the tweet has been extracted.

The training set includes for each tweet the annotation for the `irony` and `sarcasm` fields, according to the format explained above. Instead, the test set only containes values for the `idtwitter`, `text` and `topic` fields.

4 Evaluation Measures

Task A: Irony detection. Systems have been evaluated against the gold standard test set on their assignment of a 0 or 1 value to the `irony` field. We measured the precision, recall and F1-score of the prediction for both the `ironic` and `not-ironic` classes:

$$precision_{class} = \frac{\#correct\_class}{\#assigned\_class}$$

$$recall_{class} = \frac{\#correct\_class}{\#total\_class}$$

$$F1_{class} = 2\frac{precision_{class} recall_{class}}{precision_{class} + recall_{class}}$$

The overall F1-score is the average of the F1-scores for the `ironic` and `not-ironic` classes (i.e. macro F1-score).

[1] According to the parameters proposed by Fleiss (1971).

[2] For more details on this regard, please refer to the guidelines: `https://github.com/AleT-Cig/IronITA-2018/blob/master/Definition%20of%20Sarcasm.pdf`

[3] Link to the datasets: `http://www.di.unito.it/~tutreeb/ironita-evalita18/data.html`

topic	irony	sarcasm	**text**
TWITTIRÒ	0	0	*@SteGiannini @sdisponibile Semmai l'anno DELLA buona scuola. De la, in italiano, non esiste*
TWITTIRÒ	1	1	*#labuonascuola Fornitura illimitata di rotoli di carta igienica e poi, piano piano, tutti gli altri aspetti meno importanti.*
HSC	1	0	*Di fronte a queste forme di terrorismo siamo tutti sulla stessa barca. A parte Briatore. Briatore ha la sua.*
HSC	1	1	*Anche oggi sono in arrivo 2000migranti dalla Libia avanti in italia ce posto per tutti vero @lauraboldrini ? Li puoi accogliere a casa tua*

Table 2: Examples for each combinations

Task B: Different types of irony. Systems have been evaluated against the gold standard test set on their assignment of a 0 or 1 value to the `sarcasm` field, assuming that the `irony` field is also provided as part of the results.

We have measured the precision, recall and F1-score for each of the three classes:

- not-ironic
  `irony` = 0, `sarcasm` = 0
- ironic-not-sarcastic
  `irony` = 1, `sarcasm` = 0
- sarcastic
  `irony` = 1, `sarcasm` = 1

The evaluation metric is the macro F1-score computed over the three classes. Note that for the purpose of the evaluation of task B, the following combination is always considered wrong:

- `irony` = 0, `sarcasm` = 1

Our scheme imposes that a tweet can be annotated as sarcastic only if it is also annotated as ironic, which correspond to interpreting sarcasm as a specific type of irony, as reported in Table 2.

5 Participants and Results

A total of 7 teams, both from academia and industry sector participated to at least one of the two tasks of IronITA. Table 3 provides an overview of the teams, their affiliation, and the tasks they took part in.

Four teams participated to both tasks A and B. Teams were allowed to submit up to four runs (2 constrained and 2 unconstrained) in case they implemented different systems. Furthermore, each team had to submit at least a constrained run. Participants have been invited to submit multiple runs to experiment with different models and architectures. However, they have been discouraged from submitting slight variations of the same model. Overall we have 17 runs for Task A and 7 runs for Task B.

5.1 Task A: Irony Detection

Table 4 shows the results for the irony detection task, which attracted 17 total submissions from 7 different teams. The best scores are achieved by the ItaliaNLP team (Cimino et al., 2018) that, with a constrained run, obtained the best score for both the `ironic` and `not-ironic` class, thus obtaining the highest averaged F1-score of 0.731.

Among the unconstrained systems, the best F1-score for the `not-ironic` class is achieved by the X2Check team (Di Rosa and Durante, 2018) with F = 0.708, and the best F1-score for the `ironic` class is obtained by the UNITOR team (Santilli et al., 2018) with F = 0.733.

All participating systems show an improvement over the baselines, with the exception of the only unsupervised system (venses-itgetarun, see details in Section 6).

team name	**id**	**F1-score**		
		not-iro	**iro**	**macro**
ItaliaNLP	1	**0.707**	**0.754**	**0.731**
ItaliaNLP	2	0.693	0.733	0.713
UNIBA	1	0.689	0.730	0.710
UNIBA	2	0.689	0.730	0.710
X2Check	1	**0.708**	0.700	**0.704**
UNITOR	1	0.662	0.739	0.700
UNITOR	2	0.668	**0.733**	0.700
X2Check	2	0.700	0.689	0.695
Aspie96	1	0.668	0.722	0.695
X2Check	2	0.679	0.708	0.693
X2Check	1	0.674	0.693	0.683
UO_IRO	2	0.603	0.700	0.651
UO_IRO	1	0.626	0.665	0.646
UO_IRO	2	0.579	0.678	0.629
UO_IRO	1	0.652	0.577	0.614
baseline-random		0.503	0.506	0.505
venses-itgetarun	1	0.651	0.289	0.470
venses-itgetarun	2	0.645	0.195	0.420
baseline-mfc		0.668	0.000	0.334

Table 4: Results Task A. Unconstrained runs are marked by grey background.

5.2 Task B: Different types of irony

Table 5 shows the results for the different types of irony task, which attracted 7 total submis-

team name	institution	tasks
ItaliaNLP	ItaliaNLP group ILC-CNR	A,B
UNIBA	University of Bari	A
X2Check	App2Check srl	A
UNITOR	University of Roma "Tor Vergata"	A,B
Aspie96	University of Torino	A,B
UO_IRO	CERPAMID, Santiago de Cuba / University of Informatics Sciences, Havana	A
venses-itgetarun	Ca' Foscari University of Venice	A,B

Table 3: Participants

sions from 4 different teams. The best scores are achieved by the UNITOR team that with an unconstrained run obtained the highest macro F1-score of 0.520.

Among the constrained systems, the best F1-score for the `not-ironic` class is achieved by the ItaliaNLP team with F1-score = 0.707, and the best F1-score for the `ironic` class is obtained by the Aspie96 team (Giudice, 2018) with F1-score = 0.438. The best score for the `sarcastic` class is obtained by a constrained run of the UNITOR team with F1-score = 0.459. The best performing UNITOR team is also the only team that participated to Task B with an unconstrained run.

team name	id	F1-score			
		not-iro	iro	sarc	macro
UNITOR	2	**0.668**	**0.447**	**0.446**	**0.520**
UNITOR	1	0.662	0.432	**0.459**	**0.518**
ItaliaNLP	1	**0.707**	0.432	0.409	0.516
ItaliaNLP	2	0.693	0.423	0.392	0.503
Aspie96	1	0.668	**0.438**	0.289	0.465
baseline-random		0.503	0.266	0.242	0.337
venses-itgetarun	1	0.431	0.260	0.018	0.236
baseline-mfc		0.668	0.000	0.000	0.223
venses-itgetarun	2	0.413	0.183	0.000	0.199

Table 5: Results Task B. Unconstrained runs are marked by grey background.

All participating systems show an improvement over the baselines, with the exception of the only unsupervised system (venses-itgetarun, see details in Section 6).

6 Discussion

We compare the participating systems according to the following main dimensions: classification framework (approaches, algorithms, features), text representation strategy, use of additional annotated data for training, external resources (e.g. sentiment lexica, NLP tools, etc.), and interdependency between the two tasks. This discussion is based on the information contained in the reports submitted by the participants (we received 6 reports out of 7 participating teams) and on the answers to a questionnaire sent by the organizers to the participants.

System architecture. Most submitted runs to IronITA are produced by supervised machine learning systems. In fact, all but one systems are supervised, although the nature and complexity of their architectures varies significantly. UNIBA (Basile and Semeraro, 2018) and UNITOR use Support Vector Machine (SVM) classifiers, with different parameter settings. UNITOR, in particular, employs a multiple kernel-based approach to create two SVM classifiers that work on the two tasks. X2Check uses several models based on Multinomial Naive Bayes and SVM in a voting ensemble. Three systems implemented deep learning neural networks for the classification of irony and sarcasm. Sequence-learning networks were a popular choice, in the form of Bidirectional Long Short-term Memory Networks (used by ItaliaNLP and UO_IRO (Ortega-Bueno and Medina Pagola, 2018)) and Gated Recurrent Units (Aspie96). The venses-itgetarun team proposed the only unsupervised system submitted to IronITA. The system is based on an extension of the ITGETARUN rule-based fully symbolic semantic parser (Delmonte, 2014). The performance of the venses-itgetarun system is penalized mainly by its low recall (see the detailed results on the task website).

Features. In addition to explore a broad spectrum of supervised and unsupervised architectures, the submitted systems leverage different kinds of linguistic and semantic information extracted from the tweets. Word n-grams of varying size are used by ItaliaNLP, UNIBA, and X2Check. Word embeddings were used as features by three systems, namely ItaliaNLP (built with word2vec on a concatenation of ItWaC[4] and a custom tweet corpus), UNITOR (built with

[4] `https://www.sketchengine.eu/itwac-italian-corpus/`

word2vec on a custom Twitter corpus) and UNIBA (built with Random Indexing (Sahlgren, 2005)) on a subset of TWITA (Basile et al., 2018). Affective lexicons were also employed to extract polarity-related features from the words in the tweets, by UNIBA, ItaliaNLP and UNITOR and UO_IRO (see the "Lexical Resources" section for details on the lexica). UNIBA and UO_IRO also computed sentiment variation and contrast in order to extract the ironic content from the text. Features derived from sentiment analysis are also employed by the unsupervised system venses-itgetarun. Aspie96 performs its classification based on the single characters of the tweet. Finally, a great number of other features is employed by the systems, including stylistic and structural features (UO_IRO), special tokens and emoticons (X2Check). See the details in the EVALITA proceedings (Caselli et al., 2018).

Lexical Resources. Several systems employed affective resources, mainly as a tool to compute the sentiment polarity of words and each tweet. ItaliaNLP used two affective lexica generated automatically by means of distant supervision and automatic translation. UNIBA used an automatic translation of SentiWordNet (Esuli and Sebastiani, 2006). UNITOR used the Distributed Polarity Lexicon by Castellucci et al. (2016). UO_IRO used the affective lexicon derived from the OpeNER project (Russo et al., 2016) and a polarity lexicon of emojis by Kralj Novak et al. (2015). venses-itgetarun used several lexica, including some specifically built for ITGETARUNS and a translation of SentiWordNet (Esuli and Sebastiani, 2006).

Additional training data. Three teams took the opportunity to send unconstrained runs along with constrained runs. X2Check included in the unconstrained training set a balanced version of the SENTIPOLC 2016 dataset, Italian tweets annotated with irony (Barbieri et al., 2016). UNITOR used for their unconstrained runs a dataset of 6,000 tweets obtained by distant supervision (searching for the hashtag #ironia — #irony). UO_IRO employed tweets annotated with fine-grained irony from TWITTIRÒ (Cignarella et al., 2018).

The team ItaliaNLP did not send unconstrained runs, although they used the information about polarity of Italian tweets from the SENTIPOLC 2016 dataset (Barbieri et al., 2016) and the data annotated for hate speech from the HaSpeeDe task at EVALITA 2018 (Bosco et al., 2018). We do not consider their runs unconstrained, because the phenomena annotated in the data they employed are different from irony.

Interdependency of tasks. Since the tasks A and B are inherently linked (a tweet can be sarcastic only if it is also ironic), some of the participating teams leveraged this information in their classification systems. ItaliaNLP employed a Multi-task learning approach, thus solving the two tasks simultaneously. UNITOR adopted a cascade architecture where only tweets that were classified as ironic were passed through to the sarcasm classifier. In the system by venses-itgetarun, the decision on whether to assign a tweet to `sarcasm` or `irony` is based on the contemporary presence of features common to the two tasks.

7 Concluding remarks

Differently from the previous sub-tasks on irony detection in Italian language proposed as part of the previous SENTIPOLC shared tasks, having Sentiment Analysis as reference framework, the IronITA tasks specifically focus on the irony and sarcasm identification.

Comparing the results for irony detection obtained within the SENTIPOLC sub-task (the best performing system in the 2016 edition reached $F = 0.5412$ and in 2014 $F = 0.575$) with the ones obtained in IronITA, it is worth to notice that a dedicated task on irony detection leaded to a remarkable improvement of the scores, with the highest value here being $F = 0.731$.

Surprisingly, scores for Italian are in line with those obtained at SemEval2018-Task3 on irony detection in English tweets, even if a lower amount of linguistic resources is available for Italian than for English, especially in term of affective lexica, a type of resource that is frequently exploited in this kind of task. Actually, some teams used resources provided by the Italian NLP community also in the framework of previous EVALITA's edition (e.g. additional information from annotated corpora as SENTIPOLC, HaSpeeDe and POSTWITA).

The good results obtained in this edition can be read also as a confirmation that linguistic resources for Italian language are increasing in quantity and quality, and they are helpful also for a very challenging task as irony detection.

Another interesting factor in this edition is the use of the innovative deep learning techniques, mirroring the growing interest in deep learning by the NLP community at large. Indeed, the best performing system is based on a deep learning approach revealing its usefulness also for irony detection. The high performance of deep learning methods is an indication that irony and sarcasm are phenomena involving more complex features than n-grams and lexical polarity.

The number of participants in task B was lower. Even though we wanted to encourage the investigation in the identification of sarcasm, we are aware that addressing the finer-grained task to discriminate between irony and sarcasm is still really difficult.

In hindsight, the organization of such a shared task, specifically dedicated to irony detection in Italian tweets, and also focused on diverse types of irony has been a hazard. It was intended to foster research teams in the exploitation of lexical and affective resources in Italian, developed in our NLP community and to encourage the investigation especially on data about politics and immigration.

Our proposal for this shared task arose from the intuition that a better recognition of figurative language like irony in social media data could also lead to a better resolution of other Sentiment Analysis tasks such as Hate Speech Detection (Bosco et al., 2018), Stance Detection (Mohammad et al., 2017), and Misogyny Detection (Fersini et al., 2018). IronITA wanted to be a first try-out and a first stimulus in this challenging field.

Acknowledgments

V. Basile, C. Bosco and V. Patti were partially supported by Progetto di Ateneo/CSP 2016 (*Immigrants, Hate and Prejudice in Social Media-IhatePrejudice*, S1618_L2_BOSC_01). The work of S.Frenda and P. Rosso was partially funded by the Spanish research project SomEMBED TIN2015-71147-C2-1-P (MINECO/FEDER).

References

Salvatore Attardo. 2007. Irony as relevant inappropriateness. In H. Colston and R. Gibbs, editors, *Irony in language and thought: A cognitive science reader*, pages 135–172. Lawrence Erlbaum.

Francesco Barbieri, Valerio Basile, Danilo Croce, Malvina Nissim, Nicole Novielli, and Viviana Patti. 2016. Overview of the Evalita 2016 sentiment polarity classification task. In *Proceedings of 3rd Italian Conference on Computational Linguistics (CLiC-it 2016) & 5th Evaluation Campaign of Natural Language Processing and Speech Tools for Italian*, Naples, Italy. CEUR.org.

Pierpaolo Basile and Giovanni Semeraro. 2018. UNIBA - Integrating distributional semantics features in a supervised approach for detecting irony in Italian tweets. In *Proceedings of the 6th evaluation campaign of Natural Language Processing and Speech tools for Italian (EVALITA'18)*, Turin, Italy. CEUR.org.

Valerio Basile, Andrea Bolioli, Malvina Nissim, Viviana Patti, and Paolo Rosso. 2014. Overview of the Evalita 2014 sentiment polarity classification task. In *Proceedings of the 4th evaluation campaign of Natural Language Processing and Speech tools for Italian (EVALITA'14)*, Pisa, Italy. Pisa University Press.

Valerio Basile, Mirko Lai, and Manuela Sanguinetti. 2018. Long-term Social Media Data Collection at the University of Turin. In *Proceedings of the 5th Italian Conference on Computational Linguistics (CLiC-it 2018)*, Turin, Italy. CEUR.org.

Cristina Bosco, Felice Dell'Orletta, Fabio Poletto, Manuela Sanguinetti, and Maurizio Tesconi. 2018. Overview of the Evalita 2018 Hate Speech Detection Task. In *Proceedings of the 6th evaluation campaign of Natural Language Processing and Speech tools for Italian (EVALITA'18)*, Turin, Italy. CEUR.org.

Andrea Bowes and Albert Katz. 2011. When sarcasm stings. *Discourse Processes: A Multidisciplinary Journal*, 48(4):215–236.

Tommaso Caselli, Nicole Novielli, Viviana Patti, and Paolo Rosso. 2018. EVALITA 2018: Overview of the 6th Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. In *Proceedings of 6th Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2018)*, Turin, Italy. CEUR.org.

Giuseppe Castellucci, Danilo Croce, and Roberto Basili. 2016. A language independent method for generating large scale polarity lexicons. In *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016)*, Portorož, Slovenia. ELRA.

Alessandra Teresa Cignarella, Cristina Bosco, Viviana Patti, and Mirko Lai. 2018. Application and Analysis of a Multi-layered Scheme for Irony on the Italian Twitter Corpus TWITTIRÒ. In *Proceedings of the 11th International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. ELRA.

Andrea Cimino, Lorenzo De Mattei, and Felice Dell'Orletta. 2018. Multi-task Learning in Deep Neural Networks at EVALITA 2018. In *Proceedings of the 6th evaluation campaign of Natural Language Processing and Speech tools for Italian (EVALITA'18)*, Turin, Italy. CEUR.org.

Rodolfo Delmonte. 2014. A linguistic rule-based system for pragmatic text processing. In *Proceedings of Fourth International Workshop EVALITA 2014*, Pisa. Edizioni PLUS, Pisa University Press.

Emanuele Di Rosa and Alberto Durante. 2018. Irony detection in tweets: X2Check at Ironita 2018. In *Proceedings of the 6th evaluation campaign of Natural Language Processing and Speech tools for Italian (EVALITA'18)*, Turin, Italy. CEUR.org.

César Chesneau Du Marsais, Jean Paulhan, and Claude Mouchard. 1981. *Traité des tropes*. Le Nouveau Commerce.

Marta Dynel. 2014. Linguistic approaches to (non) humorous irony. *Humor - International Journal of Humor Research*, 27(6):537–550.

Andrea Esuli and Fabrizio Sebastiani. 2006. Sentiwordnet: A publicly available lexical resource for opinion mining. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC 2006)*, Genova, Italy.

Elisabetta Fersini, Maria Anzovino, and Paolo Rosso. 2018. Overview of the Task on Automatic Misogyny Identification at IberEval. In *Proceedings of 3rd Workshop on Evaluation of Human Language Technologies for Iberian Languages (IberEval 2018)*. CEUR-WS.org.

Joseph L. Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological bulletin*.

Raymond W. Gibbs. 2000. Irony in talk among friends. *Metaphor and symbol*, 15(1-2):5–27.

Rachel Giora, Adi Cholev, Ofer Fein, and Orna Peleg. 2018. On the superiority of defaultness: Hemispheric perspectives of processing negative and affirmative sarcasm. *Metaphor and Symbol*, 33(3):163–174.

Valentino Giudice. 2018. Aspie96 at IronITA (EVALITA 2018): Irony Detection in Italian Tweets with Character-Level Convolutional RNN. In *Proceedings of the 6th evaluation campaign of Natural Language Processing and Speech tools for Italian (EVALITA'18)*, Turin, Italy. CEUR.org.

Aditya Joshi, Pushpak Bhattacharyya, and Mark James Carman. 2017. Automatic sarcasm detection: A survey. *ACM Comput. Surv.*, 50(5):73:1–73:22.

Petra Kralj Novak, Jasmina Smailović, Borut Sluban, and Igor Mozetič. 2015. Sentiment of emojis. *PLOS ONE*, 10(12):1–22, 12.

Christopher J. Lee and Albert N. Katz. 1998. The differential role of ridicule in sarcasm and irony. *Metaphor and Symbol*, 13(1):1–15.

A. Marchetti, D. Massaro, and A. Valle. 2007. *Non dicevo sul serio. Riflessioni su ironia e psicologia*. Collana di psicologia. Franco Angeli.

Saif M Mohammad, Parinaz Sobhani, and Svetlana Kiritchenko. 2017. Stance and sentiment in tweets. *ACM Transactions on Internet Technology (TOIT)*, 17(3):26.

Reynier Ortega-Bueno and José E. Medina Pagola. 2018. UO_IRO: Linguistic informed deep-learning model for irony detection. In *Proceedings of the 6th evaluation campaign of Natural Language Processing and Speech tools for Italian (EVALITA'18)*, Turin, Italy. CEUR.org.

Irene Russo, Francesca Frontini, and Valeria Quochi. 2016. OpeNER sentiment lexicon italian - LMF. ILC-CNR for CLARIN-IT repository hosted at Institute for Computational Linguistics "A. Zampolli", National Research Council, in Pisa.

Magnus Sahlgren. 2005. An introduction to random indexing. In *In Methods and Applications of Semantic Indexing Workshop at the 7th International Conference on Terminology and Knowledge Engineering, TKE 2005*.

Manuela Sanguinetti, Fabio Poletto, Cristina Bosco, Viviana Patti, and Marco Stranisci. 2018. An Italian Twitter Corpus of Hate Speech against Immigrants. In *Proceedings of the 11th International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan.

Andrea Santilli, Danilo Croce, and Roberto Basili. 2018. A Kernel-based Approach for Irony and Sarcasm Detection in Italian. In *Proceedings of the 6th evaluation campaign of Natural Language Processing and Speech tools for Italian (EVALITA'18)*, Turin, Italy. CEUR.org.

Marco Stranisci, Cristina Bosco, Delia Irazú Hernández Farías, and Viviana Patti. 2016. Annotating Sentiment and Irony in the Online Italian Political Debate on #labuonascuola. In *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016)*, Portorož, Slovenia. ELRA.

Emilio Sulis, D. Irazú Hernández Farías, Paolo Rosso, Viviana Patti, and Giancarlo Ruffo. 2016. Figurative messages and affect in Twitter: Differences between #irony, #sarcasm and #not. *Knowledge-Based Systems*, 108:132 – 143. New Avenues in Knowledge Bases for Natural Language Processing.

Cynthia Van Hee, Els Lefever, and Véronique Hoste. 2018. Semeval-2018 task 3: Irony detection in English tweets. In *Proceedings of The 12th International Workshop on Semantic Evaluation*.

Appendix: Detailed results per class for all tasks

ranking	team name	run type	run id	precision (non-ironic)	recall (non-ironic)	F1-score (non-ironic)	precision (ironic)	recall (ironic)	F1-score (ironic)	average F1-score
1	ItaliaNLP	c	1	0.785	0.643	0.707	0.696	0.823	0.754	0.731
2	ItaliaNLP	c	2	0.751	0.643	0.693	0.687	0.786	0.733	0.713
3	UNIBA	c	1	0.748	0.638	0.689	0.683	0.784	0.730	0.710
4	UNIBA	c	2	0.748	0.638	0.689	0.683	0.784	0.730	0.710
5	X2Check	u	1	0.700	0.716	0.708	0.708	0.692	0.700	0.704
6	UNITOR	c	1	0.778	0.577	0.662	0.662	0.834	0.739	0.700
7	UNITOR	u	2	0.764	0.593	0.668	0.666	0.816	0.733	0.700
8	X2Check	u	2	0.690	0.712	0.700	0.701	0.678	0.689	0.695
9	Aspie96	c	1	0.742	0.606	0.668	0.666	0.789	0.722	0.695
10	X2Check	c	2	0.716	0.645	0.679	0.676	0.743	0.708	0.693
11	X2Check	c	1	0.697	0.652	0.674	0.672	0.715	0.693	0.683
12	UO_IRO	u	2	0.722	0.517	0.603	0.623	0.800	0.700	0.651
13	UO_IRO	u	1	0.667	0.590	0.626	0.631	0.703	0.665	0.646
14	UO_IRO	c	2	0.687	0.501	0.579	0.606	0.770	0.678	0.629
15	UO_IRO	c	1	0.600	0.714	0.652	0.645	0.522	0.577	0.614
16	*baseline-random*	*c*	*1*	*0.506*	*0.501*	*0.503*	*0.503*	*0.508*	*0.506*	*0.505*
17	venses-itgetarun	c	1	0.520	0.872	0.651	0.597	0.191	0.289	0.470
18	venses-itgetarun	c	2	0.505	0.892	0.645	0.525	0.120	0.195	0.420
19	*baseline-mfc*	*c*	*1*	*0.501*	*1.000*	*0.668*	*0.000*	*0.000*	*0.000*	*0.334*

Detailed results of Task A (Irony Detection)

ranking	team name	run type	run id	precision (non-ironic)	recall (non-ironic)	F1-score (non-ironic)	precision (ironic)	recall (ironic)	F1-score (ironic)	precision (sarcastic)	recall (sarcastic)	F1-score (sarcastic)	average F1-score
1	UNITOR	u	2	0.764	0.593	0.668	0.362	0.584	0.447	0.492	0.407	0.446	0.520
2	UNITOR	c	1	0.778	0.577	0.662	0.355	0.553	0.432	0.469	0.449	0.459	0.518
3	ItaliaNLP	c	1	0.785	0.643	0.707	0.343	0.584	0.432	0.518	0.338	0.409	0.516
4	ItaliaNLP	c	2	0.751	0.643	0.693	0.340	0.562	0.423	0.507	0.319	0.392	0.503
5	Aspie96	c	1	0.742	0.606	0.668	0.353	0.575	0.438	0.342	0.250	0.289	0.465
6	*baseline-random*	*c*	*1*	*0.506*	*0.501*	*0.503*	*0.267*	*0.265*	*0.266*	*0.239*	*0.245*	*0.242*	*0.337*
7	venses-itgetarun	c	1	0.606	0.334	0.431	0.341	0.210	0.260	0.500	0.009	0.018	0.236
8	*baseline-mfc*	*c*	*1*	*0.501*	*1.000*	*0.668*	*0.000*	*0.000*	*0.000*	*0.000*	*0.000*	*0.000*	*0.223*
9	venses-itgetarun	c	2	0.559	0.327	0.413	0.296	0.132	0.183	0.000	0.000	0.000	0.199

Detailed results of Task B (Sarcasm Detection)

Overview of the EVALITA 2018 Cross-Genre Gender Prediction (GxG) Task

Felice Dell'Orletta
ItaliaNLP Lab, ILC-CNR
Pisa, Italy
felice.dellorletta@ilc.cnr.it

Malvina Nissim
CLCG, University of Groningen
The Netherlands
m.nissim@rug.nl

Abstract

English. The Gender Cross-Genre (**GxG**) task is a shared task on author profiling (in terms of gender) on Italian texts, with a specific focus on cross-genre performance. This task has been proposed for the first time at EVALITA 2018, providing different datasets from different textual genres: Twitter, YouTube, Children writing, Journalism, Personal diaries. Results from a total of 50 different runs show that the task is difficult to learn in itself: while almost all runs beat a 50% baseline, no model reaches an accuracy above 70%. We also observe that cross-genre modelling yields a drop in performance, but not as substantial as one would expect.

Italiano. ***GxG*** *(Gender Cross-Genre) è la prima campagna di valutazione per l'identificazione del genere di un autore di testi scritti in lingua italiana e fa parte di quell'area di studio detta author profiling. In questa edizione di* ***GxG*** *particolare attenzione è stata posta nella valutazione dei sistemi in contesti di analisi cross-dominio. I domini testuali presi in esame sono stati: Twitter, YouTube, Children writing, Journalism, Personal diaries. I risultati ottenuti da un totale di 50 diverse run (prodotte da tre diversi gruppi di ricerca) mostrano che il task è complesso: mentre quasi tutti i sistemi testati superano la baseline del 50%, nessun modello raggiunge un'accuratezza superiore al 70%. Si osserva inoltre che i risultati raggiunti nel contesto di analisi cross-dominio mostrano un calo delle prestazioni non così sostanziale come ci si sarebbe potuto aspettare.*

1 Introduction

As publishing has become more and more accessible and basically cost-free, virtually anyone can get their words spread, especially online. Such ease of disseminating content doesn't necessarily go together with author identifiability. In other words: it's very simple for anyone to publicly write any text, but it isn't equally simple to always tell who the author of a text is.

In the interest of companies who want to advertise, or legal institutions, finding out at least some characteristics of an author is of crucial importance. *Author profiling* is the task of automatically discovering latent user attributes from text. Gender, which we focus on in this paper, and which is traditionally characterised as a binary feature, is one of such attributes.

Thanks to a series of tasks introduced at the PAN Labs in the last five years (pan.webis.de, and the production of a variety of gender-annotated datasets focused on social media (Verhoeven et al., 2016; Emmery et al., 2017, e.g.), gender prediction has been addressed quite substantially in NLP. State-of-the-art gender prediction on Twitter for English, the most common platform and language used for this task, is approximately 80–85% (Rangel et al., 2015; Alvarez-Carmona et al., 2015; Rangel et al., 2017; Basile et al., 2017), as obtained at the yearly PAN evaluation campaigns (pan.webis.de).

However, in the context of the 2016 PAN evaluation campaign, a cross-genre setting was introduced for gender prediction on English, Spanish, and Dutch, and best scores were recorded at an average accuracy of less than 60% (Rangel et al., 2016). This was achieved by training models on tweets, and testing them on datasets from a different source still in the social media domain, namely blogs. To further explore the cross-genre issue, (Medvedeva et al., 2017) ran additional experi-

ments using PAN data from previous years with the model that had achieved best results at the cross-genre PAN 2016 challenge (Busger op Vollenbroek et al., 2016). The picture they obtain is mixed in terms of accuracy of cross-genre performance, eventually showing that models are not yet general enough to capture gender accurately across different datasets.

This is evidence that we have not yet found the actual dataset-independent features that do indeed capture the way females and males might write differently. To address this issue, we have designed a task specifically focused on cross-genre gender detection, and launched it within the EVALITA 2018 Campaign (Caselli et al., 2018). The rationale behind the cross-genre settings is as follows: if we can make gender prediction stable across very different genres, then we are more likely to have captured deeper gender-specific traits rather than dataset characteristics. As a by product, this task yields a variety of models for gender prediction in Italian, also shedding light on which genres favour or discourage in a way gender expression, by looking at whether they are easier or harder to model.

While Italian has featured in multi-lingual gender prediction at PAN (Rangel et al., 2015), this is the first task that addresses author profiling for Italian specifically, within and across genres.

2 Task

GxG (Gender Cross-Genre) is a task on author profiling (in terms of gender) on Italian texts, with a specific focus on cross-genre performance.

Given a (collection of) text(s) from a specific genre, the gender of the author has to be predicted. The task is cast as a binary classification task, with gender represented as F (female) or M (male).

Evaluation settings were designed bearing in mind the question at the core of this task: are there indicative traits across genres that can be leveraged to model gender in a rather genre-independent way?

We hoped to provide some answers to this question by making participants train and test their models on datasets from different genres. For comparison, participants were also recommended to submit genre-specific models, i.e., tested on the very same genre they were trained on. In-genre modelling can (i) shed light on which genres might be easier to model, i.e. where gender traits are more prominent; and (ii) make it easier to quantify the loss when modelling gender across genres. Therefore, the gender prediction task must be done in two ways:

- using a model which has been trained on the same genre
- using a model which has been trained on anything but that genre.

We selected five different genres (Section 3), and asked participants to submit up to ten different models, as per the overview in Table 1. Obviously, if one participant wanted to have one single model for everything, they could submit one model for all settings. In the cross-genre setting, the only constraint is **not** using in training any single instance from the genre they are testing on. Other than that, participants were free to combine the other datasets as they wished.

Participants were also free to use external resources, provided the cross-genre settings were carefully preserved, and everything used was described in detail in their final report.

Measures As standardly done in binary classification tasks with balanced classes (see Section 3), we will evaluate performance using accuracy.

For each of the 10 models, five in the in-genre settings, and five in the cross-genre settings, we calculate the accuracy for the two classes, i.e. F and M. In order to derive two final scores, one for the in-genre and of for the cross-genre settings, we will simply average over the five accuracies obtained per genre. In-genre:

$$Acc^{in-genre} = \frac{\sum_{j=1}^{5} Acc_j^{in-genre}}{5}$$

and cross-genre:

$$Acc^{cross-genre} = \frac{\sum_{j=1}^{5} Acc_j^{cross-genre}}{5}$$

We keep the two scorings separate. For determining the official "winner", we can consider the cross-genre ranking, as more specific to this task.

Baselines For all settings, given that the datasets are balanced for gender distribution, through random assignment we will have 50% accuracy.

Table 1: Models to submit.

IN-GENRE	CROSS-GENRE
Twitter in-genre model	non-Twitter model for Twitter
YouTube in-genre model	non-YouTube model for YouTube
Children in-genre model	non-Children model for Children
Journalism in-genre model	non-Journalism model for Journalism
Diaries in-genre model	non-Diaries model for Diaries

3 Data

In order to test the portability and stability of profiling models across genres, we created datasets from five genres. We describe them below, together with the format and train/test split of the materials distributed to participants. In Figure 1 we provide a few samples to illustrate the variety of the data, and the format provided to the participations (Section 3.3).

3.1 Genres

We selected data from the following genres grounding our choice of both availability and wide variety.

Twitter Tweets were downloaded using the Twitter API and a language identification module to restrict the selection to Italian messages[1]. Names from usernames were matched with a list of unambiguous male and female names.

YouTube YouTube comments were scraped using the YouTube API and an available scraper[2]. Videos were pre-selected manually with the aim to avoid gender biases, resulting in a selection from a few general topics: travel, music, documentaries, politics. The names of the comments' authors are visible, and gender was automatically assigned via matching first names to the same list of male and female proper names used for the Twitter dataset.

Children writing This dataset is a collection of essays written by Italian L1 learners collected during the first and second year of lower secondary school called CItA (Corpus Italiano di Apprendenti L1, (Barbagli et al., 2016)). CItA contains essays written by the same students chronologically ordered and covering a two-year temporal span. The corpus contains a total of 1,352 essays written by 153 students the first year and 155 the second year. It was collected during the two school years 2012–2013 and 2013–2014.

News/journalism This dataset was created scraping two famous Italian online newspapers (*La Repubblica* and *Corriere della Sera*) and selecting only single-authored newspaper articles. Gender assignment was done manually.

Personal diaries In order to include personal writing which is more distant from social media, we collected personal diaries that are freely available as part of the Fondazione Archivio Diaristico Nazionale della Città di Pieve Santo Stefano.[3] The documents are of varying but comparable sizes, and the author's name is clearly specified in their metadata. Gender assignment was done manually.

3.2 Train and test sets

For each genre we have a portion of training and a portion of test data. The distribution of gender labels was controlled for in each dataset (50/50). Additionally, we aimed at providing sets of comparable sizes in terms of tokens so as to avoid including training size as a relevant factor. This was intended for test, too, so as to have the same amount of evaluation samples, but due to limited availability we eventually used a smaller test set for the Diary genre.

Table 2 shows the size of the training and the test sets in terms of tokens and authors. The datasets are composed by texts written by multiple users, with possibly multiple documents per user. It is also possible that in the Twitter and YouTube datasets, different texts by the same user ended up both in training and in test. For what concerns the Children writing dataset, training and test contain texts written by same 155 children. Differently from the Children training set, the Children test set is composed by texts written on the same

[1] https://developer.twitter.com/

[2] https://github.com/philbot9/youtube-comment-scraper

[3] http://archiviodiari.org/index.php/iniziative-e-progetti/brani-di-dirai.html.

Twitter

```
<doc id="778" genre="twitter" gender="M">
@edmond644 @ilsussidiario Sarebbe vero se li avessimo eletti ma,
non avendolo fatto, "altri" se li meritano.
</doc>
```

Children

```
<doc id="1" genre="children" gender="M">
Questa estate mi sono divertito molto perché mio padre ha preso
casa nella località del Circeo. La casa era a due piani, al piano
terra c'era un giardino dove il mio gatto sela spassava.   C'era
molta ombra nel giardino e io mi ci addormivo sempre. Il mare era
poco lontano da casa e ci andavamo ogni giorno e giocavamo a fare
i subacquei. Siamo andati a mangiare la pizza  fuori ed era molto
buona.
</doc>
```

YouTube

```
<doc id="8493" genre="youtube" gender="F">
alla fine esce sempre il tuo lato gattaro! sei forte! bellissimo
video come sempre!
</doc>
```

Journalism

```
<doc id="118" genre="journalism" gender="F">
Elogio alla longevità, l'intervista bresciana a Rita Levi
Montalcini
Trent'anni fa il Nobel a Rita Levi Montalcini. Ecco l'ultima
intervista bresciana a cura di Luisa Monini: «I giovani credano
nei valori, i miei collaboratori sono tutte donne»
Tra le numerose interviste che Rita Levi Montalcini ha avuto la
bontà di concedermi, mi piace ricordare l'ultima, quella dei suoi
100 anni. Eravamo nello studio della sua Fondazione e lei era
particolarmente serena, disponibile. Elegante come sempre. [...]
</doc>
```

Diaries

```
<doc id="107" genre="diary" gender="F">
23.9.80
Sergio, volutamente stai coinvolgendo Alessandro in questa nostra
situazione, invece di tenerlo fuori: sai quanto è sensibile,
quanto è fragile, quanto è difficile anche - né puoi ignorare
che non solo lui in particolare ma nessun ragazzino di 14 anni
è in grado di subire o di affrontare o di sostenere una prova così
dolorosa.
Lo stai distruggendo, impedendogli di riflettere da solo,
martellando di parole (o scritti addirittura, come quella tua
dichiarazione) per sentirti meno solo o per annullare la sua
volontà e imporgli la tua, come volevi fare con me: ma non ti
rendi conto che non è amore il tuo, [...]
</doc>
```

Figure 1: Sample instances of all five genres from the training sets, as distributed to participants. Children, Diaries, and Journalism samples are cut due to space constraints.

Table 2: Size of datasets and label distribution.

	TRAINING			TEST		
Genre	**F**	**M**	**Tokens**	**F**	**M**	**Tokens**
Children	100	100	65,986	100	100	48,913
Diaries	100	100	82,989	37	37	22,209
Journalism	100	100	113,437	100	100	112,036
Twitter	3000	3000	101,534	3000	3000	129,846
Youtube	2200	2200	90,639	2200	2200	61,008

topic at the same time, at the end of the two school years. For News and Diaries we made sure no author was included in both training and test. We did not balance the number of users per genre, nor the number of documents per user, assuming these as rather natural conditions.

3.3 Format

The data was distributed as simil-XML. The format can be seen in Figure 1. Although we distributed one file per genre, we still included the genre information in the XML so as to ease the combination of the different files.

4 Participants and Results

Following a call for interest, 15 teams registered for the task and thus obtained the training data. Eventually, three teams submitted their predictions, for a total of 50 runs. Three different runs were allowed per task, and one team experimented with three different models submitting three different predictions for each of the 10 subtasks. A summary of participants is provided in Table 3, while Tables 4 and 5 report the final results on the test sets of the EVALITA 2018 GxG Task.

CapetownMilanoTirana proposed a classifier based on Support Vector Machine (SVM) as learning algorithm. They tested different n-gram features extracted at the word level as well as at the character level. In addition, they experimented feature abstraction transforming each word into a list of symbols and computing the length of the obtained word and its frequency (Basile et al., 2018).

UniOr tested several binary classifiers based on different learning algorithms. For the official run, they used their two best systems based on Logistic Regression (LR) and Random Forest (RF) depending on the dataset analyzed. As features, they exploited linguistic parameters extracted using stylometric analysis, such as the vocabulary richness, use of the first or third person, etc.[4]

ItaliaNLP tested three different classification models: one based on linear SVM, and two based on Bi-directional Long Short Term Memory (Bi-LSTM). The two deep neural network architectures use 2-layers of Bi-LSTM. The first Bi-LSTM layer encodes each sentence as a token sequence, the second layer encodes the sentence sequence. These two architectures differ in the learning approaches they use: Single-Task Learning (STL) and Multi-Task Learning (MTL) (Cimino et al., 2018).

5 Analysis and Discussion

In this section we provide both a discussion of the approaches and an analysis of the results.

5.1 Approaches

Participants experimented with more classical machine learning approaches as well as with neural networks. Results show that while neural models achieve globally more accurate results, feature engineered SVMs are as competitive. This holds both in the in-genre and in the cross-genre settings.

All models suffer to some extent from the shift to cross-genre, though the CapetownMilanoTirana-SVM system appears to be the most robust. This might be due more to the choice of (abstract) features, rather than the learning algorithm itself. This system also employs *bleaching* (a technique to fade out lexicon in favour of more abstract token representation) in this GxG cross-genre setting, after it had shown promise in a cross-lingual profiling task, where it was firstly introduced (van der Goot et al., 2018). However, from their cross-validation

[4]The participation of this team was not followed by a system description paper.

Table 3: Participants to the EVALITA 2018 GxG Task with number of runs.

Team Name	**Research Group**	**# Runs**
CapetownMilanoTirana	Symanto Research, CoGrammar, freelance researcher	10
UniOr	Università Orientale di Napoli	10
ItaliaNLP Lab	ItaliaNLP Lab, ILC-CNR, Pisa	30

Table 4: Results in terms of Accuracy of the EVALITA 2018 GxG In-Domain Task.

Team Name-Model	**CH**	**DI**	**JO**	**TW**	**YT**	TOT
CapetownMilanoTirana-SVM	0.615	0.635	0.480	0.545	0.547	0.564
UniOr-LR-RF	0.550	0.550	**0.585**	0.49	0.500	0.535
ItaliaNLP Lab-SVM	0.550	0.649	0.555	0.567	**0.555**	0.575
ItaliaNLP Lab-STL	0.545	0.541	0.500	**0.595**	0.512	0.538
ItaliaNLP Lab-MTL	**0.640**	**0.676**	0.470	0.561	0.546	0.578
avg-Accuracy	0.580	0.610	0.518	0.552	0.532	0.558

Table 5: Results in terms of Accuracy of the EVALITA 2018 GxG Cross-Domain Task.

Team Name-Model	**CH**	**DI**	**JO**	**TW**	**YT**	TOT
CapetownMilanoTirana-SVM	0.535	**0.635**	**0.515**	0.555	0.503	0.549
UniOr-LR-RF	0.525	0.550	0.415	0.500	0.500	0.498
ItaliaNLP Lab-SVM	0.540	0.514	0.505	0.586	**0.513**	0.532
ItaliaNLP Lab-STL	**0.640**	0.554	0.495	**0.609**	0.510	0.562
ItaliaNLP Lab-MTL	0.535	0.595	0.510	0.500	0.500	0.528
avg-Accuracy	0.555	0.570	0.488	0.550	0.505	0.534

results on training data, where they also perform an evaluation of feature contribution, it seems that bleaching in this context does not yield the expected benefits (Basile et al., 2018).

The use of external resources was globally little explored, with the exception of generic word embeddings (ItaliaNLP Lab). While such embeddings do not seem to have contributed much to performance, specialised lexica or embeddings could be something to be investigated in the future.

From a learning settings' perspective, teams chose quite straightforward strategies. In-genre, all models were trained using data from the target genre only, with the exception of the ItaliaNLP Lab-MTL model, where the adopted multi-task strategy could use the knowledge from other genres, even in the in-genre settings. This seems confirmed comparing the ItaliaNLP Lab-MTL's results with the twin model ItaliaNLP Lab-STL (the same architecture with a single-task setting).

In the cross-genre scenario, all systems have used as training all of the available datasets apart from the dataset from the target genre. It appears that no team tried to exploit functions of (dis)similarity among genres in order to select sub-portions of data for training when testing on a given genre. The only different model in the way the training data from the other genres is used is the ItaliaNLP Lab-MTL, but its performance on the cross-genre setting indicates that this approach is not robust for this task.

5.2 Results

The in-domain results (Table 4) are useful to identify which genres are overall easier to model in terms of the author's gender, and provide an overview of gender detection in Italian.

As could be expected, Diaries are the easiest genre to model. This might result from the fact that the texts are longer, and are characterised by a more personal and subjective writing style. For example, the collected diaries present an extensive use of the first and second singular person verbs and a higher distribution of possessive adjectives.

Due to availability, the Diaries test set is smaller than the others, providing thus fewer instances for evaluation (see Table 2) and possibly weaker reliability of results. However, from the analysis of results reported by (Basile et al., 2018), we see that even in the cross-validated training set (100 samples), accuracy on Diaries is the highest out of the five genres.

We also see that Children writings carry better signal towards gender detection than social media. This might be due to the fact that the Children test set is composed by documents characterised by a common prompt (in the original collection settings, this was meant to provide evidence of how students perceive the different writing instructions received in the considered school years). This feature makes the children texts a reflexive textual typology, typically characterised by a more subjective writing style as we observed also for Diaries.

Both Twitter and YouTube score above a 50% baseline, but are clearly harder to model. This could be due to the short texts, which in some cases offer very little evidence to go by. Compared to previous results reported for gender detection on Twitter in Italian, as obtained at the 2015 PAN Lab challenge on author profiling (Rangel et al., 2015), and on the TwiSty dataset (Verhoeven et al., 2016), scores at GxG are substantially lower (PAN 2015's best performance on Italian: .8611, TwiSty's reported F-measure: .7329). The reason for this could be the fact that in the PAN Lab and Twisty datasets authors are represented by a collection of tweets, while we do not control for this aspect at all, under the assumption that if gender traits emerge, these should not depend on having large evidence from a single author. It could therefore be that the PAN's and TwiSty's results are inflated by partially modelling authors rather than gender. Another interesting observation regarding Twitter is that when cross-validating the training set, (Basile et al., 2018) report accuracy in the 70s, while their in-genre results on the test data are just above 50% (Table 4).

The most difficult genre is Journalism. While texts can be as long as diaries, results suggest that the requested jargon and writing conventions typical of this genre overshadow the gender signal. Moreover, while we have selected only articles written by a single author, there is always the chance that revisions are made by an editor before the piece is published. This is the only genre where some models fail to beat the 50% baseline. However, it is interesting to note that the highest score in-genre is achieved by UniOr, which uses a selection of stylometric features (Koppel et al., 2002; Schler et al., 2006, e.g.), which have long been thought to capture unconscious behaviour better than just lexical choice (on which most of the other models are based, as they mainly use n-grams). The highest Journalism score in the cross-settings is achieved by CapetownMilanoTirana.

Cross-genre, we observe that results are on average lower, but only by 2.5 percentage points (55.8 vs 53.4), which is less than one would expect. Some models clearly drop more heavily from in-genre to cross-genre (ItaliaNLP Lab-MTL: .578 vs .528 average accuracy, ItaliaNLP Lab-SVM: .575 vs .532, UniOr: .535 vs .498). However, others appear more stable in both settings (CapetownMilanoTirana-SVM: .564 vs .549), or even better at the cross- rather than in-genre prediction (ItaliaNLP Lab-STL: .538 vs .562).

From a genre perspective, the drop is more substantial for some genres, with Diaries losing the most, with large variation though across systems. For example, the model that achieves best performance on Diaries in-genre (ItaliaNLP Lab-MTL, .676) suffers a drop of almost eight percentage points on the same dataset cross-genre (.595). Conversely, CapetownMilanoTirana preserve a high performance on the Diaries testset in both in- and cross-settings (.635), yielding the highest cross-performance on this genre. Twitter shows the least variation between in- and cross-genre testing. Not only the losses for all systems are minimal, but in some cases we even observe higher scores in the cross-genre setting. ItaliaNLP Lab-STL obtains the highest score on this test set in both settings, but the performance is higher for cross- than in-genre (.609 vs .595).

Finally, some possibly interesting insights also emerge from looking at the precision and recall for the two classes (which we do not report in tables as there are too many data points to show). For example, we observe that for some genres only one gender ends up being assigned almost entirely by all classifiers. This happens in the cross-genre settings, while the same test set has rather balanced assignments of the two classes in the in-genre settings. In the case of Journalism, all systems in cross-genre modelling almost only predict female as the author's gender. At the opposite side of the

spectrum we find YouTube, where almost all test instances are classified as male by almost all systems, cross-genre. In this case though we also see high recall for male in the in-genre setting, though not so dominant. While more grounded considerations are left to a deeper analysis in the future, we could speculate that some genres are globally seen by classifiers as more characteristic of one gender or the other, as learnt from a large amount of mixed-genre, gender-balanced data.

6 Conclusions

Gender detection was for the first time the focus of a dedicated task at EVALITA. The GxG task specifically focussed on comparing performance within and across genres, building on previous observations that high performing systems were likely to be modelling datasets rather than gender, as their accuracy substantially dropped when tested on different, even related, domains.

Results from 50 different runs were mostly above baseline for most prediction tasks, both in-genre and cross-genre, but not particularly high overall. Also, the drop between in-genre and cross-genre performance is noticeable, but marginal. Neural models appear to perform only slightly better than a more classic SVM which leverages character and word n-grams. The use of the recently introduced *text bleaching* strategy among the engineered features (aimed at reducing lexicon bias (van der Goot et al., 2018)), does not seem to yield the desired performance in the cross-genre settings.

In the near future, it will be interesting to compare these results to human performance, as it has been observed that for profiling human do not perform usually better than machines (Flekova et al., 2016; van der Goot et al., 2018), to which they provide complementary performance in terms of correctly predicted instances.

Acknowledgments

We would like to thank Eleonora Cocciu and Claudia Zaghi for their help with data collection and annotation. We are also grateful to the EVALITA chairs for hosting this task.

References

Miguel A Alvarez-Carmona, A Pastor López-Monroy, Manuel Montes-y Gómez, Luis Villasenor-Pineda, and Hugo Jair Escalante. 2015. INAOE's participation at PAN'15: Author profiling task. In *CLEF Working Notes*.

Alessia Barbagli, Pietro Lucisano, Felice Dell'Orletta, Simonetta Montemagni, and Giulia Venturi. 2016. Cita: an l1 italian learners corpus to study the development of writing competence. In *In Proceedings of the 10 thConference on Language Resources and Evaluation (LREC 2016)*, pages 88–95.

Angelo Basile, Gareth Dwyer, Maria Medvedeva, Josine Rawee, Hessel Haagsma, and Malvina Nissim. 2017. N-gram: New groningen author-profiling model. In *Proceedings of the CLEF 2017 Evaluation Labs and Workshop – Working Notes Papers, 11-14 September, Dublin, Ireland (Sept. 2017)*.

Angelo Basile, Gareth Dwyer, and Chiara Rubagotti. 2018. *CapetownMilanoTirana* for GxG at Evalita2018. Simple n-gram based models perform well for gender prediction. Sometimes. In Tommaso Caselli, Nicole Novielli, Viviana Patti, and Paolo Rosso, editors, *Proceedings of Sixth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2018)*, Turin, Italy. CEUR.org.

M. Busger op Vollenbroek, T. Carlotto, T. Kreutz, M. Medvedeva, C. Pool, J. Bjerva, H. Haagsma, and M. Nissim. 2016. Gron-UP: Groningen user profiling notebook for PAN at CLEF. In *CLEF 2016 Evaluation Labs and Workshop, Working Notes*.

Tommaso Caselli, Nicole Novielli, Viviana Patti, and Paolo Rosso. 2018. Evalita 2018: Overview of the 6th evaluation campaign of natural language processing and speech tools for italian. In Tommaso Caselli, Nicole Novielli, Viviana Patti, and Paolo Rosso, editors, *Proceedings of Sixth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2018)*, Turin, Italy. CEUR.org.

Andrea Cimino, Lorenzo De Mattei, and Felice Dell'Orletta. 2018. Multi-task Learning in Deep Neural Networks at EVALITA 2018. In Tommaso Caselli, Nicole Novielli, Viviana Patti, and Paolo Rosso, editors, *Proceedings of Sixth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2018)*, Turin, Italy. CEUR.org.

Chris Emmery, Grzegorz Chrupała, and Walter Daelemans. 2017. Simple queries as distant labels for predicting gender on twitter. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pages 50–55, Copenhagen, Denmark.

Lucie Flekova, Jordan Carpenter, Salvatore Giorgi, Lyle Ungar, and Daniel Preoţiuc-Pietro. 2016. Analyzing biases in human perception of user age and gender from text. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 843–854, Berlin, Germany, August.

Moshe Koppel, Shlomo Argamon, and Anat Rachel Shimoni. 2002. Automatically categorizing written texts by author gender. *Literary and Linguistic Computing*, 17:401–412.

Maria Medvedeva, Hessel Haagsma, and Malvina Nissim. 2017. An analysis of cross-genre and in-genre performance for author profiling in social media. In *Experimental IR Meets Multilinguality, Multimodality, and Interaction - 8th International Conference of the CLEF Association, CLEF 2017, Dublin, Ireland, September 11-14, 2017*, pages 211–223.

Francisco Rangel, Paolo Rosso, Martin Potthast, Benno Stein, and Walter Daelemans. 2015. Overview of the 3rd author profiling task at pan 2015. In *CLEF*.

Francisco Rangel, Paolo Rosso, Ben Verhoeven, Walter Daelemans, Martin Potthast, and Benno Stein. 2016. Overview of the 4th author profiling task at pan 2016: cross-genre evaluations. In *Working Notes Papers of the CLEF 2016 Evaluation Labs. CEUR Workshop Proceedings*, pages 750–784.

Francisco Rangel, Paolo Rosso, Martin Potthast, and Benno Stein. 2017. Overview of the 5th author profiling task at pan 2017: Gender and language variety identification in Twitter. *CLEF Working Notes*.

Jonathan Schler, Moshe Koppel, Shlomo Argamon, and James W. Pennebaker. 2006. Effects of Age and Gender on Blogging. In *AAAI Spring Symposium: Computational Approaches to Analyzing Weblogs*, pages 199–205. AAAI.

Rob van der Goot, Nikola Ljubesic, Ian Matroos, Malvina Nissim, and Barbara Plank. 2018. Bleaching text: Abstract features for cross-lingual gender prediction. In Iryna Gurevych and Yusuke Miyao, editors, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 2: Short Papers*, pages 383–389.

Ben Verhoeven, Walter Daelemans, and Barbara Plank. 2016. Twisty: A multilingual twitter stylometry corpus for gender and personality profiling. In *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016)*.

Overview of the Evalita 2018 itaLIan Speech acT labEliNg (iLISTEN) Task

Pierpaolo Basile and **Nicole Novielli**
Università degli Studi di Bari Aldo Moro
Dipartimento di Informatica
Via E. Orabona, 4 - 70125 Bari (ITALY)
{pierpaolo.basile|nicole.novielli}@uniba.it

Abstract

English. We describe the first edition of the " itaLIan Speech acT labEliNg" (iLISTEN) task at the EVALITA 2018 campaign (Caselli et al., 2018). The task consists in automatically annotating dialogue turns with *speech act labels*, i.e. with the communicative intention of the speaker, such as statement, request for information, agreement, opinion expression, or general answer. The task is justified by the large number of applications that could benefit from automatic speech act annotation of natural language interactions such as tools for the intelligent information access, that is by relying on natural dialogues. We received two runs from two teams, one from academia and the other one from industry. In spite of the inherent complexity of the tasks, both systems largely outperformed the baseline.

Italiano. *Descriviamo la prima edizione del task di "itaLIan Speech acT labEliNg" (iLISTEN) organizzato nell'ambito della campagna di valutazione EVALITA 2018. Il task consiste nell'annotazione automatica di turni di dialogo con la label di speech act corrispondente. Ciascuna categoria di speech act denota l'intenzione comunicativa del parlante, ossia l'intenzione di formulare un'affermazione oggettiva, l'espressione di un'opinione, la richiesta di informazioni, una risposta, un'espressione di consenso. Riteniamo che il task sia rilevante per la il dominio della linguistica computazionale e non solo, alla luce del recente interesse da parte della comunità scentifica nei confronti dei paradigmi di interazione e accesso intelligente all'informazione basati su dialogo. Il task ha visto la partecipazione di due team, uno accademico e uno industriale. Nonostante la complessità del task proposto, entrabi i team hanno ampiamente superato la baseline.*

1 Introduction

Speech acts have been extensively investigated in linguistics (Austin, 1962; Searle, 1969), and computational linguistics (Traum, 2000; Stolcke et al., 2000) since long. Specifically, the task of automatic speech act recognition has been addressed leveraging both supervised (Stolcke et al., 2000; Vosoughi and Roy, 2016) and unsupervised approaches (Novielli and Strapparava, 2011). This interest is justified by the large number of applications that could benefit from automatic speech act annotation of natural language interactions.

In particular, a recent research trend has emerged to investigate methodologies to enable intelligent access to information, that is by relying on natural dialogues as interaction metaphor. In this perspective, chat-oriented dialogue systems are attracting the increasing attention of both research and practitioners interested in the simulation of natural dialogues with embodied conversational agents (Klüwer, 2011), conversational interfaces for smart devices (McTear et al., 2016) and the Internet of Things (Kar and Haldar, 2016). As a consequence, we are assisting to the flourishing of dedicated research venues on chat-oriented interaction. It is the case of WOCHAT[1], the Special Session on Chatbots and Conversational Agents, now at its second edition, as well as the Natural Language Generation for Dialogue Systems special session[2], both co-located with the Annual

[1] http://workshop.colips.org/wochat/@sigdial2017/

[2] https://sites.google.com/view/nlg4ds2017

SIGdial Meeting on Discourse and Dialogue.

While not representing any deep understanding of the interaction dynamics, speech acts can be successfully employed as a coding standard for natural dialogues tasks. In this report, we describe the first edition of the "itaLIan Speech acT labEliNg" (iLISTEN) task at the EVALITA 2018 campaign (Caselli et al., 2018). Among the various challenges posed by the problem of enabling conversational access to information, this shared task tackles the problem of recognition of the illocutionary force, i.e. the speech act, of a dialogue turn, that is the communicative goal of the speaker.

The remainder of the paper is organized as follows. We start by explaining the task in Section 2. In Section 3, we provide a detailed description of the dataset of dialogues, the annotation schema, and the data format and distribution protocol. Then, we report about the evaluation methodology (see Section 4) and describe the participating systems and their performance (see Section 5). We provide final remarks in Section 6.

2 Task Description

The task consists in automatically annotating dialogue turns with *speech act labels*, i.e. with the communicative intention of the speaker, such as statement, request for information, agreement, opinion expression, general answer, etc. Table 1 reports the full set of speech act labels used for the classification task, with definition, examples, and distribution in our corpus. Regarding the evaluation procedure, we assess the ability of each system to issue the correct speech act label among those included in the taxonomy used for annotation, described in the Section 3. Please, note that the participating systems are requested to issue labels only for the speech act used for labeling the user's dialogue turns, as futher detailed in the following.

3 Development and Test Data

3.1 A Dataset of Dialogues

We leverage the corpus of natural language dialogues collected in the scope of previous research about interaction with Embodied Conversational Agents (ECAs) (Clarizio et al., 2006), in order to speed up the process of building a gold standard. The corpus contains overall transcripts of 60 dialogues, 1,576 user dialogue turns, 1,611 system turns and about 22,000 words.

The dialogues were collected using a Wizard of Oz tool as dialogue manager. Sixty subjects (aged between 21–28) were involved in the study, in two interaction mode conditions: thirty of them interacted with the system in a written-input setting, using keyboard and mouse; the remaining thirty dialogues were collected with users interacting with the ECA in a spoken-input condition. The dialogues collected using the spoken interaction mode were manually transcribed based on audio-recording of the dialogue sessions.

During the interaction, the ECA played the role of an artificial therapist and the users were free to interact with it in natural language, without any particular constraint: they could simply answer the question of the agent or taking the initiative and ask questions in their turn, make comments about the agent behavior or competence, argument in favor or against the agent's suggestion or persuasion attempts. The Wizard, on his behalf, had to choose among a set of about 80 predefined possible system moves. As such, the system moves (see Table 2) are provided only as a context information but are not subject to evaluation and do not contribute to the final ranking of the participant systems. Conversely, the participating systems are evaluated on the basis of the performance observed for the user dialogue turns (see Table 1).

3.2 Annotation Schema

Speech acts can be identified with the communicative goal of a given utterance, i.e. it represents its meaning at the level of its *illocutionary force* (Austin, 1962). In defining dialogue act taxonomies, researchers have been trying to solve the trade-off between the need for formal semantics and the need for computational feasibility, also taking into account the specificity of the many application domains that have been investigated (see (Traum, 2000) for an exhaustive overview). The Dialogue Act Markup in Several Layers (DAMSL) represents an attempt by (Core and Allen, 1997) to define a domain independent framework for speech act annotation.

Defining a speech act markup language is out of the scope of the present study. Therefore, we adopt the original annotation of the Italian advice-giving dialogues. Table 1 shows the set of nine labels employed for the purpose of this study, with definitions and examples. These labels are used for the annotation of the users' dialogue turns and are the object of classification for this task. In ad-

Table 1: The set of *user* speech act labels employed in our annotation schema. The participating systems are required to issue a label for the user moves only.

Speech Act	Description	Example	Freq.
OPENING	Dialogue opening or self-introduction	*'Ciao, io sono Antonella'*	2%
CLOSING	Dialogue closing, e.g. farewell, wishes, intention to close the conversation	*'Va bene, ci vediamo prossimamente'*	2%
INFO-REQUEST	Utterances that are pragmatically, semantically, and syntactically questions	*'E cosa mi dici delle vitamine?'*	25%
SOLICIT-REQ-CLARIF	Request for clarification (please explain) or solicitation of system reaction	*'Mmm, si ma in che senso?'*	7%
STATEMENT	Descriptive, narrative, personal statements	*'Penso che dovrei controllare maggiormente il consumo di dolciumi.'*	33%
GENERIC-ANSWER	Generic answer	*'Si'*, *'No'*, *'Non so.'*	10%
AGREE-ACCEPT	Expression of agreement, e.g. acceptance of a proposal, plan or opinion	*'Si, so che è importante.'*	5%
REJECT	Expression of disagreement, e.g. rejection of a proposal, plan, or opinion	*'Ho sentito tesi contrastanti al proposito.'*	5%
KIND-ATT-SMALLTALK	Expression of kind attitude through politeness, e.g. thanking, apologizing or smalltalk	*'Thank you.'*, *'Sei per caso offesa per qualcosa che ho detto?'*	11%

dition, in Table 1 we report the speech act labels used for the dialogue moves of the system, i.e. the conversational agent playing the role of the artificial therapist. The speech act taxonomy refines the DAMSL categories to allow appropriate tagging of the communicative intention with respect to the application domain, i.e. persuasion dialogues in the healthy eating domain.

In Table 3 we provide an excerpt from a dialogue from our gold standard. The system moves (dialogue moves and corresponding speech act labels) are chosen from a set of predefined dialogue moves that can be played by the ECA. As such, they are not interesting for the evaluation and ranking of participating systems and are provided only as contextual information. Conversely, the final ranking of the participating systems is based on the performance observed only on the prediction of speech acts for the users' move, with respect to the set of labels provided in Table 1. Please, note that the two sets of speech act labels for the user and the system moves, in Table 1 and Table 2, respectively, only partially overlap. This is due to the fact that the set of agent's moves includes also speech acts (such as persuasion attempts) that are observed only for the agent, given its caregiver role in the dialogue systems. Vice versa, some speech act labels (such as clarification questions) are relevant only for the user moves.

3.3 Data Format and Distribution

We provide both the training and testing dialogues in the XML format following the structure proposed in Figure 1. Each participating initially had access to the training data only. Later, the unlabeled test data were released during the evaluation period. The development and test data set contain 40 and 20 dialogues, respectively, equally distributed with respect to the interaction mode (text- vs. speech-based interaction).

4 Evaluation

Regarding the evaluation procedure, we assess the ability of each system to issue the correct speech act label for the user moves. The speech act label used for annotation of the user moves are reported in Table 1.

Specifically, we compute precision, recall and F1-score (macroaveraging) with respect to our gold standard. This approach, while more verbose than a simple accuracy test, arise from the need to correctly address the unbalanced distribution of labels in the dataset. Furthermore, by providing detailed performance metrics, we intend to enhance interesting discussion on the nature of the problem and the data, as they might emerge from the participants' final reports. As a baseline, we use the most frequent label for the user speech acts (i.e., STATEMENT).

Table 2: The set of *system* speech act labels in our annotation schema. These labels are provided as context information, i.e. the participating systems are *not* required to issue a label for the system moves.

Speech Act	Description	Example	Freq.
OPENING	Initial self-introduction by the ECA	*'Ciao, il mio nome è Valentina e sono qui per darti suggerimenti su come migliorare la tua dieta.'*	4%
CLOSING	Dialogue closing, e.g. farewell, wishes, intention to close the conversation	*'Grazie e arrivederci!'*	4%
QUESTION	Question about the user eating habits or information interests	*'...'*	32%
TALK-ABOUT-SELF	Statement describing own abilities, role and skills	*'Non sono in grado di mangiare, e quindi non seguo diete particolari.'*	3%
ANSWER	Provision of generic information after a user request	*'Purtroppo sono poco esperta in questo campo.'*	17%
OFFER-GIVE-INFO	Generic offer of help or specific information about the diet domain.	*'Idealmente un pasto dovrebbe contenere da 75 a 100 grammi di carne, un frutto di media grandezza, un piccolo piatto di pasta o 50 grammi di gelato. I cibi pronti permettono di controllare le porzioni riportando sulla confezione il valore energetico in calorie.'*	8%
PERSUASION-SUGGEST	Persuasion attempt performed by the conversational agent about dieting.	*'Lo so: mangiare, e forse vivere, in modo disordinato, sembra a molti piacevole,e forse hanno ragione. Ma a lungo andare gli effetti negativi si fanno sentire.'*	24%
ENCOURAGE-SORRY	Statements aimed at enhancing the user motivation or to apologize.	*'Questa e' una buona idea! '*	9%

Table 3: An excerpt from one of the dialogues in our dataset.

Actor	ID	Speech Act	Dialogue Move
SYSTEM	5-S1	OPENING	Ciao, il mio nome è Valentina. Sono qui per darti dei suggerimenti su come migliorare la tua dieta
USER	5-U1	OPENING	Ciao, il mio nome è Oriana.
SYSTEM	5-S2	INFO-REQUEST	Quali sono le tue abitudini alimentari?
USER	5-U2	STATEMENT	Ho delle abitudini disordinate, mangio anche fuori orario.
SYSTEM	5-S3	INFO-REQUEST	Ti piace mangiare?
USER	5-U3	STATEMENT	Si, ma dipende dal periodo: se sono sotto stress mangio meno, quando sono più rilassata mangio molto
SYSTEM	5-S4	INFO-REQUEST	Segui una dieta variata?
USER	5-U3	GENERIC-ANSWER	No.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<dialogues>
  <dialogue>
    <speechAct act="OPENING" actId="13" id="T_5_S1">Ciao, il mio nome e' Valentina. Sono
qui per darti dei suggerimenti su come migliorare la tua dieta.</speechAct>
    <speechAct act="OPENING" actId="10" id="T_5_U1">ciao il mio nome è oriana</speechAct>
    <speechAct act="QUESTION" actId="12" id="T_5_S2">Quali sono le tue abitudini
alimentari?</speechAct>
    <speechAct act="STAT-ABOUT-SELF" actId="5" id="T_5_U2">ho delle abitudini un pò
disordinate,nel senso che mangio anche fupri orario</speechAct>
    <speechAct act="QUESTION" actId="12" id="T_5_S3">Ti piace mangiare?</speechAct>
    <speechAct act="STAT-ABOUT-SELF" actId="5" id="T_5_U2">si,ma dipende dal periodo,se
sono sotto stress mangio meno,quando sono più rilassata mangio molto</speechAct>
    <speechAct act="QUESTION" actId="12" id="T_5_S4">Segui una dieta variata?</speechAct>
    <speechAct act="GENERIC-ANSWER" actId="7" id="T_5_U4">no</speechAct>
    ...
```

Figure 1: Data format

Table 4: Overall micro- and macro-averaged Precision, Recall, and F-score for the participating systems

	Micro			***Macro***		
System	**Prec**	**Rec**	**F**	**Prec**	**Rec**	**F**
UNITOR.kelp	0.7328	0.7328	0.7328	0.6810	0.6274	0.6531
X2Check.c2c	0.6848	0.6848	0.6848	0.6076	0.5844	0.5957
Baseline	0.3403	0.3403	0.3403	0.0378	0.1111	0.0564

Table 5: Precision, Recall, and F-score values by speech act labels

	Unitor			*X2Check*		
Class	**Prec**	**Rec**	**F**	**Prec**	**Rec**	**F**
OPENING	1.0000	1.0000	1.0000	1.0000	0.7273	0.8421
CLOSING	0.7778	0.7000	0.7368	0.8182	0.9000	0.8571
INFO-REQUEST	0.7750	0.8304	0.8017	0.7355	0.7946	0.7639
SOLICITATION-REQ-CLARIF	0.4000	0.3333	0.3636	0.4444	0.3333	0.3810
STATEMENT	0.7500	0.9444	0.8361	0.6667	0.8957	0.7644
GENERIC-ANSWER	0.8571	0.9231	0.8889	0.7581	0.9038	0.8246
AGREE-ACCEPT	0.6471	0.4583	0.5366	0.5714	0.5000	0.5333
REJECT	0.4286	0.0769	0.1304	0.0000	0.0000	0.0000
KIND-ATT-SMALLTALK	0.5000	0.3864	0.4359	0.4737	0.2045	0.2857

5 Participants and Results

The task was open to everyone from industry and academia. Sixteen participants registered, but only two teams actually submitted the results for the evaluation. A short description of each system follows:

UNITOR - The system described in (Croce and Basili, 2018) is a supervised system which relies on a Structured Kernel-based Support Vector Machine for making the classification of the dialogue turns sensitive to the syntactic and semantic information of each utterance. The Structured Kernel is a Smoothed Partial Tree Kernel (Croce et al., 2011) that exploits both the parse tree and the cosine similarity between the word vectors in a distributional semantics model. The authors use the tree parser provided by SpaCy[3] and the Kelp framework[4] for SVM.

X2Check - The team did not submit the report.

The performance of the participating systems is evaluated based on the macro (and micro) precision and recall (Sebastiani, 2002). However, the official task measure used to rank the systems is the macro-F. Results are reported in Table 4.

[3] `https://spacy.io/`

[4] KeLP is a Java Kernel-based Learning Platform: `http://www.kelp-ml.org/`

The best performance (0.6531) is provided by the UNITOR system. Both systems are able to overcome the baseline also for micro-F. The baseline has a low macro-F since it predicts always the same class (STATEMENT) and for the other classes the F-measure is zero. As expected, the micro-F overcomes the macro-F since some classes are hard to predict due to the low number of examples in the training data, such as AGREE, SOLICITATION-REQ-CLARIF and REJECT. Precision, Recall, and F-score values by speech act labels are showed in Table 5.

We also provide the confusion matrix for each system, respectively Table 6 for UNITOR and Table 7 for X2Check. We observe that, for both systems, the class REJECT is the most difficult to classify. This evidence is consistent with the findings from previous research on the same corpus of dialogues (Novielli and Strapparava, 2011). In particular, we observe that dialogue moves belonging to the REJECT class are often misclassified as STATEMENT. More in general, the main cause of error is the misclassification as STATEMENT. One possible reason is that statements represent the majority class, thus inducing a bias in the classifiers. Another possible explanation, is that dialogue moves that appear to be linguistically consistent with the typical structure of statements have been annotated differently, according to the actual communicative role they play.

Table 6: Confusion Matrix of the UNITOR system w.r.t. gold standard. In column the number of classes from the gold standard, while rows report the system decisions. In bold correct classifications.

	STATEMENT	KIND-ATT.	GEN.-ANSW.	REJECT	CLOSING	SOL.-CLAR.	OPENING	AGREE	INFO-REQ.
STATEMENT	**153**	6	3	24	0	3	0	2	13
KIND-ATT.	4	**17**	0	5	1	2	0	3	2
GEN.-ANSW.	1	0	**48**	0	0	1	0	6	0
REJECT	0	3	0	**3**	0	0	0	0	1
CLOSING	0	0	0	0	**7**	1	0	1	0
SOL.-CLAR.	0	6	0	2	1	**8**	0	1	2
OPENING	0	0	0	0	0	0	**11**	0	0
AGREE	0	3	1	1	0	0	0	**11**	1
INFO-REQ.	4	9	0	4	1	9	0	0	**93**

Table 7: Confusion Matrix of the X2Check system w.r.t. gold standard. In column the number of classes from the gold standard, while rows report the system decisions. In bold correct classifications.

	STATEMENT	KIND-ATT.	GEN.-ANSW.	REJECT	CLOSING	SOL.-CLAR.	OPENING	AGREE	INFO-REQ.
STATEMENT	**146**	15	3	30	1	2	1	2	19
KIND-ATT.	2	**9**	0	0	0	1	0	5	2
GEN.-ANSW.	5	3	**47**	2	0	3	0	2	0
REJECT	0	0	0	0	0	0	0	0	0
CLOSING	0	0	0	1	**9**	0	0	1	0
SOL.-CLAR.	1	4	0	2	0	**8**	1	0	2
OPENING	0	0	0	0	0	0	**8**	0	0
AGREE	2	5	1	0	0	1	0	**12**	0
INFO-REQ.	7	8	1	4	0	9	1	2	**89**

6 Final Remarks and Conclusions

We presented the first edition of the new shared task about itaLIan Speech acT labEliNg (iLISTEN) at EVALITA 2018. The task fits in the fast-growing research trend focusing on conversational access to the information, e.g. using chatbots or conversational agents. The task consists in automatically annotating dialogue turns with speech act labels, representing the communicative intention of the speaker. The corpus of dialogues has been collected in the scope of previous research on natural language interaction with embodied conversational agents. Specifically, the participating systems had to annotate the speech acts associated to the user dialogue moves while the agent's dialogue turns were provided as context.

We received two runs from two teams, one from academia and the other one from industry. In spite of the inherent complexity of the tasks, both systems largely outperformed the baseline, represented by the trivial classifier always predicting the majority class for users' moves. The best performing system leverages syntactic features and relies on a Structured Kernel-based Support Vector Machine. Follow-up editions might involve extending the benchmark with dialogues from different domains. Similarly, dialogues in different languages might be also included in the gold standard, as done for Automatic Misogyny Identification task at EVALITA 2018 (Fersini et al., 2018). This would enable to assess to what extent the task is inherently dependent on the language and how the proposed approaches are able to generalize.

References

John L. Austin. 1962. *How to do things with words*. William James Lectures. Oxford University Press.

Tommaso Caselli, Nicole Novielli, Viviana Patti, and Paolo Rosso. 2018. EVALITA 2018: Overview of the 6th Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. In Tommaso Caselli, Nicole Novielli, Viviana Patti, and Paolo Rosso, editors, *Proceedings of Sixth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2018)*, Turin, Italy. CEUR.org.

Giuseppe Clarizio, Irene Mazzotta, Nicole Novielli, and Fiorella De Rosis. 2006. Social attitude towards a conversational character. pages 2–7.

Mark G. Core and James F. Allen. 1997. Coding Dialogs with the DAMSL Annotation Scheme.

Danilo Croce and Roberto Basili. 2018. A Markovian Kernel-based Approach for itaLIan Speech acT labEliNg. In Tommaso Caselli, Nicole Novielli, Viviana Patti, and Paolo Rosso, editors, *Proceedings of the 6th evaluation campaign of Natural Language Processing and Speech tools for Italian (EVALITA'18)*, Turin, Italy. CEUR.org.

Danilo Croce, Alessandro Moschitti, and Roberto Basili. 2011. Structured lexical similarity via convolution kernels on dependency trees. In *Proceedings of EMNLP*.

Elisabetta Fersini, Debora Nozza, and Paolo Rosso. 2018. Overview of the Evalita 2018 Task on Automatic Misogyny Identification (AMI). In Tommaso Caselli, Nicole Novielli, Viviana Patti, and

Paolo Rosso, editors, *Proceedings of the 6th evaluation campaign of Natural Language Processing and Speech tools for Italian (EVALITA'18)*, Turin, Italy. CEUR.org.

Rohan Kar and Rishin Haldar. 2016. Applying Chatbots to the Internet of Things: Opportunities and Architectural Elements. *CoRR*, abs/1611.03799.

Tina Klüwer. 2011. "I Like Your Shirt" - Dialogue Acts for Enabling Social Talk in Conversational Agents. In *Intelligent Virtual Agents*, pages 14–27.

Michael McTear, Zoraida Callejas, and David Griol Barres. 2016. *The Conversational Interface: Talking to Smart Devices*. Springer International Publishing.

Nicole Novielli and Carlo Strapparava. 2011. Dialogue act classification exploiting lexical semantics. In *Conversational Agents and Natural Language Interaction: Techniques and Effective Practices*, chapter 4, pages 80–106. IGI Global.

John R. Searle. 1969. *Speech Acts: An Essay in the Philosophy of Language*. Cambridge University Press, Cambridge, London.

Fabrizio Sebastiani. 2002. Machine learning in automated text categorization. *ACM computing surveys (CSUR)*, 34(1):1–47.

Andreas Stolcke, Noah Coccaro, Rebecca Bates, Paul Taylor, Carol Van Ess-Dykema, Klaus Ries, Elizabeth Shriberg, Daniel Jurafsky, Rachel Martin, and Marie Meteer. 2000. Dialogue Act Modeling for Automatic Tagging and Recognition of Conversational Speech. *Comput. Linguist.*, 26(3):339–373, September.

David R. Traum. 2000. 20 Questions for Dialogue Act Taxonomies. *Journal of Semantics*, 17(1):7–30.

Soroush Vosoughi and Deb Roy. 2016. A Semi-automatic Method for Efficient Detection of Stories on Social Media. In *Proc. of the 10th AAAI Conf. on Weblogs and Social Media.*, ICWSM 2016, pages 711–714.

Overview of the EVALITA 2018 Evaluation of Italian DIALogue systems (IDIAL) Task

Francesco Cutugno[1], **Maria Di Maro**[1], **Sara Falcone**[2,3], **Marco Guerini**[2], **Bernardo Magnini**[2], **Antonio Origlia**[1]
[1] Università degli Studi di Napoli 'Federico II'
[2] Fondazione Bruno Kessler, Via Sommarive 18, Povo, Trento — Italy
[3] University of Trento, Italy.
{cutugno, maria.dimaro2, antonio.origlia}@unina.it
{sfalcone, guerini, magnini}@fbk.eu

Abstract

English. We report about the organization of the IDIAL (Evaluation of Italian DIALogue systems) task at EVALITA 2018, the first shared task aiming at assessing interactive characteristics of conversational agents for the Italian language. In this perspective, IDIAL considers a dialogue system as a "black box" (i.e., evaluation can not access internal components of the system), and measures the system performance on three dimensions: task completion, effectiveness of the dialogue and user satisfaction. We describe the IDIAL evaluation protocol, and show how it has been applied to the three participating systems. Finally, we briefly discuss current limitations and future improvements of the IDIAL methodology.

Italiano. *Riportiamo circa l'organizzazione del task IDIAL (Valutazione di sistemi di dialogo per l'italiano) a Evalita 2018. IDIAL é il primo task condiviso per la valutazione delle caratteristiche di interazione di agenti conversazionali per l'italiano. In questa prospettiva, IDIAL considera un sistema di dialogo come una "black box" (in quanto la valutazione non puó accedere ai componenti interni del sistema), e misura le prestazioni del sistema su tre dimensioni: la capacitá di portare a termine il task, l'efficacia del dialogo, e la soddisfazione dell'utente. Descriviamo il protocollo di valutazione IDIAL, e mostriamo come esso é stato applicato a tre sistemi partecipanti. Infine, discutiamo brevemente le limitazioni attuali e i miglioramenti futuri della metodologia.*

1 Task Motivations

The IDIAL (Evaluation of Italian DIALogue systems) task at EVALITA 2018 (Caselli et al., 2018) intends to develop and apply evaluation protocols for the quality assessment of existing task-oriented dialogue systems for the Italian language. Conversational Agents are one of the most impressive evidence of the recent resurgence of Artificial Intelligence. In fact, there is now a high expectation for a new generation of dialogue systems that can naturally interact and assist humans in several scenarios, including virtual coaches, personal assistants and automatic help desks. However, despite the growing commercial interest for various task-oriented conversational agents, there is still a general lack of methodologies for their evaluation. During the last years, the scientific community has studied the evaluation of dialogue systems under different perspectives, concerning for instance the appropriateness of the answer (Tao et al., 2017), or user satisfaction metrics (Hartikainen et al., 2004; Guerini et al., 2018).

IDIAL proposes an objective evaluation framework, which consists in both of user perception towards the ease of use and the utility of the system, and of consistency, robustness and correctness of the task-oriented conversational agent. The IDIAL starting point are previous evaluation models, comprising the observation of users and systems' behaviour, the judgment process inside the users, and the quality of the system regarding its service objectives (Möller and Ward, 2008).

2 IDIAL Evaluation Protocol

IDIAL assumes that the systems under assessment are task-oriented dialogue systems (TODSs), providing specific services in an application domain, such as hotel booking or technical support service. Systems can be either monomodal (spoken or written) or multimodal, and are used for com-

pleting a number of predefined tasks (or intents), each of which can be achieved in one or more interactions, or conversational turn pairs (i.e. sometimes a question-answer pair is not sufficient to accomplish the intended action, since other conversational turns are needed). TODS to be examined can be on-line or off-line applications, and are evaluated as "black-boxes", meaning that evaluators will not have access to the internal characteristics and components of the system. Given the peculiar nature of the evaluation, which is carried out by human users, IDIAL does not require neither training nor testing data. We target the evaluation of existing TODSs (both industrial and academic prototypes), which are on operation at the date of the test period (September 2018). The output of the evaluation is not a ranking. Conversely, we provide a qualitative assessment for each participating system, based on detailed and coherent set of technological and interactive characteristics of the system.

2.1 Evaluation Method

The IDIAL evaluation procedure is defined to address the following three characteristics of a task oriented-conversational agent:

A. Task completion. This is the capacity of the system to achieve the goals of the task for which the system has been designed, in a reasonable amount of time.

B. Effectiveness of the dialogue. This is the capacity of the system to interact with the user in order to achieve its task. It includes, among the others, the capacity to interpret commands accurately, the robustness of the system to unexpected input, the ease of use of the system, and the fluency of the dialogue.

C. User satisfaction. This is the reaction of the user after having used the system. It includes aspects like the degree of empathy of the system, the ability to read and respond to moods of human participant, the capacity of the system to give conversational cues, and the use appropriate degrees of formality.

The three characteristics (A-C) mentioned above are assessed in IDIAL by means of two evaluation methods, a questionnaire, and a set of linguistic stress tests.

Questionnaire. A questionnaire is given to the user after s/he has interacted with the system for a certain number of tasks. Questions may address each of the three main behaviours of the system (task completion, effectiveness of the dialogue and user satisfaction), and require the user to estimate the degree of acceptability (on a Likert scale), of a number of statements about the system. The questionnaire is prepared by experts, and it is intended to address questions both about Quality of Service and Quality of Experience. Whereas Quality of Service is about the accomplishment of the task, concerning the correct transferring of the needed information to the user, Quality of Experience consists of how the task was accomplished, if the user enjoyed the experience and would use the system again or recommend it (Moller et al., 2009).

Linguistic stress tests. A stress test is intended to assess the system behaviour under an unconventional interaction situation (i.e. a stressful situation), in order to evaluate the robustness of the system itself. In IDIAL 2018 we consider only linguistic stress tests, which are designed and applied by expert computational linguists. Stress tests are applied on real interactions through a substitution mechanism: given a user utterance in a dialogue, the utterance is minimally modified substituting some of the elements of the sentence, according to a pre-defined list of linguistic phenomena (e.g. typos, lexical choices, different kinds of syntactic structures, semantic reformulations, anaphora resolution). Other phenomena related to dialogue (e.g. requests of explanation, requests of interruption of the conversation) have not be considered in IDIAL 2018, and will be discussed for future editions. After application, a stress test is considered as "passed" if the behaviour of the system is not negatively affected by the substitution, otherwise the application is considered as "fail". The final score for a system is given by the ratio between the number of successfully applied stress tests over the total number of applied stress tests.

Table 1 summarize the system behaviours that are considered by the IDIAL evaluation, as well as the respective evaluation tools and their expected output.

2.2 Evaluation Procedure

Given a dialogue system to be evaluated, the evaluation phases described in Table 1 are practically applied according to the following steps:

1. Organizers prepare a user satisfaction ques-

System behaviour	Evaluation tool	Evaluation output
A. Task completion	Questionnaire	Summary report based on average scores on Likert scale
B. Effectiveness of the dialogue	Stress tests + Questionnaire	Summary report based on stress test success rate + summary report based on average scores on Likert scale
C. User satisfaction	Questionnaire	Summary report based on average scores on Likert scale

Table 1: Summary of IDIAL evaluation protocol.

tionnaire, which will be applied to all systems under evaluation (i.e. the questionnaire is not personalized). The questionnaire is reported in Section 4, and the Italian version is available as Appendix A of the IDIAL evaluation protocol.

2. Organizers write instructions on how to use the system on the base of a system submission (see Section 2.3). Typical instructions contain a task to be achieved by using the system (e.g. "book a train ticket for one person – you are free to decide destination and date").

3. Organizers individuate few users with average expertise for the system domain and task. For instance, if the system has been designed to serve a high school student, it seems reasonable to involve high school students as users.

4. Selected users interact with the system in order to achieve the goals defined at point 2. All interactions are recorded, and logs are made available. Depending of the task complexity, organizers decide how many runs will be experimented with the system. Overall, each user should not spend more than one hour with a system.

5. Just after the interaction, organizers provide each user with the questionnaire to be filled in. The same questionnaire is used for all participating systems.

6. Organizers select a sample of user interactions, and use them to design applicable stress test. The stress tests actually implemented in the evaluation are reported in Section 4.

7. Organizers run stress tests on user interactions and record system behaviour. In order to keep the experimental setting under control, only one stress test per interaction is applied.

8. For each system, organizers write the final evaluation summary report, on the base of both the questionnaire and the stress tests, according to the metrics reported in Table 1.

2.3 Submission Requirements

At submission time, IDIAL participants are asked to provide the following information concerning their system.

- Specify the tasks that the system can do, in the form of user intents (e.g. buy a train ticket, search for point of interest, take an appointment for a meeting, block credit card, etc.). More than one task for a system are allowed.

- Specify as much as possible the application domain of the system, in order to understand the knowledge which is managed during a dialogue (e.g. Italian railway stations, restaurants in Trento, meetings within one month, etc.).

- Interaction channel of the system (i.e. spoken, written, multimodal).

- System interface (e.g. messenger, telegram, twitter, proprietary interface, etc.).

- Access to the system (i.e. on-line, off-line, telephony service, etc.).

3 Participant Systems

Three different dialogue systems were tested according with the IDIAL evaluation protocol: a chatbot developed by the NLP research group at *Fondazione Bruno Kessler*[1], aiming at calculating the amount of carbohydrates in a meal; a vocal call-steering service developed by the Italian company *Interactive Media*[2], already in operation at a financial company, aiming at understanding the customer call and routing the call either to a human operator or to an automatic service; a spoken dialogue system developed by the Italian speech recognition company *Cedat 85*[3], aiming at supporting customers of a telco company in a number of services. The three systems, being the result of the research of groups with diverse background and application goals, were therefore very different in nature. Hence, this allowed us to test the *scalability* of the proposed protocol.

3.1 CH1 Conversational System for Diabetics

CH1 is a prototype (i.e. it is not yet operative) conversational agent capable of computing the grams of carbohydrates in a meal (Magnini et al., 2018). The chatbot is based on a written interaction in Italian, runs on Telegram and is designed to help diabetics who need to perform a "carbs count" for each consumed meal. The interaction is system-initiative, starting with a question posed by the bot concerning with the food eaten by the users during their last meal. The conversational exchange goes on with the list of food given by the user. In case the typed keywords do not exactly correspond to the vocabulary known by the system, the system provides a list of most similar dishes or ingredients to correctly compute the quantity of carbohydrates. The knowledge base used both to extract the similar food and to perform the carbohydrates computation is a domain ontology called Hellis, based on available nutritional scientific literature. The system makes use of machine learning approaches trained on a manually-annotated Italian corpus (DPD - Diabetic Patients Diary), containing diary entries written by diabetic patients.

3.2 Interactive Media Call-Steering System

Interactive Media submitted to IDIAL a virtual agent for receiving and routing calls to human or automatic service operators. The system is operative at the time of IDIAL evaluation. The spoken interaction takes place via the telephone channel and is user-initiative. As a matter of fact, after the initial user identification, through which the system asks name, surname and date of birth of users for their recognition, the human interlocutor is left free of asking open questions related to the field of application of the system itself, i.e. customer service for banks (for instance, *I want to get a loan*), call-steering for offices and companies, etc. Afterwards, the system can ask questions for disambiguation purposes (for instance, *Are you interested in a loan higher or lower than 3000?*), in order to properly classify the call in the correct category for the proper operator. The system has been developed within the IM-MIND platform integrated with Cisco CTI (*Computer Telephony Integration*)[4] and Nuance speech technologies[5].

3.3 Cedat 85: Speech Technologies in Action

The spoken dialogue system provided by Cedat 85 is a prototype (i.e. it is not yet operative) performing specific tasks suggested by the system at the beginning of the interaction (system-initiative) at the telephone. The following tasks can be addressed: i) informative operations concerning the final invoice, the list of transactions, the phone credit, and the tariff plan; ii) active operations such as loading your prepaid phone card (specifying the amount of money) and making a wire transfer (specifying the amount of money and the recipient). In case the request is not well understood, the system guides the user clarifying the possible actions to be performed. After that the user intent is understood and carried out, the user can continue with a new request or end the conversation.

4 Application of the IDIAL Evaluation protocol

The evaluation of the participating dialogue systems, as introduced in Section 2, was accomplished through two modules:

1. User experience, measured through a questionnaire;

2. Stress tests, mostly based on the log of the previous evaluation. The stress tests were evaluated using a pass/fail modality.

[1] https://ict.fbk.eu/units/nlp/
[2] https://www.imnet.com/it/
[3] http://www.cedat85.com/
[4] https://www.cisco.com/
[5] https://www.nuance.com/index.html

4.1 User Experience

We selected 10 different users for each system, who differed for age (range 19-60), sex and cultural background, for a total of 30 users. Each user had to interact with the system for three randomized tasks, for a total of 30 interactions for a system. After the completion of the tasks, each user was asked to fill in a questionnaire. It took in average 10 minutes to explain the tasks to be accomplished and make the users achieve them, and 5 minutes to fill the questionnaire, for a total of 15 minutes for each experiment.

The questionnaire was realized based on the current literature (Ives and Olson, 1984; Zviran and Erlich, 2003) and it considers the two main aspects that a task oriented conversational agent should cover, namely the Quality of Service and the Quality of Experience. The questionnaire used a Likert Scale (Graham et al., 2013) (*never, rarely, sometimes, often, always*) to evaluate each of the following questions:

1. The system was efficient in accomplishing the task.
2. The system quickly provided all the information that I needed.
3. The system is easy to use.
4. The system was incoherent when I interacted using a non-standard or unexpected input.
5. The system has a fluent dialogue.
6. The system was flexible to my needs.
7. I am satisfied by my experience.
8. I would recommend the system.
9. The system is charming.
10. I enjoyed the time that I spent using the system.

During the experiments we asked for feedback from the users, and what came out was that there should be more correlation between the tasks that we asked to users to accomplish, and the questions of the questionnaire. In addition, it would be interesting to add more questions in order to cover more aspects of the interaction that could be perceived and evaluated by the user. In order to do that, we should even better study the task that we ask the users to accomplish.

4.2 Linguistic Stress Tests

A stress test operates a substitution in a user utterance in order to test the behavior of the system in unconventional situations of interaction. Starting from the user interactions described in Section 4.1, we were able to collect the audio and textual logs of the interactions, which were in turn used to model our stress tests. We have analyzed and studied the logs of the conversations obtained by the users' test of each system. According to the literature (Danieli and Gerbino, 1995; Ruane et al., 2018), we proposed three categories of linguistics tests: spelling substitutions, lexical substitution and syntactic substitutions. In total we defined eleven tests, which were divided as follows:

- *Spelling substitutions*: it aims to test the system behavior when words are misspelled or confused, including cases of wrong speech recognition

  (ST-1) Confused Words (e.g. substitute "there" with "their", or "a fianco" and "affianco").

  (ST-2) Misspelled Words (e.g. substitute "accommodation" with "acommodation").

  (ST-3) Character Replacement (e.g. substitute "don't" with "dont", or "po'" with "po", or "lui dà" changed to "lui da").

  (ST-4) Character Swapping (e.g. substitute "casualmente" with "casuamlente" ,or "therefore" with "therefroe").

- *Lexical substitutions*: it aims to test the system behavior when a word is substituted with a less common word or with a more complex expression, preserving the meaning of the utterance

  (ST-5) Less frequent Synonyms (e.g. substitute "home" with "habitation").

  (ST-6) Synonyms specific to a register of speech or a geographical region (e.g. substitute "buongiorno, vorrei mangiare in un ristorante indiano" with the less formal "c'e un posto indiano").

  (ST-7) Coreference (e.g. substitute "Rome" with "the capital of Italy").

- *Syntactic substitutions*: it aims to test the system behavior when a less common grammatical structure of the utterance is used

(ST-8) Active-Passive Alternation (e.g. substitute “I would like to block the credit card” with the less common “I would like that the credit card is blocked”).

(ST-9) Inverted Order of Nouns and Adjectives (e.g. substitute “un piatto di pasta” with the less common “pasta un piatto”).

(ST-10) Anaphora Resolution (e.g. following the system question “did you say Rome or Milan?” substitute “Milan” with the less natural “the second”).

(ST-11) Verbal Modifier Inversion (e.g. substitute “I would like to buy a ticket to Milan for tomorrow” with the less used “I would like to buy a ticket for tomorrow to Milan”).

In order to apply as many tests as possible, among the ones listed above, even when it was not possible to use the log to obtain a suitable test, we created *ad hoc* tests. Before the application of the ad hoc tests, we checked whether the system was able to achieve the task in a non-stressful situation and, afterwards, we applied the stressful condition to our input. As far as spoken dialogue systems are concerned, it was not possible to apply the character replacement test, since it is a condition that can be tested only in a textual context.

5 Qualitative Analysis and Discussion

After having assessed the systems behaviour, we have set up an evaluation report for each of the system. The report includes the following sections: Evaluation summary, Detailed evaluation: questionnaire, and Detailed evaluation: stress test. We now briefly present the content of the three sections, using the CH1 system as example.

Evaluation summary. Here we give an high level statement about each of the three aspects reported in Table 1 (task completion, Effectiveness of the dialogue, user satisfaction). The statement reports the performance obtained on both the questionnaire and the stress tests. For instance, the following is the statement received by CH1 as far as the Effectiveness of the dialogue is concerned:

Effectiveness of the dialogue:

- Questions 3, 4, 7 and 10 of the questionnaire: average CH1 score is 2.03/4

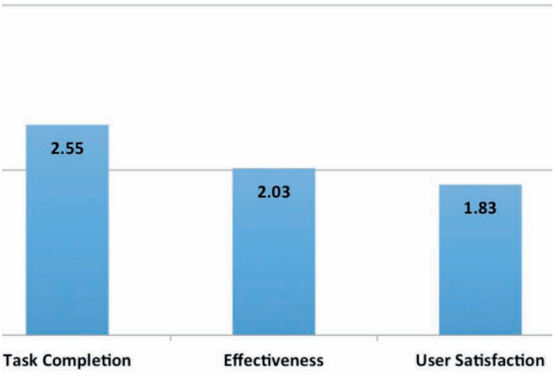


Figure 1: Example report of IDIAL questionnaire.

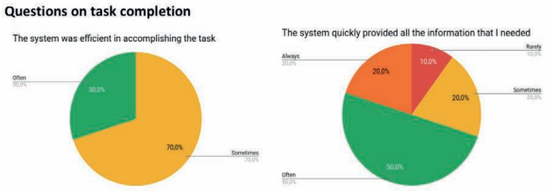


Figure 2: Example report of IDIAL questions on task completion.

- Linguistic stress tests: average CH1 score on the three groups (spelling substitutions, lexical substitution and syntactic substitutions) is 0.59/1.

Detailed evaluation: questionnaire. This section of the IDIAL evaluation reports about the questionnaire on the three aspects. Figure 1 shows the synthetic view on the three aspects of the system, while Figure 2 provides the diagrams reported for the two task completion questions.

Detailed evaluation: stress tests. This section of the IDIAL evaluation reports about the application of the linguistic stress tests. Figure 3 shows

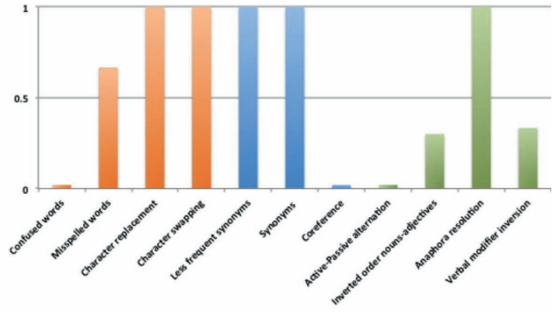


Figure 3: Example report of IDIAL linguistic stress tests.

the CH-1 performance on the eleven stress tests (average success rate (scale 0-1) on the eleven stress tests applied).

6 Post-Evaluation Questionnaire

One of the main aim of the IDIAL task at Evalita is the development of a scalable and domain independent methodology for assessing the performance of conversational agents. In this perspective, we were interested to know how the IDIAL evaluation protocol is perceived by the developers of the conversational agent participating in the task. As a first step in this direction, we submitted a post-evaluation questionnaire to the participants. The questionnaire comprised five questions, as follows:

- How do you judge the evaluation methodology used for IDIAL?
  - a) the user experience questionnaire?
  - b) the linguistic-oriented stress tests?
- How do you explain the successes or failure of the examined linguistic features?
- Are there aspects of your system which should be better considered in the IDIAL protocol?
- Which evaluation system do you normally use to test the functionality of your system? Which is its reference literature?
- Would you use the IDIAL evaluation protocol as the official metrics for evaluating your systems? Why? Eventually, after what kind of adjustments?

The double nature of the IDIAL protocol, which not only tests the user satisfaction but also proves the effectiveness of the system interactions in unconventional linguistic contexts of use, was generally perceived as a good choice for testing conversational agents. As a matter of fact, stress tests are judged to be a good starting point to improve the quality of linguistic performances for each system. Moreover, this kind of framework is seen to be particularly adequate to compare different systems accomplishing similar tasks.

On the other hand, participants stated that the results returned to them, although being graphically clear and understandable, were not fully satisfying, mainly as far as the variety of the tested interaction situations is concerned. This limitation was particularly relevant for systems showing many interaction capabilities and tasks (i.e. intents): for such systems poor performance on the tested tasks might not be representative of the overall behaviour of the system.

As a second feedback that we received, results in the evaluation report should be enriched with textual explanations, in order to better describe the reasons of failure.

Among other suggestions provided by the participants in the post-evaluation, we mention the need of including a comparison between the expected time of task completion and the actual time spent by users in accomplishing the requested service, the need of extending the number of intents to test, the ability of distinguishing the system ability to understand different entities within the same utterance, and the need of testing different system modules separately (i.e. ASR, TTS, SLU).

7 Conclusion

IDIAL (Evaluation of Italian DIALogue systems) is the first shared task aiming at assessing interactive characteristics of conversational agents for the Italian language. IDIAL considers a dialogue system as a "black box" (i.e., evaluation can not access internal components of the system), and measures the system performance on three dimensions: task completion, effectiveness of the dialogue and user satisfaction. The IDIAL evaluation protocol includes both a questionnaire with subjective user judgments, and a set of linguistic stress tests applied to interactions. The long term goal is the development of a scalable and domain independent methodology for assessing the performance of conversational agents.

Being the evaluated systems different with respect to the task they have been designed to address, the output of the IDIAL evaluation can not be a ranking. Conversely, for each system, we provide an evaluation report with a set of qualitative assessments based on a detailed and coherent set of interactive characteristics of the system. The method is flexible, since both the questions of the questionnaire and the stress tests can be adapted and personalized respecting the general principles of the methodology.

As for future improvements, there are few main aspects that need attention. First, the method should better test the variety of intents covered by

the system. The selection we made in our evaluation is not fully representative of the interaction situations of a complex system. Second, the relation between the time of task completion and the actual time spent by users in accomplishing the requested service is not considered in the current protocol. Finally, it would be interesting to deeply test the IDIAL protocol with dialogue systems with differ interaction modalities.

References

Tommaso Caselli, Nicole Novielli, Viviana Patti, and Paolo Rosso. 2018. Evalita 2018: Overview of the 6th evaluation campaign of natural language processing and speech tools for italian. In Tommaso Caselli, Nicole Novielli, Viviana Patti, and Paolo Rosso, editors, *Proceedings of Sixth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2018)*, Turin, Italy. CEUR.org.

Morena Danieli and Elisabetta Gerbino. 1995. Metrics for evaluating dialogue strategies in a spoken language system. In *Proceedings of the 1995 AAAI spring symposium on Empirical Methods in Discourse Interpretation and Generation*, volume 16, pages 34–39.

Yvette Graham, Timothy Baldwin, Alistair Moffat, and Justin Zobel. 2013. Continuous measurement scales in human evaluation of machine translation. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 33–41.

Marco Guerini, Sara Falcone, and Bernardo Magnini. 2018. A methodology for evaluating interaction strategies of task-oriented conversational agents. In *Proceedings of the 2018 EMNLP Workshop SCAI: The 2nd International Workshop on Search-Oriented Conversational AI*, pages 24–32.

Mikko Hartikainen, Esa-Pekka Salonen, and Markku Turunen. 2004. Subjective evaluation of spoken dialogue systems using ser vqual method. In *Eighth International Conference on Spoken Language Processing*.

Blake Ives and Margrethe H Olson. 1984. User involvement and mis success: A review of research. *Management science*, 30(5):586–603.

Bernardo Magnini, Vevake Balaraman, Mauro Dragoni, Marco Guerini, Simone Magnolini, and Valerio Piccioni. 2018. Ch1: A conversational system to calculate carbohydrates in a meal. In *Proceedings of the 17th International Conference of the Italian Association for Artificial Intelligence (AI*IA 2018)*.

Sebastian Möller and Nigel G Ward. 2008. A framework for model-based evaluation of spoken dialog systems. In *Proceedings of the 9th SIGdial Workshop on Discourse and Dialogue*, pages 182–189. Association for Computational Linguistics.

Sebastian Moller, Klaus-Peter Engelbrecht, Christine Kuhnel, Ina Wechsung, and Benjamin Weiss. 2009. A taxonomy of quality of service and quality of experience of multimodal human-machine interaction. In *Quality of Multimedia Experience, 2009. QoMEx 2009. International Workshop on*, pages 7–12. IEEE.

Elayne Ruane, Théo Faure, Ross Smith, Dan Bean, Julie Carson-Berndsen, and Anthony Ventresque. 2018. Botest: a framework to test the quality of conversational agents using divergent input examples. In *Proceedings of the 23rd International Conference on Intelligent User Interfaces Companion*, page 64. ACM.

Chongyang Tao, Lili Mou, Dongyan Zhao, and Rui Yan. 2017. Ruber: An unsupervised method for automatic evaluation of open-domain dialog systems. *arXiv preprint arXiv:1701.03079*.

Moshe Zviran and Zippy Erlich. 2003. Measuring is user satisfaction: review and implications. *Communications of the Association for Information Systems*, 12(1):5.

Overview of the Evalita 2018 Task on Automatic Misogyny Identification (AMI)

Elisabetta Fersini[1], **Debora Nozza**[1], **Paolo Rosso**[2]
[1]DISCo, Universitá degli Studi di Milano-Bicocca
[2]PRHLT Research Center, Universitat Politècnica de València
{fersini, debora.nozza}@disco.unimib.it
prosso@dsic.upv.es

Abstract

English. Automatic Misogyny Identification (AMI) is a new shared task proposed for the first time at the Evalita 2018 evaluation campaign. The AMI challenge, based on both Italian and English tweets, is distinguished into two subtasks, i.e. Subtask A on misogyny identification and Subtask B about misogynistic behaviour categorization and target classification. Regarding the Italian language, we have received a total of 13 runs for Subtask A and 11 runs for Subtask B. Concerning the English language, we received 26 submissions for Subtask A and 23 runs for Subtask B. The participating systems have been distinguished according to the language, counting 6 teams for Italian and 10 teams for English. We present here an overview of the AMI shared task, the datasets, the evaluation methodology, the results obtained by the participants and a discussion of the methodology adopted by the teams. Finally, we draw some conclusions and discuss future work.

Italiano. *Automatic Misogyny Identification (AMI) è un nuovo shared task proposto per la prima volta nella campagna di valutazione Evalita 2018. La sfida AMI, basata su tweet italiani e inglesi, si distingue in due sottotask ossia Subtask A relativo al riconoscimento della misoginia e Subtask B relativo alla categorizzazione di espressioni misogine e alla classificazione del soggetto target. Per quanto riguarda la lingua italiana, sono stati ricevuti un totale di 13 run per il Subtask A e 11 run per il Subtask B. Per quanto riguarda la lingua inglese, sono stati ricevuti 26 run per il Subtask A e 23 per Subtask B. I sistemi partecipanti sono stati distinti in base alla lingua, raccogliendo un totale di 6 team partecipanti per l'italiano e 10 team per l'inglese. Presentiamo di seguito una sintesi dello shared task AMI, i dataset, la metodologia di valutazione, i risultati ottenuti dai partecipanti e una discussione sulle metodologie adottate dai diversi team. Infine, vengono discusse conclusioni e delineati gli sviluppi futuri.*

1 Introduction

During the last years, the phenomenon of hate against women increased exponentially especially in online environment such as microblogs (Hewitt et al., 2016; Poland, 2016). According to the Pew Research Center Online Harassment report (2017) (Duggan, 2017), we can highlight that 41% of people were personally targeted, whose 18% were subjected to serious kinds of harassment because of the gender (8%) and that women are more likely to be targeted than men (11% vs 5%). Misogyny, defined as the hate or prejudice against women, can be linguistically manifested in numerous ways, ranging from less aggressive behaviours like social exclusion and discrimination to more dangerous expressions related to threats of violence and sexual objectification (Anzovino et al., 2018). Given this relevant social problem, the Automatic Misogyny Identification (AMI) task has been proposed first at IberEval 2018 (Spanish and English) (Fersini et al., 2018) and later at Evalita 2018 (Italian and English) (Caselli et al., 2018). The main goal of AMI is to distinguish misogynous contents from non-misogynous ones, to categorize misogynistic behaviours and finally to classify the target of a tweet.

Table 1: Examples of misogynous and non-misogynous tweets

Misogynous	**Text**
Misogynous	I've yet to come across a nice girl. They all end up being bit**es in the end.
Non-misogynous	@chiellini you are a bi*ch!

2 Task Description

The AMI shared task is organized according to two main subtasks:

- **Subtask A - Misogyny Identification**: a system must discriminate misogynistic contents from the non-misogynistic ones. Examples of misogynous and non-misogynous tweets are reported in Table 1.

- **Subtask B - Misogynistic Behaviour and Target Classification**: a system must recognize the targets that can be either specific users or groups of women together with the identification of the type of misogyny against women.

Regarding the misogynistic behaviour, a tweet must be classified as belonging to one of the following categories:

- *Stereotype & Objectification*: a widely held but fixed and oversimplified image or idea of a woman; description of women's physical appeal and/or comparisons to narrow standards.

- *Dominance*: to assert the superiority of men over women to highlight gender inequality.

- *Derailing*: to justify woman abuse, rejecting male responsibility; an attempt to disrupt the conversation in order to redirect women's conversations on something more comfortable for men.

- *Sexual Harassment & Threats of Violence*: to describe actions as sexual advances, requests for sexual favours, harassment of a sexual nature; intent to physically assert power over women through threats of violence.

- *Discredit*: slurring over women with no other larger intention.

Examples of Misogynistic Behaviours are reported in Table 2.

Concerning the target classification, the main goal is to classify each misogynous tweet as belonging to one of the following two target categories:

- *Active (individual)*: the text includes offensive messages purposely sent to a specific target;

- *Passive (generic)*: it refers to messages posted to many potential receivers (e.g. groups of women).

Examples of targets of misogynous tweets are reported in Table 3.

3 Training and Testing Data

In order to provide training and testing data both for Italian and English, three approaches were employed to collect misogynous text on Twitter:

- Streaming download using a set of manually defined representative keywords, e.g. *bi**h, w**re, c*nt* for English and *pu****a, tr**a, f**a di legno* for Italian;

- Monitoring of potential victims' accounts, e.g. *gamergate victims and public feminist women*;

- Downloading the history of identified misogynist, i.e. *explicitly declared hate against women on their Twitter profiles*.

Among all the collected tweets we selected a subset of text querying the database with the copresence of keywords, originating two corpora initially composed of 10000 tweets for each language. In order to label both the Italian and English datasets, we involved a group of 6 experts exploiting the CrowdFlower[1] platform for internal use. At the end of the labelling phase, we provided one corpus for Italian and one corpus for English to all the participants. The inter-rater annotator agreement on the English dataset for the fields of "misogynous", "misogyny_category" and "target"

[1]Now Figure Eight: `https://figure-eight.com/`

Table 2: Examples of tweets for each misogyny category

Misogyny Category	**Text**
Discredit	@AsiaArgento brainless fuc**ing bi**ch
Stereotype	Women play your role..#gobacktothekitchen
Objectification	Big te**ts and no brain..this is what I want as wife!
Sexual Harassment	I wanna touch your te**ts..you can't immagine what I can do on your body.
Threats of Violence	Stupid bi**ch I'll put you down on the floor and I'll rape you! You should be scared!
Dominance	Women are inferior to men..so shut up please!
Derailing	@yesallwomen wearing a tiny skirt is "asking for it". Your teasing a (hard working, taxes paying) dog with a bone. That's cruel. #YesAllMen

Table 3: Examples of targets.

Target	**Text**
Active	@JulieB stupid crazy psychopathic woman..you should die...
Passive	Women: just an inferior breed!!!

is 0.81, 0.45 and 0.49 respectively, while for the Italian dataset is 0.96, 0.68 and 0.76. Each corpus is distinguished in Training and Test datasets. Regarding the training data, both the Italian and English corpora are composed of 4000 tweets. Concerning the test data, we provided 1000 tweets for each language. The training data has been provided as tab-separated, according to the following fields:

- **id** denotes a unique identifier of the tweet.
- **text** represents the tweet text.
- **misogynous** defines if the tweet is misogynous or not misogynous; it takes values as 1 if the tweet is misogynous, 0 if the tweet is not misogynous.
- **misogyny_category** denotes the type of misogynistic behaviour; it takes value as:
  - *stereotype*: denotes the category "Stereotype & Objectification";
  - *dominance*: denotes the category "Dominance";
  - *derailing*: denotes the category "Derailing";
  - *sexual_harassment*: denotes the category "Sexual Harassment & Threats of Violence";
  - *discredit*: denotes the category "Discredit";
  - *0* if the tweet is not misogynous.
- **target** denotes the subject of the misogynous tweet; it takes value as:
  - *active*: denotes a specific target (individual);
  - *passive*: denotes potential receivers (generic);
  - *0* if the tweet is not misogynous.

Concerning the test data, only "id" and "text" have been provided to the participants. Examples of all possible allowed combinations are reported below. Additionally to the field "id", we report all the combinations of labels to be predicted, i.e. "misogynous", "misogyny_category" and "target":

```
0 0 0
1 stereotype active
1 stereotype passive
1 dominance active
1 dominance passive
1 derailing active
1 derailing passive
1 sexual_harassment active
1 sexual_harassment passive
1 discredit active
1 discredit passive
```

The label distribution related to the Training and Test datasets is reported in Table 4. While the distribution of labels related to the field "misogynous" is almost balanced (for both languages), the classes related to the other fields are quite unbalanced. Regarding the "misogyny_category", we

can distinguish between the two considered languages. In particular, for the Italian language, the most frequent label is related to the category *Stereotype & Objectification*, while for English the most predominant one is *Discredit*. Concerning the "target", the most predominant victims are specific users (*active*) with a strong imbalanced distribution on the Italian corpus, while it is almost balanced for the English training dataset and strongly imbalanced on the (*active*) targets for the corresponding test dataset.

4 Evaluation Measures and Baseline

Considering the distribution of labels of the dataset, we have chosen different evaluation metrics. In particular, we distinguished as follows:

Subtask A. Systems have been evaluated on the field "misogynous" using the standard accuracy measure, and ranked accordingly.

Subtask B. Each field to be predicted has been evaluated independently on the other using a Macro F1-score. In particular, the Macro F1-score for the "misogyny_category" field has been computed as average of F1-scores obtained for each category (stereotype, dominance, derailing, sexual_harassment, discredit), estimating $F_1(mi \quad \_c \quad)$. Analogously, the Macro F1-score for the "target" field has been computed as average of F1-scores obtained for each category (active, passive), $F_1(t \quad)$. The final ranking of the systems participating to Subtask B was based on the Average Macro F1-score (F_1), computed as follows:

$$F_1 = \frac{F_1(misogyny\_category)+F_1(target)}{2} \quad (1)$$

In order to compare the submitted runs with a baseline model, we provided a benchmark (AMI-BASELINE) based on Support Vector Machine trained on a unigram representation of tweets. In particular, we created one training set for each field to be predicted, i.e. "misogynous", "misogyny_category" and "target", where each tweet has been represented as a bag-of-words (composed of 1000 terms) coupled with the corresponding label. Once the representations have been obtained, Support Vector Machines with linear kernel have been trained, and provided as AMI-BASELINE.

5 Participants and Results

A total of 6 teams for Italian and 10 teams for English from 10 different countries participated in at least one of the two subtasks of AMI. Each team had the chance to submit up to three runs for English and three runs for Italian. Runs could be constrained, where only the provided training data and lexicons were admitted, and unconstrained, where additional data for training were allowed. Table 5 shows an overview of the teams[2] reporting their affiliation, their country, the number of submissions for each language and the subtasks they addressed.

5.1 Subtask A: Misogyny Identification

Table 6 reports the results for the Misogyny Identification task, which received 13 submissions for Italian and 26 runs for English submitted respectively from 6 and 10 teams. The highest Accuracy has been achieved by *bakarov* at 0.844 for Italian and by *hateminers* at 0.704 for English, both in a constrained setting. Most of the systems have shown an improvement with respect to the *AMI-BASELINE*. While the *bakarov* team submitted only one run based on TF-IDF coupled with Singular Value Decomposition and Boosting classifier, *hateminers* achieved the highest performance with a run based on vector representation that concatenates sentence embedding, TF-IDF and average word embeddings coupled with a Logistic Regression model.

5.2 Subtask B: Misogynistic Behaviour and Target Classification

Table 7 reports the results for the Misogynistic Behaviour and Target Classification task, which received 11 submissions by 5 teams for Italian and 23 submissions by 9 teams for English. The highest Average Macro F1-score has been achieved by *bakarov* at 0.501 for Italian (even if the amended run of CrotoneMilano achieved the highest effective performance) and by *himani* at 0.406 for English, both in a constrained setting. On the contrary of the previous task, most of the systems have shown lower performance compared to the *AMI-BASELINE*. It can be easily noted by looking at the Average Macro F1-score of all the approaches, that the problem of recognizing the misogyny category and the target is more difficult than the

[2]The teams *himani* and *resham* described their systems in the same report (Ahluwalia et al., 2018).

Table 4: Distribution of labels for "misogynous", "misogyny_category" and "target" on the Training and Test datasets. Percentages for "misogyny_category" and "target" are computed with respect to the number of misogynous tweets.

	Training		Testing	
	Italian	**English**	**Italian**	**English**
Misogynous	1828 (46%)	1785 (45%)	512 (51%)	460 (46%)
Non-misogynous	2172 (54%)	2215 (55%)	488 (49%)	540 (54%)
Discredit	634 (35%)	1014 (57%)	104 (20%)	141 (31%)
Sexual Harassment & Threats of Violence	431 (24%)	352 (20%)	170 (33%)	44 (10%)
Derailing	24 (1%)	92 (5%)	2 (1%)	11 (2%)
Stereotype & Objectification	668 (37%)	179 (10%)	175 (34%)	140 (30%)
Dominance	71 (3%)	148 (8%)	61 (12%)	124 (27%)
Active	1721 (94%)	1058 (59%)	446 (87%)	401 (87%)
Passive	107 (6%)	727 (41%)	66 (13%)	59 (13%)

Table 5: Team overview

Team Name	**Affiliation**	**Country**	**Runs**	**Subtask**
14-exlab (Pamungkas et al., 2018)	University of Turin Universitat Politècnica de València	IT ES	3 (EN), 3 (IT)	A, B
bakarov (Bakarov, 2018)	Huawei Technologies	RUS	3 (EN), 3 (IT)	A, B
CrotoneMilano (Basile and Rubagotti, 2018)	Symanto Research Independent Researcher	DE IT	1 (EN), 1 (IT)	A, B
hateminers (Saha et al., 2018)	Indian Institute of Technology	IND	3 (EN), 0 (IT)	A, B
himani (Ahluwalia et al., 2018)	University of Washington Tacoma	USA	3 (EN), 0 (IT)	A, B
ITT (Shushkevich and Cardiff, 2018)	Institute of Technology Tallaght Yandex	IRL RUS	3 (EN), 0 (IT)	A, B
RCLN (Buscaldi, 2018)	Université Paris 13	FR	1 (EN), 1 (IT)	A, B
resham (Ahluwalia et al., 2018)	University of Washington	USA	3 (EN), 0 (IT)	A, B
SB (Frenda et al., 2018b)	University of Turin Universitat Politècnica de València INAOE	IT ES MEX	3 (EN), 3 (IT)	A, B
StopPropagHate (Fortuna et al., 2018)	INESC TEC Eurecat Porto University	PT ES	3 (EN), 2 (IT)	A

misogyny identification task.

This is due to the fact that there can be a high overlapping between textual expressions of different misogyny categories, therefore it is highly subjective for an annotator (and consequently for a system) to select a category rather than another one. Regarding the target classification, systems can be easily misled by the presence of mentions that are not the target of the misogynous content.

While for the *bakarov* team the system for Subtask B is the same one of Subtask A, *himani* achieved the highest performance on the English language with a run based on a Bag of N-Gram representation coupled with an Ensemble of 5 models for classifying the Misogynistic Behaviour and 2 models for Target Classification.

6 Discussion

The submitted systems can be compared by taking into consideration the kind of input features that they have considered for representing tweets and the machine learning model that has been used as classification model.

Textual Feature Representation. The systems submitted by the challenge participants' consider various techniques for representing the tweet contents. Some teams have concentrated the effort on considering a single type of representation, i.e. the team *ITT* adopted the traditional TF-IDF representation, while *bakarov* and *RCLN* proposed systems considering only weighted n-grams at character level for better dealing with misspellings and capturing few stylistic aspects.

Additionally to the traditional textual feature representation techniques (i.e. bag of words/characters, n-grams of words/characters eventually weighted with TF-IDF) several teams proposed specific lexical features for improving the input space and consequently the classification performances. The team of *CrotoneMilano* experimented feature abstraction following the bleaching approach proposed by Goot et al. (Goot et al.,

Table 6: Results of Subtask A. Constrained runs are marked as *.c*, while the unconstrained ones with *.u*. After the deadline one team reported a format error. The resubmitted amended runs are marked with **.

ITALIAN			ENGLISH		
Rank	**Team**	**Accuracy**	**Rank**	**Team**	**Accuracy**
1	bakarov.c.run2	0.844	1	hateminers.c.run1	0.704
**	CrotoneMilano.c.run1	0.843	2	hateminers.c.run3	0.681
2	bakarov.c.run1	0.842	3	hateminers.c.run2	0.673
3	14-exlab.c.run3	0.839	4	resham.c.run3	0.651
4	bakarov.c.run3	0.836	5	bakarov.c.run3	0.649
5	14-exlab.c.run2	0.835	6	resham.c.run1	0.648
6	StopPropagHate.c.run1	0.835	7	resham.c.run2	0.647
7	AMI-BASELINE	0.830	8	ITT.c.run2	0.638
8	StopPropagHate.u.run2	0.829	9	ITT.c.run1	0.636
9	SB.c.run1	0.824	10	ITT.c.run3	0.636
10	RCLN.c.run1	0.824	11	himani.c.run2	0.628
11	SB.c.run3	0.823	12	bakarov.c.run2	0.628
12	SB.c.run2	0.822	13	14-exlab.c.run3	0.621
13	14-exlab.c.run1	0.765	14	himani.c.run1	0.619
			**	CrotoneMilano.c.run1	0.617
			15	himani.c.run3	0.614
			16	14-exlab.c.run1	0.614
			17	SB.c.run2	0.613
			18	AMI-BASELINE	0.605
			19	bakarov.c.run1	0.605
			20	StopPropagHate.c.run1	0.593
			21	SB.c.run1	0.592
			22	StopPropagHate.u.run3	0.591
			23	StopPropagHate.u.run2	0.590
			24	RCLN.c.run1	0.586
			25	SB.c.run3	0.584
			26	14-exlab.c.run2	0.500

2018) for modelling gender through the language. Specific lexicons for dealing with hate speech language have been included as features in the systems of *SB*, *resham* and *14-exlab*. In particular, *resham* and *14-exlab* made also use of environment-specific features, such as links, hashtags and emojis, and task-specific features, such as swear word, sexist slurs and women-related words.

Differently from these approaches, *StopPropagHate* and *hateminers* teams proposed systems that consider the popular Embeddings techniques both at word and sentence level.

Machine Learning Models. Concerning the machine learning models, we can distinguish between approaches that work with traditional Support Vector Machines and Logistic Regression, Ensemble Models and finally Deep Learning methods. Following, we report the models adopted by the systems that participated in the AMI shared task, according to the type of the machine learning model that has been adopted:

- Support Vector Machines have been exploited by *14-exlab* by using both linear and RBF kernel, by *SB* investigating only a radial basis function kernel, and by *CrotoneMilano* by adopting again a simple linear kernel;
- Logistic Regression has been used by *bakarov* and *hateminers*;
- Ensemble Models have been adopted by three teams according to different settings, i.e. *ITT* and *himani* used a Simple Voting of different classifiers, *resham* induced a Simple Voting over different input features and *RCLN* used an Ensemble based on Random Forest;
- A Deep Learning classifier has been adopted by only one team, i.e *StopPropagHate* that trained a simple dense neural network.

External Resources Several participants exploited external resources for providing task-specific lexical features.

The lexicons for addressing AMI for Italian have been mostly obtained from lists available online. The team *SB* used an available specific Italian

Table 7: Results of Subtask B. Constrained runs are marked as *.c*, while the unconstrained ones with *.u*. After the deadline one team reported a format error. The resubmitted amended runs are marked with **.

ITALIAN			ENGLISH		
Rank	**Team**	**Average Macro F1-score**	**Rank**	**Team**	**Average Macro F1-score**
**	CrotoneMilano.c.run1	0.501	1	himani.c.run3	0.406
1	bakarov.c.run1	0.493	2	himani.c.run2	0.377
2	AMI-BASELINE	0.487	3	AMI-BASELINE	0.370
3	14-exlab.c.run3	0.485	**	CrotoneMilano.c.run1	0.369
4	14-exlab.c.run2	0.482	4	hateminers.c.run3	0.369
5	bakarov.c.run3	0.478	5	hateminers.c.run1	0.348
6	bakarov.c.run2	0.463	6	SB.c.run2	0.344
7	SB.c.run3	0.449	7	himani.c.run1	0.342
8	SB.c.run1	0.448	8	SB.c.run1	0.335
9	RCLN.c.run1	0.448	9	hateminers.c.run2	0.329
10	SB.c.run2	0.446	10	SB.c.run3	0.328
11	14-exlab.c.run1	0.292	11	resham.c.run2	0.322
			12	resham.c.run1	0.316
			13	bakarov.c.run1	0.309
			14	resham.c.run3	0.283
			15	RCLN.c.run1	0.280
			16	ITT.c.run2	0.276
			17	bakarov.c.run2	0.275
			18	14-exlab.c.run1	0.260
			19	bakarov.c.run3	0.254
			20	14-exlab.c.run3	0.239
			21	ITT.c.run1	0.238
			22	ITT.c.run3	0.237
			23	14-exlab.c.run2	0.232

lexicon called "Le parole per ferire" built by Tullio De Mauro[3]. Starting from this lexicon provided by De Mauro, the HurtLex multilingual lexicon has been created (Bassignana et al., 2018). Beyond HurtLex, the team *14-exlab* gathered a swear word list from several sources[4] including a translated version of the *noswearing dictionary*[5] and a list of swear words from (Capuano, 2007).

Regarding the English language, both *resham* and *14-exlab* used the list of swear words from *noswearing dictionary* and the sexist slur list provided by (Fasoli et al., 2015). The team *resham* further investigated the sentiment polarity retrieved from SentiWordNet (Baccianella et al., 2010). Differently, the team *SB* exploited a manually modeled lexicon for the misogyny detection task proposed in (Frenda et al., 2018a). The HurtLex lexicon has been used by the team *14-exlab* also for the English task.

Finally, pre-trained Word Embeddings have been considered by *SB* and *hateminers* teams, specifically *GloVe* (Pennington et al., 2014) for the English task and Word Embeddings built on the TWITA corpus for the Italian one (Basile and Novielli, 2014).

7 Conclusions and Future Work

We presented here a new shared task about Automatic Misogyny Identification on Twitter for Italian and English. By analysing the runs submitted by the participants we can conclude that the problem of misogyny identification has been satisfactorily addressed by all the teams, while the misogynistic behaviour and target classification still remains a challenging problem. Concerning the future work, several issues should be considered to improve the quality of the collected data, especially for capturing those less frequent misogynistic behaviours such as Dominance and Derailing. The problem of hate speech against women will be further addressed in the HatEval shared task at SemEval in English and Spanish tweets[6].

[3] https://www.internazionale.it/opinione/tullio-de-mauro/2016/09/27/razzismo-parole-ferire

[4] https://www.parolacce.org/2016/12/20/dati-frequenza-turpiloquio/ and https://it.wikipedia.org/wiki/Turpiloquio_nella_lingua_italiana

[5] https://www.noswearing.com/dictionary

[6] SemEval 2019 Task 5: *HatEval: Multilingual Detection of Hate Speech Against Immigrants and Women in Twitter* https://competitions.codalab.org/competitions/19935

Acknowledgements

The work of the third author was partially funded by the SomEMBED TIN2015-71147-C2-1-P research project (MINECO/FEDER). We thank Maria Anzovino for her initial help in collecting the tweets subsequently used for the labelling phase and the final creation of the Italian and English corpora used for the AMI shared task.

References

Resham Ahluwalia, Himani Soni, Edward Callow, Anderson Nascimento, and Martine De Cock. 2018. Detecting Hate Speech Against Women in English Tweets. In *Proceedings of Sixth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2018)*, Turin, Italy. CEUR.org.

Maria Anzovino, Elisabetta Fersini, and Paolo Rosso. 2018. Automatic Identification and Classification of Misogynistic Language on Twitter. In *International Conference on Applications of Natural Language to Information Systems*, pages 57–64. Springer.

Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. 2010. Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *Proceedings of the International Conference on Language Resources and Evaluation*.

Amir Bakarov. 2018. Vector Space Models for Automatic Misogyny Identification. In *Proceedings of Sixth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2018)*, Turin, Italy. CEUR.org.

Pierpaolo Basile and Nicole Novielli. 2014. UNIBA at EVALITA 2014-SENTIPOLC Task: Predicting tweet sentiment polarity combining micro-blogging, lexicon and semantic features. In *Proceedings of Fourth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2014)*. CEUR.org.

Angelo Basile and Chiara Rubagotti. 2018. Automatic Identification of Misogyny in English and Italian Tweets at EVALITA 2018 with a Multilingual Hate Lexicon. In *Proceedings of Sixth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2018)*, Turin, Italy. CEUR.org.

Davide Buscaldi. 2018. Tweetaneuse AMI EVALITA2018: Character-based Models for the Automatic Misogyny Identification Task. In *Proceedings of Sixth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2018)*, Turin, Italy. CEUR.org.

R.G. Capuano. 2007. *Turpia: sociologia del turpiloquio e della bestemmia*. Riscontri (Milan, Italy). Costa & Nolan.

Tommaso Caselli, Nicole Novielli, Viviana Patti, and Paolo Rosso. 2018. EVALITA 2018: Overview of the 6th Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. In Tommaso Caselli, Nicole Novielli, Viviana Patti, and Paolo Rosso, editors, *Proceedings of Sixth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2018)*, Turin, Italy. CEUR.org.

Maeve Duggan. 2017. Online Harassment. `http://www.pewinternet.org/2017/07/11/online-harassment-2017/`. Last accessed 2018-10-28.

Fabio Fasoli, Andrea Carnaghi, and Maria Paola Paladino. 2015. Social acceptability of sexist derogatory and sexist objectifying slurs across contexts. *Language Sciences*, 52:98–107.

Elisabetta Fersini, M Anzovino, and P Rosso. 2018. Overview of the task on automatic misogyny identification at ibereval. In *Proceedings of the Third Workshop on Evaluation of Human Language Technologies for Iberian Languages (IberEval 2018). CEUR Workshop Proceedings. CEUR-WS. org, Seville, Spain*.

Paula Fortuna, Ilaria Bonavita, and Sérgio Nunes. 2018. INESC TEC, Eurecat and Porto University.

Simona Frenda, Ghanem Bilal, et al. 2018a. Exploration of Misogyny in Spanish and English tweets. In *Third Workshop on Evaluation of Human Language Technologies for Iberian Languages (IberEval 2018)*, volume 2150, pages 260–267. Ceur Workshop Proceedings.

Simona Frenda, Bilal Ghanem, Estefanía Guzmán-Falcón, Manuel Montes-y-Gómez, and Luis Villaseñor-Pineda. 2018b. Automatic Lexicons Expansion for Multilingual Misogyny Detection. In *Proceedings of Sixth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2018)*, Turin, Italy. CEUR.org.

Rob Goot, Nikola Ljubešić, Ian Matroos, Malvina Nissim, and Barbara Plank. 2018. Bleaching Text: Abstract Features for Cross-lingual Gender Prediction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 383–389.

Sarah Hewitt, Thanassis Tiropanis, and Christian Bokhove. 2016. The Problem of identifying Misogynist Language on Twitter (and other online social spaces). In *Proceedings of the 8th ACM Conference on Web Science*, pages 333–335. ACM.

Overview of the EVALITA 2018 Hate Speech Detection Task

Cristina Bosco
University of Torino
Italy
bosco@di.unito.it

Felice Dell'Orletta
ILC-CNR, Pisa
Italy
felice.dellorletta@ilc.cnr.it

Fabio Poletto
Acmos, Torino
Italy
fabio.poletto@edu.unito.it

Manuela Sanguinetti
University of Torino
Italy
msanguin@di.unito.it

Maurizio Tesconi
IIT-CNR, Pisa
Italy
maurizio.tesconi@iit.cnr.it

Abstract

English. The Hate Speech Detection (HaSpeeDe) task is a shared task on Italian social media (Facebook and Twitter) for the detection of hateful content, and it has been proposed for the first time at EVALITA 2018. Providing two datasets from two different online social platforms differently featured from the linguistic and communicative point of view, we organized the task in three tasks where systems must be trained and tested on the same resource or using one in training and the other in testing: HaSpeeDe-FB, HaSpeeDe-TW and Cross-HaSpeeDe (further subdivided into Cross-HaSpeeDe_FB and Cross-HaSpeeDe_TW sub-tasks). Overall, 9 teams participated in the task, and the best system achieved a macro F1-score of 0.8288 for HaSpeeDe-FB, 0.7993 for HaSpeeDe-TW, 0.6541 for Cross-HaSpeeDe_FB and 0.6985 for Cross-HaSpeeDe_TW. In this report, we describe the datasets released and the evaluation measures, and we discuss results.

Italiano. *HaSpeeDe è la prima campagna di valutazione di sistemi per l'identificazione automatica di discorsi di incitamento all'odio su social media (Facebook e Twitter) in lingua italiana, proposta nell'ambito di EVALITA 2018. Fornendo ai partecipanti due insiemi di dati estratti da due piattaforme differenti dal punto di vista linguistico e della comunicazione, abbiamo articolato HaSpeeDe in tre compiti in cui i sistemi sono addestrati e testati sulla stessa tipologia di dati oppure addrestrati su una tipologia e testati sull'altra: HaSpeeDe-FB, HaSpeeDe-TW e Cross-HaSpeeDe (a sua volta suddiviso in Cross-HaSpeeDe_FB e Cross-HaSpeeDe_TW). Nel complesso, 9 gruppi hanno partecipato alla campagna, e il miglior sistema ha ottenuto un punteggio di macro F1 pari a 0,8288 in HaSpeeDe-FB, 0,7993 in HaSpeeDe-TW, 0,6541 in Cross-HaSpeeDe_FB e 0.6985 in Cross-HaSpeeDe_TW. L'articolo descrive i dataset rilasciati e le modalità di valutazione, e discute i risultati ottenuti.*

1 Introduction and Motivations

Online hateful content, or Hate Speech (HS), is characterized by some key aspects (such as virality, or presumed anonymity) which distinguish it from offline communication and make it potentially more dangerous and hurtful. Therefore, its identification becomes a crucial mission in many fields.

The task that we have proposed for this edition of EVALITA namely consists in automatically annotating messages from two popular microblogging platforms, Twitter and Facebook, with a boolean value indicating the presence (or not) of HS.

HS can be defined as any expression "*that is abusive, insulting, intimidating, harassing, and/or incites to violence, hatred, or discrimination. It is directed against people on the basis of their race, ethnic origin, religion, gender, age, physical condition, disability, sexual orientation, political conviction, and so forth*" (Erjavec and Kovačič, 2012).

Although definitions and approaches to HS vary a lot and depend on the juridical tradition of the country, many agree that what is identified as

such can not fall under the protection granted by the right to freedom of expression, and must be prohibited. Also for transposing in practical initiatives the Code of Conduct of the European Union[1], online platforms like Twitter, Facebook or YouTube discourage hateful content, but its removal mainly relies on users and trusted flaggers reports, and lacks a systematic control.

Although HS analysis and identification requires a multidisciplinary approach that includes knowledge from different fields (psychology, law, social sciences, among others), NLP plays a fundamental role in this respect. Therefore, the development of high-accuracy automatic tools able to identify HS assumes the utmost relevance not only for NLP – and Italian NLP in particular – but also for all the practical applications a similar task lends itself to. Furthermore, as also suggested in Schmidt and Wiegand (2017), the community would considerably benefit from a benchmark dataset for HS detection underlying a commonly accepted definition of the task.

As regards the state of the art, a large number of contributions have been proposed on this topic, that adopt from lexicon-based (Gitari et al., 2015) to various machine learning approaches, and with different learning techniques, ranging from naïve Bayes classifiers (Kwok and Wang, 2013), Logistic Regression and Support Vector Machines (Burnap and Williams, 2015; Davidson et al., 2017), to the more recent Recurrent and Convolutional Neural Networks (Mehdad and Tetreault, 2016; Gambäck and Sikdar, 2017). However, there exist no comparative studies which would allow making judgement on the most effective learning method (Schmidt and Wiegand, 2017).

Furthermore, a large number of academic events and shared tasks took place in the recent past, thus reflecting the interest in HS and HS-related topics by the NLP community; to name a few, the first and second edition of the Workshop on Abusive Language[2] (Waseem et al., 2017), the First Workshop on Trolling, Aggression and Cyberbullying (Kumar et al., 2018), that also included a shared task on aggression identification, the tracks on Automatic Misogyny Identification (AMI) (Fersini et al., 2018b) and on authorship and aggressiveness analysis (MEX-A3T) (Carmona et al., 2018) proposed at the 2018 edition of IberEval, the GermEval Shared Task on the Identification of Offensive Language (Wiegand et al., 2018), the Automatic Misogyny Identification task at EVALITA 2018 (Fersini et al., 2018a), and finally the SemEval shared task on hate speech detection against immigrants and women (HatEval), that is still ongoing at the time of writing[3].

On the other hand, such contributions and events are mainly based on other languages (English, for most part), while very few of them deal with Italian (Del Vigna et al., 2017; Musto et al., 2016; Pelosi et al., 2017). Precisely for this reason, the Hate Speech Detection (HaSpeeDe)[4] task has been conceived and proposed within the EVALITA context (Caselli et al., 2018); its purpose is namely to encourage and promote the participation of several research groups, both from academia and industry, making a shared dataset available, in order to allow an advancement in the state of the art in this field for Italian as well.

2 Task Organization

Considering the linguistic, as well as meta-linguistic, features that distinguish Twitter and Facebook posts, namely due to the differences in use between the two platforms and the character limitations posed for their messages (especially on Twitter), the task has been further organized into three sub-tasks, based on the dataset used (see Section 3):

- **Task 1: HaSpeeDe-FB**, where only the Facebook dataset could be used to classify the Facebook test set
- **Task 2: HaSpeeDe-TW**, where only the Twitter dataset could be used to classify the Twitter test set
- **Task 3: Cross-HaSpeeDe**, which has been further subdivided into two sub-tasks:
  - **Task 3.1: Cross-HaSpeeDe_FB**, where only the Facebook dataset could be used to classify the Twitter test set
  - **Task 3.2: Cross-HaSpeeDe_TW**, where, conversely, only the Twitter

[1] On May 31, 2016, the EU Commission presented with Facebook, Microsoft, Twitter and YouTube a "Code of conduct on countering illegal hate speech online".

[2] https://sites.google.com/view/alw2018/

[3] https://competitions.codalab.org/competitions/19935

[4] http://www.di.unito.it/~tutreeb/haspeede-evalita18/

dataset could be used to classify the Facebook test set

Cross-HaSpeeDe, in particular, has been proposed as an out-of-domain task that specifically aimed on one hand at highlighting the challenging aspects of using social media data for classification purposes, and on the other at enhancing the systems' ability to generalize their predictions with different datasets.

3 Datasets and Format

The datasets proposed for this task are the result of a joint effort of two research groups on harmonizing the annotation previously applied to two different datasets, in order to allow their exploitation in the task.

The first dataset is a collection of Facebook posts developed by the group from Pisa and created in 2016 (Del Vigna et al., 2017), while the other one is a Twitter corpus developed in 2017-2018 by the Turin group (Sanguinetti et al., 2018). Section 3.1 and 3.2 briefly introduce the original datasets, while Section 3.3 describes the unified annotation scheme adopted in both corpora for the purposes of this task.

3.1 Facebook Dataset

This is a corpus of comments retrieved from the Facebook public pages of Italian newspapers, politicians, artists, and groups. Those pages were selected because typically they host discussions spanning across a variety of topics.

The comments collected were related to a series of web pages and groups, chosen as being suspected to possibly contain hateful content: *salviniofficial*, *matteorenziufficiale*, *lazanzarar24*, *jenusdinazareth*, *sinistracazzateliberta2*, *ilfattoquotidiano*, *emosocazzi*, *noiconsalviniufficiale*.

Overall, 17,567 Facebook comments were collected from 99 posts crawled from the selected pages. Five bachelor students were asked to annotate comments, in particular 3,685 received at least 3 annotations. The annotators were asked to assign one class to each post, where classes span over the following levels of hate: *No hate*, *Weak hate*, *Strong hate*.

Hateful messages were then divided into distinct categories: *Religion*, *Physical and/or mental handicap*, *Socio-economical status*, *Politics*, *Race*, *Sex and Gender issues*, and *Other*.

3.2 Twitter Dataset

The Twitter dataset released for the competition is a subset of a larger hate speech corpus developed at the Turin University. The corpus forms indeed part of the Hate Speech Monitoring program[5], coordinated by the Computer Science Department with the aim at detecting, analyzing and countering HS with an inter-disciplinary approach (Bosco et al., 2017). Its preliminary stage of development has been described in Poletto et al. (2017), while the fully developed corpus is described in Sanguinetti et al. (2018).

The collection includes Twitter posts gathered with a classical keyword-based approach, more specifically by filtering the corpus using neutral keywords related to three social groups deemed as potential HS targets in the Italian context: immigrants, Muslims and Roma.

After a first annotation step that resulted in a collection of around 1,800 tweets, the corpus has been further expanded by adding new annotated data. The newly introduced tweets were annotated partly by experts and partly by CrowdFlower (now Figure Eight) contributors. The final version of the corpus consists of 6,928 tweets.

The main feature of this corpus is its annotation scheme, specifically designed to properly encode the multiplicity of factors that can contribute to the definition of a hate speech notion, and to offer a broader tagset capable of better representing all those factors which may increase, or rather mitigate, the impact of the message. This resulted in a scheme that includes, besides HS tags (*no-yes*), also its intensity degree (from *1* through *4* if HS is present, and *0* otherwise), the presence of aggressiveness (*no-weak-strong*) and offensiveness (*no-weak-strong*), as well as irony and stereotype (*no-yes*).

In addition, given that irony has been included as annotation category in the scheme, part of this hate speech corpus (i.e. the tweets annotated as ironic) has also been used in another task proposed in this edition of EVALITA, namely the one on irony detection in Italian tweets (IronITA)[6](Cignarella et al., 2018). More precisely, the overlapping tweets in the IronITA datasets are 781 in the training set and just 96 in the test set.

[5] http://hatespeech.di.unito.it/

[6] http://www.di.unito.it/~tutreeb/ironita-evalita18/

3.3 Format and Data in HaSpeeDe

The annotation format provided for the task is the same for both datasets described above, and it consists of a simplified version of the schemes adopted in the two corpora introduced in Section 3.1 and 3.2.

The data have been encoded in UTF-8 plain-text files with three tab-separated columns, each one representing the following information:

1. the ID of the Facebook comment or tweet[7],
2. the text,
3. the class: `1` if the text **contains** HS, and `0` otherwise (see Table 1 and 2 for a few examples).

id	text	hs
8	*Io voterò NO NO E NO*	0
36	*Matteo serve un colpo di stato. Qua tra poco dovremo andare in giro tutti armati come in America.*	1

Table 1: Annotation examples from the Facebook dataset.

id	text	hs
1,783	*Corriere: Mafia Capitale, 4 patteggiamenti Gli appalti truccati dei campi rom*	0
3,290	*altro che profughi? sono zavorre e tutti uomini*	1

Table 2: Annotation examples from the Twitter dataset.

Both Facebook and Twitter datasets consist of a total amount of 4,000 comments/tweets retrieved from the main corpora introduced in Section 3.1 and 3.2. The data were randomly split into development and test set, of 3,000 and 1,000 messages respectively.

The distribution in both datasets of the labels expressing the presence or not of HS is summarized in Table 3 and 4.

	0	1
Train	1,618	1,382
Test	323	677
total	1,941	2,059

Table 3: Label distribution in the Facebook dataset.

	0	1
Train	2,028	972
Test	676	324
total	2,704	1,296

Table 4: Label distribution in the Twitter dataset.

4 Evaluation

Participants were allowed to submit up to 2 runs for each task, and a separate official ranking has been provided.

The evaluation has been performed according to the standard metrics known in literature, i.e Precision, Recall and F1-score. However, given the imbalanced distribution of hateful *vs* not hateful messages, and in order to get more useful insights on the system's performance on a given class, the scores have been computed for each class separately; finally the F1-score has been macro-averaged, so as to get the overall results.

For all tasks, the baseline score has been computed as the performance of a classifier based on the most frequent class.

5 Overview of the Task: Participation and Results

5.1 Task Participants and Submissions

A total amount of 9 teams[8] participated in at least one of the three HaSpeeDe main tasks. Table 5 provides an overview of the teams and their affiliation.

Except for one case, where one run was sent for HaSpeeDe-TW only, all teams submitted at least one run for *all* the tasks.

5.2 Systems

As participants were allowed to submit up to 2 runs for each task, several training options were adopted in order to properly classify the texts.

Furthermore, unlike other tasks, we have chosen to not establish any distinction between *constrained* and *unconstrained* runs, and to allow participants to use all the additional resources that

[7] In order to meet the GDPR requirements, texts have been pseudonymized replacing all original IDs in both datasets with newly-generated ones.

[8] In fact, 11 teams submitted their results, but one team withdrew its submissions, and another one's submissions have been removed from the official rankings by the task organizers.

Team	Affiliation
GRCP	Univ. Politècnica de València + CERPAMID, Cuba
InriaFBK	Univ. Côte d'Azur, CNRS, Inria + FBK, Trento
ItaliaNLP	ILC-CNR, Pisa + Univ. of Pisa
Perugia	Univ. for Foreigners of Perugia + Univ. of Perugia + Univ. of Florence
RuG	University of Groningen + Univ. degli Studi di Salerno
sbMMP	Zurich Univ. of Applied Sciences
StopPropagHate	INESC TEC + Univ. of Porto + Eurecat, Centre Tecn. de Catalunya
HanSEL	University of Bari Aldo Moro
VulpeculaTeam	University of Perugia

Table 5: Participants overview.

they deemed useful for the task (other annotated resources, lexicons, pre-trained word embeddings, etc.), on the sole condition that these were explicitly mentioned in their final report.

Table 6 summarizes the external resources (if any) used by participants to enhance their systems' performance, while the remainder of this section offers a brief overview of the teams' systems and core methods adopted to participate in the task .

GRCP (De la Peña Sarracén et al., 2018) The authors proposed a bidirectional Long Short-Term Memory Recurrent Neural Network with an Attention-based mechanism that allows to estimate the importance of each word; this context vector is then used with another LSTM model to estimate whether a text is hateful or not.

HanSEL (Polignano and Basile, 2018) The system proposed is based on an ensemble of three classification strategies, mediated by a majority vote algorithm: Support Vector Machine with RBF kernel, Random Forest and Deep Multilayer Perceptron. The input social media text is represented as a concatenation of word2vec sentence vectors and a TF-IDF bag of words.

InriaFBK (Corazza et al., 2018) The authors implemented three different classifier models, based on recurrent neural networks, n-gram based models and linear SVC.

ItaliaNLP (Cimino et al., 2018) Participants tested three different classification models: one based on linear SVM, another one based on a 1-layer BiLSTM and a newly-introduced one based on a 2-layer BiLSTM which exploits multi-task learning with additional data from the 2016 SENTIPOLC task (Barbieri et al., 2016).

Perugia (Santucci et al., 2018) The participants' system uses a document classifier based on a SVM algorithm. The features used by the system are a combination of features extracted using mathematical operations on FastText word embeddings and other 20 features extracted from the raw text.

RuG (Bai et al., 2018) The authors proposed two different classifiers: a SVM based on linear kernel algorithm and an ensemble system composed of a SVM classifier and a Convolutional Neural Network combined by a logistic regression meta-classifier. The features of each classifier is algorithm dependent and exploits word embeddings, raw text features and lexical resources features.

sbMMMP The authors tested two different systems, in a similar fashion to what described in von Grüningen et al. (2018). The first one is based on an ensemble of Convolutional Neural Networks (CNN), whose outputs are then used as features by a meta-classifier for the final prediction. The second system uses a combination of a CNN and a Gated Recurrent Unit (GRU) together with a transfer-learning approach based on pre-training with a large, automatically-translated dataset.

StopPropagHate (Fortuna et al., 2018) The authors use a classifier based on Recurrent Neural Networks with a binary cross-entropy as loss function. In their system, each input word is represented by a 10000-dimensional vector which is a one-hot encoding vector.

VulpeculaTeam (Bianchini et al., 2018) According to the description provided by participants, a neural network with three hidden layers was used, with word embeddings trained on a set of previously extracted Facebook comments.

5.3 Results and Discussion

In Table 7, 8, 9 and 10, we report the final results of HaSpeeDe, separated according to the respective sub-task and ranked by the macro F1-score (as described in Section 4)[9].

[9]Due to space constraints, the complete evaluation for all classes has been made available here: `https://goo.gl/xPyPRW`

Team	External Resources
GRCP	pre-trained word embeddings
InriaFBK	emotion lexicon
ItaliaNLP Lab	polarity and subjectivity lexicons + 2 word-embedding lexicons
Perugia	Twitter corpus + hate speech lexicon + polarity lexicon
RuG	pre-trained word embeddings + bad/offensive word lists
sbMMP	pre-trained word embeddings
StopPropagHate	–
HanSEL	pre-trained word embeddings
VulpeculaTeam	polarity lexicon + lists of bad words + pre-trained word embeddings

Table 6: Overview of the additional resources used by participants, besides the datasets provided by the task organizers.

In case of multiple runs, the suffixes "_1" and "_2" have been appended to each team name, in order to distinguish the run number of the submitted file. Furthermore, some of the runs in the tables have been marked with *: this means that they were re-submitted because of file incompatibility with the evaluation script or other minor issues that did not affect the evaluation process.

Team	Macro F1-score
baseline	0.2441
ItaliaNLP_2	0.8288
ItaliaNLP_1	0.8106
InriaFBK_1	0.8002
InriaFBK_2	0.7863
Perugia_2	0.7841
RuG_1	0.7751
HanSEL	0.7738
VulpeculaTeam*	0.7554
RuG_2	0.7428
GRCP_2	0.7147
GRCP_1	0.7144
StopPropagHate_2*	0.6532
StopPropagHate_1*	0.6419
Perugia_1	0.2424

Table 7: Results of the HaSpeeDe-FB task.

In absolute terms, i.e. based on the score of the first-ranked team, the best results have been achieved in the HaSpeeDe-FB task, with a macro F1 of 0.8288, followed by HaSpeeDe-TW (0.7993), Cross-HaSpeeDe_TW (0.6985) and Cross-HaSpeeDe_FB (0.6541).

The robustness of an approach benefiting from a polarity and subjectivity lexicon is confirmed by the fact that the best ranking team in both

Team	Macro F1-score
baseline	0.4033
ItaliaNLP_2	0.7993
ItaliaNLP_1	0.7982
RuG_1	0.7934
InriaFBK_2	0.7837
sbMMMP	0.7809
InriaFBK_1	0.78
VulpeculaTeam*	0.7783
Perugia_2	0.7744
RuG_2	0.753
StopPropagHate_2*	0.7426
StopPropagHate_1*	0.7203
GRCP_1	0.6638
GRCP_2	0.6567
HanSEL	0.6491
Perugia_1	0.4033

Table 8: Results of the HaSpeeDe-TW task.

HaSpeeDe-FB and HaSpeeDe-TW, i.e. ItaliaNLP, also achieved valuable results in the cross-domain sub-tasks, ranking at fifth and first position in Cross-HaSpeeDe_FB and Cross-HaSpeeDe_TW, respectively. But these results can also depend on the association of the polarity and subjectivity lexicon with word embeddings, which alone did not allow the achievement of particularly high results.

Furthermore, it is not surprising that the best results have been obtained on HaSpeeDe-FB, provided the fact that messages posted on this platform are longer and more correct than those in Twitter, allowing systems (and humans too) to find more and more clear indications of the presence of HS.

The coarse granularity of the annotation scheme,

Team	Macro F1-score
baseline	0.4033
InriaFBK_2	0.6541
InriaFBK_1	0.6531
VulpeculaTeam	0.6542
Perugia_2	0.6279
ItaliaNLP_1	0.6068
ItaliaNLP_2	0.5848
GRCP_2	0.5436
RuG_1	0.5409
RuG_2	0.4845
GRCP_1	0.4544
HanSEL	0.4502
StopPropagHate	0.443
Perugia_1	0.4033

Table 9: Results of the Cross-HaSpeeDe_FB sub-task.

Team	Macro F1-score
baseline	0.2441
ItaliaNLP_2	0.6985
InriaFBK_2	0.6802
ItaliaNLP_1	0.6693
InriaFBK_1	0.6547
VulpeculaTeam*	0.6189
RuG_1	0.6021
RuG_2	0.5545
HanSEL	0.4838
Perugia_2	0.4594
GRCP_1	0.4451
StopPropagHate*	0.4378
GRCP_2	0.318
Perugia_1	0.2441

Table 10: Results of the Cross-HaSpeeDe_TW sub-task.

which is a simplification of the schemes originally proposed for the datasets, and merged specifically for the purpose of this task, probably influenced the scores which are indeed very promising and high with respect to other tasks of the sentiment analysis area.

As regards the Cross-HaSpeeDe_FB and Cross-HaSpeeDe_TW sub-tasks, the lower results with respect to the in-domain tasks can be attributed to several factors, among which - and as expected - the different distribution in Facebook and Twitter datasets of HS and not HS classes. As a matter of fact, the percentage of HS in the Facebook train and test set is around 46% and 68%, respectively, while in the Twitter test set is around 32% in both sets. Such imbalanced distribution is reflected in the overall system outputs in the two sub-tasks: in Cross-HaSpeeDe_FB, where systems have been evaluated against the Twitter test set, most of the labels predicted as HS were not classified as such in the gold standard; conversely, in Cross-HaSpeeDe_TW, the majority of labels predicted as not HS were actually considered as HS in the gold corpus.
Another feature that distinguishes Facebook from Twitter dataset is the wider range of hate categories in the former, compared to the latter (see Section 3.1 and 3.2). Especially in Cross-HaSpeeDe_TW, the identification of hateful messsages may have been made even more difficult due to the reduced number of potential hate targets in the training set, with respect to the test set.

Overall, the heterogeneous nature of the datasets provided for the task - both in terms of class distribution and data composition - together with their quite small size, made the whole task even more challenging; nonetheless, this did not prevent participants from finding the appropriate solutions, thus improving the state of the art for HS identification in Italian language as well.

6 Closing Remarks

The paper describes the HaSpeeDe task for the detection of HS in Italian texts from Facebook and Twitter. The novelty of the task mainly consists in allowing the comparison between the results obtained on the two platforms and experiments on training on one typology of texts and testing on the other. The results confirmed the difficulty of cross-platform HS detection but also produced very promising scores in the tasks where the data from the same social network were exploited both for training and testing.
Future work can be devoted to an in-depth analysis of errors and to the observation of the contribution that different resources can give to systems performing this task.

Acknowledgments

The work of Cristina Bosco and Manuela Sanguinetti is partially funded by Progetto di Ateneo/CSP 2016 (*Immigrants, Hate and Prejudice in Social Media*, S1618_L2_BOSC_01).

References

Xiaoyu Bai, Flavio Merenda, Claudia Zaghi, Tommaso Caselli, and Malvina Nissim. 2018. RuG @ EVALITA 2018: Hate Speech Detection In Italian Social Media. In *Proceedings of Sixth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2018)*. CEUR.org.

Francesco Barbieri, Valerio Basile, Danilo Croce, Malvina Nissim, Nicole Novielli, and Viviana Patti. 2016. Overview of the Evalita 2016 SENTIment POLarity Classification Task. In *Proceedings of the Fifth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2016)*.

Giulio Bianchini, Lorenzo Ferri, and Tommaso Giorni. 2018. Text Analysis for Hate Speech Detection in Italian Messages on Twitter and Facebook. In *Proceedings of Sixth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2018)*. CEUR.org.

Cristina Bosco, Patti Viviana, Marcello Bogetti, Michelangelo Conoscenti, Giancarlo Ruffo, Rossano Schifanella, and Marco Stranisci. 2017. Tools and Resources for Detecting Hate and Prejudice Against Immigrants in Social Media. In *Proceedings of First Symposium on Social Interactions in Complex Intelligent Systems (SICIS), AISB Convention 2017, AI and Society*.

Pete Burnap and Matthew L. Williams. 2015. Cyber Hate Speech on Twitter: An Application of Machine Classification and Statistical Modeling for Policy and Decision Making. *Policy & Internet*, 7(2).

Miguel Ángel Álvarez Carmona, Estefanía Guzmán-Falcón, Manuel Montes-y-Gómez, Hugo Jair Escalante, Luis Villaseñor Pineda, Verónica Reyes-Meza, and Antonio Rico Sulayes. 2018. Overview of MEX-A3T at IberEval 2018: Authorship and Aggressiveness Analysis in Mexican Spanish Tweets. In *IberEval@SEPLN*. CEUR-WS.org.

Tommaso Caselli, Nicole Novielli, Viviana Patti, and Paolo Rosso. 2018. EVALITA 2018: Overview of the 6th Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. In *Proceedings of Sixth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2018)*. CEUR.org.

Alessandra Teresa Cignarella, Simona Frenda, Valerio Basile, Cristina Bosco, Viviana Patti, and Paolo Rosso. 2018. Overview of the Evalita 2018 Task on Irony Detection in Italian Tweets (IronITA). In *Proceedings of Sixth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2018)*. CEUR.org.

Andrea Cimino, Lorenzo De Mattei, and Felice Dell'Orletta. 2018. Multi-task Learning in Deep Neural Networks at EVALITA 2018. In *Proceedings of the 6th evaluation campaign of Natural Language Processing and Speech tools for Italian (EVALITA'18)*, Turin, Italy. CEUR.org.

Michele Corazza, Stefano Menini, Pinar Arslan, Rachele Sprugnoli, Elena Cabrio, Sara Tonelli, and Serena Villata. 2018. Comparing Different Supervised Approaches to Hate Speech Detection. In *Proceedings of Sixth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2018)*. CEUR.org.

Thomas Davidson, Dana Warmsley, Michael W. Macy, and Ingmar Weber. 2017. Automated Hate Speech Detection and the Problem of Offensive Language. *CoRR*, abs/1703.04009.

Gretel Liz De la Peña Sarracén, Reynaldo Gil Pons, Carlos Enrique Muñiz Cuza, and Paolo Rosso. 2018. Hate Speech Detection Using Attention-based LSTM. In *Proceedings of Sixth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2018)*. CEUR.org.

Fabio Del Vigna, Andrea Cimino, Felice Dell'Orletta, Marinella Petrocchi, and Maurizio Tesconi. 2017. Hate Me, Hate Me Not: Hate Speech Detection on Facebook. In *Proceedings of the First Italian Conference on Cybersecurity (ITASEC17)*.

Karmen Erjavec and Melita Poler Kovačič. 2012. "You Don't Understand, This is a New War!" Analysis of Hate Speech in News Web Sites' Comments. *Mass Communication and Society*, 15(6).

Elisabetta Fersini, Debora Nozza, and Paolo Rosso. 2018a. Overview of the EVALITA 2018 Task on Automatic Misogyny Identification (AMI). In *Proceedings of Sixth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2018)*. CEUR.org.

Elisabetta Fersini, Paolo Rosso, and Maria Anzovino. 2018b. Overview of the Task on Automatic Misogyny Identification at IberEval 2018. In *IberEval@SEPLN*. CEUR-WS.org.

Paula Fortuna, Ilaria Bonavita, and Sérgio Nunes. 2018. Merging datasets for hate speech classification in Italian. In *Proceedings of Sixth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2018)*. CEUR.org.

Björn Gambäck and Utpal Kumar Sikdar. 2017. Using Convolutional Neural Networks to Classify Hate-Speech. In *Proceedings of the First Workshop on Abusive Language*.

Njagi Dennis Gitari, Zhang Zuping, Hanyurwimfura Damien, and Jun Long. 2015. A lexicon-based approach for hate speech detection. *International Journal of Multimedia and Ubiquitous Engineering*, 10(4).

Overview of the EVALITA 2018 Solving language games (NLP4FUN) Task

Pierpaolo Basile and **Marco de Gemmis** and **Lucia Siciliani** and **Giovanni Semeraro**
Department of Computer Science
Via E. Orabona, 4 - 70125 Bari
University of Bari Aldo Moro
{firstname.lastname}@uniba.it

Abstract

English. This paper describes the first edition of the "Solving language games" (NLP4FUN) task at the EVALITA 2018 campaign. The task consists in designing an artificial player for *"The Guillotine" (La Ghigliottina, in Italian)*, a challenging language game which demands knowledge covering a broad range of topics. The game consists in finding a word which is semantically correlated with a set of 5 words called clues. Artificial players for that game can take advantage from the availability of open repositories on the web, such as Wikipedia, that provide the system with the cultural and linguistic background needed to find the solution.

Italiano. *Questo lavoro descrive la prima edizione del task "Solving language games" (NLP4FUN) task, proposto durante la campagna di valutazione EVALITA 2018. Il task consiste nella realizzazione di un giocatore artificiale per "La Gigliottina", un gioco linguistico molto sfidante, la cui soluzione richiede conoscenze in svariati campi. Il gioco consiste nel trovare una parola il cui significato è correlato a quello di un insieme di* 5 *parole, chiamate indizi. Un giocatore artificiale per questo task potrebbe sfruttare diverse sorgenti di conoscenza disponibili online, come Wikipedia, che forniscano al sistema le conoscenze linguistiche e culturali necessarie per arrivare alla soluzione.*

1 Motivation

Language games draw their challenge and excitement from the richness and ambiguity of natural language, and therefore have attracted the attention of researchers in the fields of Artificial Intelligence and Natural Language Processing. For instance, IBM Watson is a system which successfully challenged human champions of Jeopardy!, a game in which contestants are presented with clues in the form of answers, and must phrase their responses in the form of a question (Ferrucci et al., 2010; Molino et al., 2015). Another popular language game is solving crossword puzzles. The first experience reported in the literature is Proverb (Littman et al., 2002), that exploits large libraries of clues and solutions to past crossword puzzles. WebCrow is the first solver for Italian crosswords (Ernandes et al., 2008).

The proposed task consists in designing a solver for *"The Guillotine" (La Ghigliottina, in Italian)* game. It is inspired by the final game of an Italian TV show called "L'eredità". The game, broadcast by Italian National TV, involves a single player, who is given a set of five words - the clues - each linked in some way to a specific word that represents the unique solution of the game. Words are unrelated to each other, but each of them has a hidden association with the solution. Once the clues are given, the player has one minute to find the solution. For example, given the five clues: *sin*, *Newton*, *doctor*, *New York*, *bad*, the solution is *apple*, because: the apple is the symbol of original sin in Christian theology; Newton discovered the gravity by means of an apple; *"an apple a day keeps the doctor away"* is a famous proverb; New York city is also called *"the big apple"*; and *"one bad apple can spoil the whole bunch"* is a popular phrase which figuratively means that the person doing wrong can have a negative influence on those around him. "La Ghigliottina" is a challenging language game which demands knowledge covering a broad range of topics. Artificial players for that game can take advantage from the availability of open repositories on the web, such

as Wikipedia, that provide the system with the cultural and linguistic background needed to understand clues (Basile et al., 2016; Semeraro et al., 2009; Semeraro et al., 2012).

The task is part of EVALITA 2018, the periodic evaluation campaign of Natural Language Processing (NLP) and speech tools for the Italian language (Caselli et al., 2018).

The paper is organized as follows: Section 2 reports details about the task, the dataset and the evaluation protocol, while Section 3 describes the systems participating in the task, and Section 4 shows results.

2 Task Description: Dataset, Evaluation Protocol and Measures

An instance of the game consists of a set of 5 clue words and 1 word given as the official solution for that instance. We provided:

- a training set for the system development, containing 315 instances of the game;
- a test set for the evaluation, containing 105 instances of the game.

In order to measure the performance of the participants on games having different levels of difficulty, we provided instances taken both from the TV game and from the official board game. In the training set, 204 instances (64.8%) came from the TV game, 111 (35.2%) from the board game. In the test set, 66 instances (62.9%) were collected from the TV game, 39 (37.1%) from the board game. In order to discourage participants from cheating (e.g. finding the solution manually), in the test set we included 300 fake games automatically created by us. Obviously, fake games were not taken into account in the evaluation.

Any knowledge resource can be used to build an artificial player, except further instances of the game. For each instance of the game, a ranked list of maximum 100 tentative solutions must be provided.

2.1 Data Format

Both development and test set were provided in XML format:

```
<games>
  <game>
        <id>3fc953bd...</id>
        <clue>uomo</clue>
        <clue>cane</clue>
        <clue>musica</clue>
        <clue>casa</clue>
        <clue>pietra</clue>
        <solution>chiesa</solution>
        <type>TV</type>
  </game>
  ...
</games>
```

The XML file consists of a root element *games* which contains several *game* elements. Each game has five *clue* elements and one *solution*. Moreover, the element *type* specifies the type of the game: *TV* or *boardgame*.

The ranked list of solutions must be provided in a single plain text file, according to the following format:

```
id solution score rank time
```

Values were separated by a whitespace character; time taken by the system to compute the list was also reported in milliseconds. An example of a ranked list of solutions is reported below:

```
3fc953bd-... porta 0.978 1 3459
3fc953bd-... chiesa 0.932 2 3251
3fc953bd-... santo 0.897 3 4321
...
3fc953bd-... carta 0.321 100 2343
...
```

2.2 Evaluation

As evaluation measure, we adopt a weighted version of Mean Reciprocal Rank (MRR). Since time is a critical factor in this game, the Reciprocal Rank is weighted by a function which lowers the score based on the time taken by the computation. In fact, in the TV game, the player has only one minute to provide the solution. Taking into account these factors, the evaluation measure was:

$$\frac{1}{|G|}\sum_{g\in G}\frac{1}{r_g}max(\frac{1}{t_g},\frac{1}{10}) \quad (1)$$

where G is the set of games and r_g is the rank of the solution, while t_g denotes the minutes taken by the system to give the tentative solutions. Systems that took more than 10 minutes are equally penalized.

The evaluation was performed only on the 105 test games, for which we knew the correct solution (results provided for fake games were excluded).

We provided a separate ranking for TV and boardgame, but the final ranking was computed on the the whole test set.

3 Systems

Twelve teams registered in the task, but only two of them actually submitted the results for the evaluation. A short description of each system follows:

UNIOR4FUN - The system described in (Sangati et al., 2018) is based on the idea that clue words and corresponding solution are often part of a multiword expression. Therefore, the system exploits six linguistic patterns[1] that identify valid multiword expressions connecting clue and solution pairs. The core of the proposed solution is a set of freely available corpora and lexical resources built by the authors, which are used to find potential solutions by computing mutual information.

System by Luca Squadrone - In (Squadrone, 2018), the author proposed an algorithm based on two steps. In the first one, for each clue of a game, a list of relevant keywords is retrieved from linguistic corpora, so that each clue is associated with keywords representing the concepts having a relation with that clue. Then, words at the intersection of the retrieved sets are considered as candidate solutions. In the second step, another knowledge source made of proverbs, book and movie titles, word definitions, is exploited to count co-occurrences of clues and candidate solutions.

4 Results

Table 1: System results.

System	MRR	MRR (std)	Solved
UNIOR4NLP	0.6428	0.6428	81.90%
Squadrone	0.0134	0.0350	25.71%

Results of the evaluation in terms of MRR are reported in Table 1. The best performance is obtained by the *UNIOR4NLP* team. They reached a remarkable performance: MRR is very high, thus showing that the system is able to place the solution in the first positions of the ranking. We report, also, the standard MRR ($MRR(std)$) computed without taking into account the time. We notice that for UNIOR4NLP the value is equal to MRR: the system is able to provide the solution always in the first minute, while the Squadrone system takes more time for solving games.

Table 2 reports the results by game type (66 instances from the TV game and 39 instances from the boardgame). UNIOR4NLP shows similar results for both the game types, while the system proposed by Squadrone performs better on board games.

One possible explanation for this difference is that board games are meant just for fun; they are designed for the average player, whereas those taken from the TV game are more difficult to solve because they are intended to challenge the contestants of the show who try to win a money prize. Therefore, TV games generally have very specific clues and require more extensive knowledge about world facts and particular topics to find the solution than the average player has. As a consequence, the UNIOR4NLP solution based on specific multiword expressions extracted from several knowledge sources shows a more balanced performance than the other system.

However, despite the UNIOR4NLP system obtained remarkable results, very difficult games, requiring some kind of inference, are missed. For example, for the following clues: *uno, notte, la trippa, auto, palazzo*[2], the solution is *portiere* (porter). In order to solve that game, two difficult inferences are needed:

- *uno* is the number generally assigned to the role of the goolkeeper (portiere) in football teams;
- *"La Trippa"* is the surname of "Antonio La Trippa", a character of the Italian movie "Gli onorevoli", whose job is the porter (portiere) of a building.

We hope that in a further edition of this task participants will take into account these kind of games in which the simple co-occurrence of words it is not enough for solving the game. This is the most

[1] We must underline that patterns are extracted from a set of 100 games collected by authors. This is in contrast with the task guidelines; however, the games are not used for training the system.

[2] In English: one, night, "la trippa" (it was intended as a surname in this case), car, building

Table 2: System results for TV and boardgame

System	MRR (TV)	Solved (TV)	MRR (board)	Solved (board)
UNIOR4NLP	0.6528	86.36%	0.6001	71.79%
Squadrone	0.0068	25.75%	0.0245	25.64%

challenging aspect of this game. In order to compare system performance by taking into account the different levels of difficulty of the games, we plan to annotate guillottines with this information provided by human players. A deeper analysis of the results obtained by each system is provided in the corresponding technical reports (Sangati et al., 2018; Squadrone, 2018).

Finally, by looking at the statistics about the participation (12 registered teams, but only 2 of them submitted the results), we conclude that the task is attractive but perhaps it is too hard to solve. For further task editions, we plan to support the participants by providing pre-processed textual resources useful for solving the task.

5 Conclusions

Language games draw their challenge and excitement from the richness and ambiguity of natural language. This type of games are inconsistent with the closed world assumption: no fixed sets of rules are sufficient to define the game play. The proposed task consisted in building an artificial player for a challenging language game which requires from the player a strong linguistic and cultural background. The systems participating in the task were designed according to this idea: solving the game strongly depends on the background knowledge of the system. On the other hand, the results demonstrated that filling in the system with a solid background knowledge is not enough to find the solution, but strong NLP algorithms are required to discover hidden correlation among words. In fact, only the system based on specific linguistic patterns and multiword expressions was able to achieve high performance. Moreover, some games required a non-trivial inference step. For this kind of games, systems must be equipped with deeper reasoning capabilities. We hope that in further editions of the task, participants will propose solutions that deal with this issue.

References

Pierpaolo Basile, Marco de Gemmis, Pasquale Lops, and Giovanni Semeraro. 2016. Solving a complex language game by using knowledge-based word associations discovery. *IEEE Transactions on Computational Intelligence and AI in Games*, 8(1):13–26.

Tommaso Caselli, Nicole Novielli, Viviana Patti, and Paolo Rosso. 2018. Evalita 2018: Overview of the 6th evaluation campaign of natural language processing and speech tools for italian. In Tommaso Caselli, Nicole Novielli, Viviana Patti, and Paolo Rosso, editors, *Proceedings of Sixth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2018)*, Turin, Italy. CEUR.org.

Marco Ernandes, Giovanni Angelini, and Marco Gori. 2008. A web-based agent challenges human experts on crosswords. *AI Magazine*, 29(1):77.

David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya A Kalyanpur, Adam Lally, J William Murdock, Eric Nyberg, John Prager, et al. 2010. Building watson: An overview of the deepqa project. *AI magazine*, 31(3):59–79.

Michael L Littman, Greg A Keim, and Noam Shazeer. 2002. A probabilistic approach to solving crossword puzzles. *Artificial Intelligence*, 134(1-2):23–55.

Piero Molino, Pasquale Lops, Giovanni Semeraro, Marco de Gemmis, and Pierpaolo Basile. 2015. Playing with knowledge: A virtual player for who wants to be a millionaire? that leverages question answering techniques. *Artificial Intelligence*, 222:157–181.

Federico Sangati, Antonio Pascucci, and Johanna Monti. 2018. Exploiting Multiword Expressions to solve "La Ghigliottina". In Tommaso Caselli, Nicole Novielli, Viviana Patti, and Paolo Rosso, editors, *Proceedings of the 6th evaluation campaign of Natural Language Processing and Speech tools for Italian (EVALITA'18)*, Turin, Italy. CEUR.org.

Giovanni Semeraro, Pasquale Lops, Pierpaolo Basile, and Marco de Gemmis. 2009. On the Tip of My Thought: Playing the Guillotine Game. In Craig Boutilier, editor, *IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA, July 11-17, 2009*, pages 1543–1548. Morgan Kaufmann.

Giovanni Semeraro, Marco de Gemmis, Pasquale Lops, and Pierpaolo Basile. 2012. An artificial player for a language game. *IEEE Intelligent Systems*, 27(5):36–43.

Overview of the EVALITA 2018 Spoken Utterances Guiding Chef's Assistant Robots (SUGAR) Task

Maria Di Maro
Università degli Studi
di Napoli 'Federico II'
Department of Humanities
maria.dimaro2@unina.it

Antonio Origlia
Università degli Studi
di Napoli 'Federico II'
URBAN/ECO Research Center
antonio.origlia@unina.it

Francesco Cutugno
Università degli Studi
di Napoli 'Federico II'
Department of Electrical
Engineering and Information
Technology
cutugno@unina.it

Abstract

English. The SUGAR task is intended to develop a baseline to train a voice-controlled robotic agent to act as a cooking assistant. The starting point will be therefore to provide authentic spoken data collected in a simulated natural context from which semantic predicates will be extracted to classify the actions to perform. Three different approaches were used by the two SUGAR participants to solve the task. The enlightening results show the different elements of criticality underlying the task itself.

Abstract

Italiano. *Con il task SUGAR si intende sviluppare una baseline per addestrare un aiuto-cuoco robotico controllato da comandi vocali. Il punto di partenza sarà, pertanto, quello di fornire materiale vocale autentico raccolto in un contesto naturale simulato da cui saranno estratti i predicati semantici al fine di classificare le azioni da eseguire. Tre 16 diversi approcci sono stati utilizzati dai due partecipanti per risolvere il task. I risultati mostrano i veri livelli di criticità che soggiaciono il task stesso.*

1 Introduction

In the last few years, Human-Machine interaction systems have been in the spotlight, as far as computer science and linguistics are concerned, resulting in many applications such as Virtual Assistants and Conversational Agents (Cassell et al., 2000; Cauell et al., 2000; Dzikovska et al., 2003; Allen et al., 2007). The possibility to use such Artificial Intelligence technologies in domestic environments is increasingly becoming a reality (Darby, 2018; Ziefle and Valdez, 2017). In order to ensure the future possibility of making such systems even more intelligent, further researches are needed. As it has been the case with Apple SIRI and Google Assistant technologies, recent approaches transformed the former dialogue systems in direct action actuators, removing or reducing, as much as possible, clarification requests that may arise in presence of ambiguous commands. In this view, Spoken Language Understanding (SLU) is nowadays one of the major challenge of the field. Making a system able to truly understand the intention of the speaker in different contexts and react correctly, even in presence of Automatic Speech Recognition (ASR) errors, is the ultimate purpose to pursue in the field. In this context, the application of various semantic annotation schemata and criteria of knowledge modelling are of particular interest. Among different techniques used to model the interpretation process we cite: (i) *semantic-frame parsing*, where the frame classification with the recognition of its attribute can improve the information retrieval process for a more precise domain specific answer (Wang, 2010); (ii) *semantic interpretation*, for which semantic-syntactic trees can be used to extract basic semantic units and their relationships (Miller et al., 1996); (iii) *intent classification*, for which structures comprising generic predicates working as semantic primitives (Wierzbicka, 1972) and domain-dependent arguments can be used to represent a specific intent (Tur and Deng, 2011; Serban et al., 2018). With this particular task, we propose a possible framework for semantic classification to be tested, recurring to state-of-the-art SLU systems participating to the EVALITA-SUGAR challenge (Caselli et al., 2018).

2 Corpus Collection and Description

In the SUGAR challenge, the underlying task is to train a voice-controlled robotic agent to act as

Figure 1: 3D Recontruction of Bastian in his Kitchen. On the wall, the television showing frames of video recipes, from which users could extract actions to utter as commands

a cooking assistant. For this purpose, a training corpus of annotated spoken commands was collected. To collect the corpus, we designed a 3D virtual environment reconstructing and simulating a real kitchen where users could interact with a robot (named Bastian) which received commands to be performed in order to accomplish some recipes. User's orders were inspired by silent cooking videos shown in the 3D scene, thus ensuring the naturalness of the spoken production. Videos were segmented into elementary portions (frames) and sequentially proposed to the speakers who uttered a single sentence after each seen frame. In this view, speakers watched at video portions and then gave instructions to the robot to emulate what seen in the frame (Figure 1). The collected corpus then consists of a set of spoken commands, whose meaning derives from the various combination of actions, items (i.e. ingredients), tools and different modifiers.

Audio files were captured in a real acoustic environment, with a microphone posed at about 1 mt of distance from the speakers. The resulting corpus contains audio files for each speaker. These files were then segmented into sentences representing isolated commands. Orthographic transcriptions of the audio files were not be provided. Consequently, participants could use whichever ASR they prefer, whose performance was not under assessment. Nevertheless, the developed systems were expected to be strongly efficient despite the possible ASR deficiencies. Each resulting audio file was paired to a textual one containing the corresponding action annotation.

Training set Actions are represented as a finite set of generic predicates accepting an open set of parameters. For example, the action of *putting* may refer to a pot being placed on the fire

$$put(pot, fire)$$

or to an egg being put in a bowl

$$put(egg, bowl)$$

The annotation process resulted in determining the optimal action predicate corresponding to each command.

The training set consists of audio files and predicate description pairs, where the predicate serves as an interpretation of the intention to be performed by the robot. For these scenarios, the audio files are always mapped on a single interpretative predicate. The training set consists of 1721 utterances (and therefore 1721 audio files) produced by 36 different speakers annotated by two linguistic experts. The action templates, which have been inferentially defined through the video collection, are shown in Table 1, where [ ] indicates a list of ingredients, / the alternative among possible arguments, *quantity* and *modality* are not mandatory arguments, and * is used when the argument is recoverable from the context (i.e. previous instantiated arguments, which are not uttered, not even by means of clitics or other pronouns) or from the semantics of the verb. For instance,

friggere (fiori)[1]

is represented as

*aggiungere(fiori, *olio*)*[2]

because *olio* (En. *oil*) is implicitly expressed in the semantics of the verb *friggere* (En. *to fry*) as an instrument to accomplish the action. Among other phenomena, it is worth mentioning the presence of actions paired with templates, even when the syntactic structure needs a reconstruction, as in

coprire(ciotola, pellicola)[3]

which is annotated with the generic template as

mettere(pellicola, ciotola)[4].

[1] *fry(flowers)*
[2] *add(flowers, *oil*)*
[3] *cover(bowl, wrap)*
[4] *put(wrap, bowl)*

Predicate	Arguments
prendere	quantità, [ingredienti]/recipiente
aprire	quantità, [ingredienti], recipiente
mettere	quantità, utensile/[ingredienti], elettrodomestico, modalità
sbucciare	quantità, [ingredienti], utensile
schiacciare	[ingredienti, utensile
passare	[ingredienti], utensile
grattare	[ingredienti], utensile
girare	[ingredienti], utensile
togliere	utensile/prodotto, elettrodomestico
aggiungere	quantità, [ingredienti], utensile/recipiente/ elettrodomestico/[ingredienti], modalità
mescolare	[ingredienti], utensile, modalità
impastare	[ingredienti]
separare	parte/[ingredienti],ingrediente/utensile
coprire	recipiente/[ingredienti], strumento
scoprire	recipiente/[ingredienti]
controllare	temperatura, ingrediente
cuocere	quantità, [ingredienti], utensile, modalità

Table 1: Italian Action templates

In other cases, the uttered action represents the consequence of the action reported in the template, as in

separare(parte, fiori)[5]

and

pulire(fiori)[6],

or

mescolare([lievito, acqua])[7]

and

sciogliere(lievito, acqua)[8].

The argument order does not reflect the one in the audio files, but the following:

azione(quantità[9], oggetto, complemento, modalità)[10]

The modality arguments are of different types and the order is *adverb*, *cooking modality*, *temperature* and *time*.

Test set The test set consists of about 572 audio files containing uttered commands without annotations. Task participants were asked to provide, for each target command, the correct action predicate following the above-described format. Although single actions are of the same kind of the ones found in the training set and in the template file, the objects, on which such actions may be applied to, vary (i.e. different recipes, ingredients, tools...). Participants have been evaluated on the basis of correctly interpreted commands, represented in the form of predicates.

The task could be carried out either by using only the provided linguistic information of the training set or by means of other external linguistic tools, such as ontologies, specialised lexicons, and external reasoners.

3 Evaluation Protocol

The evaluation protocol covered the following possibilities:

- The proposed system correctly detects the requested action and all its parameters;
- The proposed system asks for repetition;
- The proposed system correctly detects the requested action but it assigns wrong parameters;
- The proposed system misses the action.

The possibility of asking for repetitions is left to participants to avoid forcing them to provide an answer in uncertain conditions. In this case, the evaluation protocol would assign a weaker penalisation than the one considered for missing the arguments or the action. The collected corpus did not, however, contain situations in which the system asks for repetitions.

The designed evaluation procedure outputted the following pieces of information:

1. an id comprising the listing number of the recognised predicate and the number of actions, in case of pluri-action predicates (1_1, 1_2, 2_1, etc);
2. a Boolean value (1: True, 0: False) indicating if the predicate has been recognised; when the predicates were not recognised, even the argument number is set on 0;
3. the number of expected arguments as indicated in the reference annotation files[11];

[5] *separate(part, flowers)*
[6] *clean(flowers)*
[7] *stir([yeast, water])*
[8] *melt(yeast, water)*
[9] The quantity always precedes the noun it is referred to. Therefore, it can also come before the complement
[10] action(quantity, object, complement, modality)
[11] The reference annotation files were annotation files created for the test set although not being made available

4. the distance between the participating systems' output file and the reference file computed by means of the Levenshtein distance (Levenshtein, 1966); the higher the computed distance in the output was, the more mistakes the system had detected;

5. the number of arguments for which the system asked for repetition.

Suppose the action in reference file is annotated as

1; [prendere(500 g, latte), aggiungere(latte, pentola)][12]

and the recognition procedure outputs

1; prendere(500 g, panna)[13]

instead of returning the following result, indicating a correct recognition

1_1 *(first predicate)*
(1, 2, 0, 0)

1_2 *(second predicate)*
(1, 2, 0, 0)

the evaluation outputs

1_1
(1, 2, 1, 0)

1_2
(0, 0, 0, 0)[14]

where the first predicate is recognised despite one mistaken argument, whereas the second predicate is not recognised at all.

The output format had to follow the one provided for the training data. For instance, asterisks indicating the implicitness of the arguments had to be included in the output file. As a matter of fact, retrieving the implicit function of a reconstructed argument serves to catch the degree of understanding of the system, along with making use of the processing of this information for the improvement of fine-grained action detection tasks. On the other hand, the choice between alternative arguments (separated by a slash in the reference files) do not invalidate the results. In fact, to execute an action, only one of the uttered alternatives must be chosen. Therefore, when one of the alternatives was recognised, the resulting output did not contain recognition errors. On the contrary, when the system reports both alternatives in the output file, the Levenshtein distance increased. In the reference files, alternatives were also occurring as implicit arguments, when an utterance can be completed by more than one possible argument.

[12] 1; [take(500 g, milk), add(milk, pot)]

[13] 1; take(500 g, cream)

[14] The first action was recognised; two arguments were expected but one of them was wrong. The second action was not recognised at all.

4 Participating Systems

In this section, we will report the results collected from testing the two participants' systems: the first (Section 4.1) have been developed at Fondazione Bruno Kessler (FBK), while the second by an Italian company which has decided to remain anonymous (Section 4.2). In table 2, results are summarised, showing that FBK had better performances in terms of correct predicate and arguments recognition for the intent classification, as far as the second system is concerned (Figure 2). On the other hand, the first one outputted worse results, despite the introduction of the argument repetition request. In this phase, the argument repetition percentage was not weighted in the accuracy rate of the system, which would have resulted in a slight increase of the accuracy itself, but we reported it as an additional performance of the participating system. For the anonymous system the action recognition is slightly beyond the 50%, but the argument recognition shows some issues (Figure 2) concerned with an over-fitting problem (see Section 4.2). For all three systems, recognition errors seemed to be random and not justifiable as semantically-related word selections.

4.1 FBK-HLT-NLP

To solve the proposed task, two different approaches were introduced. The first system was similar to the architecture proposed in (Madotto et al., 2018) and was based on an encoder-decoder approach. The encoder consisted of a MemNN network (Sukhbaatar et al., 2015) that stored each previous sentences in memory, from which relevant information was retrieved for the current sentence. The decoder was a combination of i) a MemNN to decode the input to an instruction containing tokens from output vocabulary and ii) a Pointer network (Vinyals et al., 2015) that chose which token from the input was to be copied to

	Correct Actions	Correct Arguments	Incorrect Actions	Incorrect Arguments	Argument Repetition
FBK System 1 [a]	50,16	28,31	49,83	71,68	**4,11**
FBK System 2	**66,36**	**46,22**	33,64	53,78	0
Anonymous System	53,89	17,46	46,11	82,54	0

[a] One user is missing.

Table 2: Percentages of accuracy and error rate for each tested system

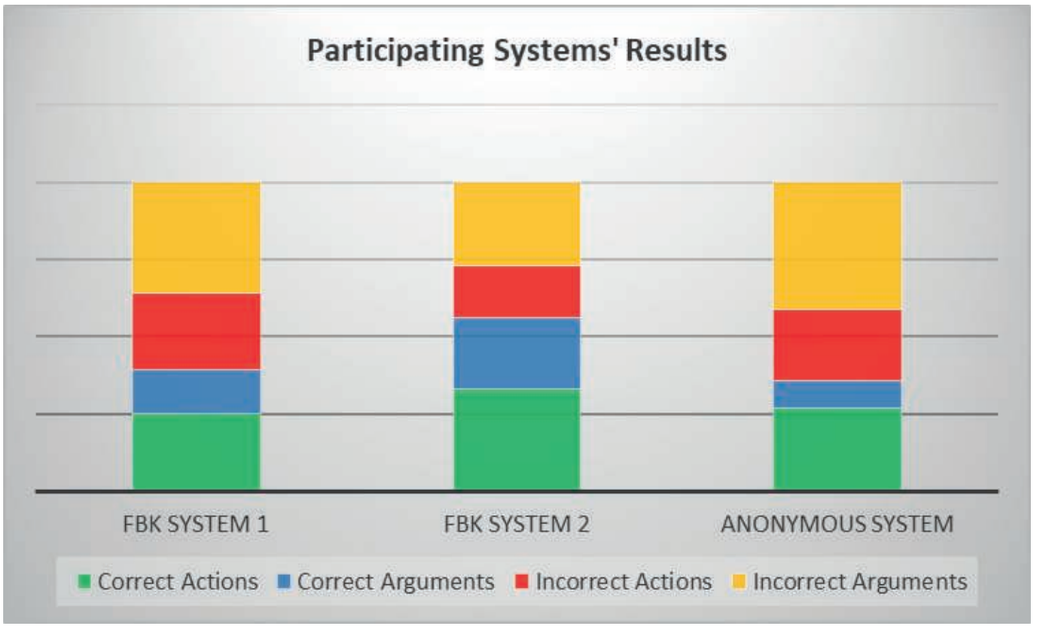


Figure 2: Results of the FBK first system

the output instruction. This system was used to classify the SUGAR corpus intents after an ASR transcription (System 1).

The second approach consisted of modeling the task as a *sequence to sequence* problem. Rather than implementing a new system, *Fairseq* (Gehring et al., 2017) - a fully convolutional architecture for sequence to sequence modeling - was used. Instead of relying on Recurrent Neural Networks (RNNs) to compute intermediate encoder states z and decoder states h convolutional neural networks (CNN) were adopted. Since the amount of training data was not big enough to train the model with such a system, written synthetic data were generated. To generate new data two main methodologies were adopted: on one hand random words were substituted with similar words based on similarity mechanisms, such as word embeddings; on the other hand, training sentences were generated by replacing verbs and names with synonyms extracted from an online vocabulary (System 2).

4.2 Deep neural network for SUGAR

The anonymous participant built a deep neural network system to tackle this task[15]. First of all, to convert the spoken utterances into text the Google Speech API was used. The neural network used a word embeddings lexicon trained on a corpus of recipes crawled on the web (4.5 million words) as features. The word embeddings, with vectors having 100 dimensions, were trained with the skip-gram algorithm of fastText[16] (Bojanowski et al., 2016).

As a preliminary step an autoencoder to embed the predicates in a vector was built. The encoder was made of a two Bi-LSTM layers. The first one was in charge of processing the token sequences for each predicate. The second layer processed the sequence of predicates and embeds them into a vector called predicates embedding. This vector was then split into n-parts where n was the

[15]The following report is a result of a conversation with the involved participant, whose report was not officially submitted to EVALITA 2018 in order to remain anonymous.

[16]https://fasttext.cc/

maximum number of predicates. The decoder was made of two Bi-LSTM layers, where the first layer was in charge of decoding the sequence of predicates and the second layer was in charge of decoding the sequence of token for each predicate. To test the autoencoder, a development test set was extracted from the training test. The autoencoder was able to encode and decode with no changes the 96.73% of the predicates in the development test set.

The different possible actions have been represented as classes in a hot-encode vector, and for each action a binary flag has been used to represent whether the action was implicit or not. The predicates have been encoded into a vector, using the aforementioned encoder, and for each predicate a flag was used to represent their alleged implicitness.

A multitask neural network was used to classify the actions, to detect whether they were implicit and to predict the predicates. The network took in input a recipe as a list of commands, each of whom was encoded by a Bi-LSTM layer. A second Bi-LSTM layer processed the command sequence and outputted a list of command embeddings. Each embeddings was split into n-parts which identified the actions included in the command. Each of these actions was passed to 4 dense layers that predicted the action class, the implicitness of the action, and the predicates embedding. Finally, the above-described decoder translated the predicates embedding into actual predicates.

5 Conclusions

With this task we proposed a field of application for spoken language understanding research concerned with intents classification of a domain-dependent system using a limited amount of training data. The results show that further analysis should be carried out to solve such semantic recognition problems, starting with an analysis of the errors occurred in the participating systems, an enlargement of the reference corpus, up to finding a suitable pipeline for data processing, including a rule-based module to model issues such as the argument implicitness, both in anaphoric- or semantic-dependent situations. This task is therefore intended to be a first reflection, whose next developments would include the creation of a corpus for the English language and the introduction of multimodality. As a matter of fact, pointing gestures or mimed actions and movements, on the basis of which the interlocutor should be capable of re-performing them with actual tools and ingredients, are multimodal activities that are of interest for this field of application as for any other spoken understanding task where a shared context of interaction is expected.

Acknowledgments

We thank the EVALITA 2018 organisers and the SUGAR participants for the interest expressed. A special thank also goes to Claudia Tortora, who helped us collect recipes and annotate our training set, and, last but not least, to the numerous testers who had fun talking with our dear Bastian.

This work is funded by the Italian PRIN project *Cultural Heritage Resources Orienting Multimodal Experience* (CHROME) #B52F15000450001.

References

James Allen, Mehdi Manshadi, Myroslava Dzikovska, and Mary Swift. 2007. Deep linguistic processing for spoken dialogue systems. In *Proceedings of the Workshop on Deep Linguistic Processing*, pages 49–56. Association for Computational Linguistics.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Tommaso Caselli, Nicole Novielli, Viviana Patti, and Paolo Rosso. 2018. Evalita 2018: Overview of the 6th evaluation campaign of natural language processing and speech tools for italian. In Tommaso Caselli, Nicole Novielli, Viviana Patti, and Paolo Rosso, editors, *Proceedings of Sixth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2018)*, Turin, Italy. CEUR.org.

Justine Cassell, Joseph Sullivan, Elizabeth Churchill, and Scott Prevost. 2000. *Embodied conversational agents*. MIT press.

Justine Cauell, Tim Bickmore, Lee Campbell, and Hannes Vilhjálmsson. 2000. Designing embodied conversational agents. *Embodied conversational agents*, pages 29–63.

Sarah J Darby. 2018. Smart technology in the home: time for more clarity. *Building Research & Information*, 46(1):140–147.

Myroslava O Dzikovska, James F Allen, and Mary D Swift. 2003. Integrating linguistic and domain knowledge for spoken dialogue systems in multiple

domains. In *Proc. of IJCAI-03 Workshop on Knowledge and Reasoning in Practical Dialogue Systems*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. *arXiv preprint arXiv:1705.03122*.

Vladimir I Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710.

Andrea Madotto, Chien-Sheng Wu, and Pascale Fung. 2018. Mem2seq: Effectively incorporating knowledge bases into end-to-end task-oriented dialog systems. *arXiv preprint arXiv:1804.08217*.

Scott Miller, David Stallard, Robert Bobrow, and Richard Schwartz. 1996. A fully statistical approach to natural language interfaces. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, pages 55–61. Association for Computational Linguistics.

Iulian Vlad Serban, Ryan Lowe, Peter Henderson, Laurent Charlin, and Joelle Pineau. 2018. A survey of available corpora for building data-driven dialogue systems: The journal version. *Dialogue & Discourse*, 9(1):1–49.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in neural information processing systems*, pages 2440–2448.

Gokhan Tur and Li Deng. 2011. Intent determination and spoken utterance classification. *Spoken Language Understanding: Systems for Extracting Semantic Information from Speech*, pages 93–118.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems*, pages 2692–2700.

Ye-Yi Wang. 2010. Strategies for statistical spoken language understanding with small amount of data-an empirical study. In *Eleventh Annual Conference of the International Speech Communication Association*.

Anna Wierzbicka. 1972. Semantic primitives.

Martina Ziefle and André Calero Valdez. 2017. Domestic robots for homecare: A technology acceptance perspective. In *International Conference on Human Aspects of IT for the Aged Population*, pages 57–74. Springer.

Multi-task Learning in Deep Neural Networks at EVALITA 2018

Andrea Cimino[⋆], **Lorenzo De Mattei**[⋆,◇] **and Felice Dell'Orletta**[⋆]
[⋆] Istituto di Linguistica Computazionale "Antonio Zampolli" (ILC–CNR)
ItaliaNLP Lab - *www.italianlp.it*
[◇] Dipartimento di Informatica, Università di Pisa
{andrea.cimino,felice.dellorletta}@ilc.cnr.it
lorenzo.demattei@di.unipi.it

Abstract

English. In this paper we describe the system used for the participation to the ABSITA, GxG, HaSpeeDe and IronITA shared tasks of the EVALITA 2018 conference. We developed a classifier that can be configured to use Bidirectional Long Short Term Memories and linear Support Vector Machines as learning algorithms. When using Bi-LSTMs we tested a multi-task learning approach which learns the optimized parameters of the network exploiting simultaneously all the annotated dataset labels and a multiclassifier voting approach based on a k-fold technique. In addition, we developed generic and specific word embedding lexicons to further improve classification performances. When evaluated on the official test sets, our system ranked 1st in almost all sub-tasks for each shared task, showing the effectiveness of our approach.

Italiano. *In questo articolo descriviamo il sistema utilizzato per la partecipazione agli shared task ABSITA, GxG, HaSpeeDe ed IronITA della conferenza EVALITA 2018. Abbiamo sviluppato un sistema che utilizza come algoritmi di apprendimento sia reti di tipo Long Short Term Memory Bidirezionali (Bi-LSTM) che Support Vector Machines. Nell'utilizzo delle Bi-LSTM abbiamo testato un approccio di tipo multi task learning nel quale i parametri della rete vengono ottimizzati utilizzando contemporaneamente le annotazioni presenti nel dataset ed una strategia di classificazione a voti di tipo k-fold. Abbiamo creato word embeddings generici e specifici per ogni singolo task per migliorare ulteriormente le performance di classificazione. Il nostro sistema quando valutato sui test set ufficiali ha ottenuto il primo posto in quasi tutti i sotto task di ogni shared task affrontato, dimostrando la validità del nostro approccio.*

1 Description of the System

The EVALITA 2018 edition has been one of the most successful editions in terms of number of shared tasks proposed. In particular, a large part of the tasks proposed by the organizers can be tackled as binary document classification tasks. This gave us the possibility to test a new system specifically designed for this EVALITA edition.

We implemented a system which relies on Bi-LSTM (Hochreiter et al., 1997) and SVM which are widely used learning algorithms in the document classification task. The learning algorithm can be selected in a configuration file. In this work we used the Keras (Chollet, 2016) library and the liblinear (Fan et al., 2008) library to generate the Bi-LSTM and SVM statistical models respectively. Since our approach relies on morphosyntactically tagged text, training and test data were automatically morphosyntactically tagged by the PoS tagger described in (Cimino and Dell'Orletta, 2016). Due to the label constraints in the dataset, if our system classified an aspect as not present, we forced the related positive and negative labels to be classified as not positive and not negative. We developed sentiment polarity and word embedding lexicons with the aim of improving the overall accuracy of our system.

Some specific adaptions were made due to the characteristics of each shared task. In the Aspect-based Sentiment Analysis (ABSITA) 2018 shared task (Basile et al., 2018) participants were asked, given a training set of Booking hotel reviews, to detect the mentioned aspect categories in a review among a set of 8 fixed categories (ACD task) and

to assign the polarity (neutral, positive, neutral, positive-negative) for each detected aspect (ACP task). Since each Booking review in the training set is labeled with 24 binary labels (8 indicating the presence of an aspect, 8 indicating positivity and 8 indicating negativity w.r.t. an aspect), we addressed the ABISTA 2018 shared task as 24 binary classification problems.

The Gender X-Genre (GxG) 2018 shared task (Dell'Orletta and Nissim, 2018) consisted in the automatic identification of the gender of the author of a text (Female or Male). Five different training sets and test sets were provided by the organizers for five different genres: Children essays (CH), Diary (DI), Journalism (JO), Twitter posts (TW) and YouTube comments (YT). For each test set the participants are requested to submit a system trained using in-domain training dataset and a system trained using cross-domain data only.

The IronITA task (Cignarella et al., 2018) consisted of two tasks. In the first task participants had to automatically label a message as ironic or not. The second task had a more fine grain: given a message, participants had to classify whether the message is sarcastic, ironic but not sarcastic or not ironic.

Finally in the HaSpeeDe 2018 shared task (Bosco et al., 2018) consisted in automatically annotating messages from Twitter and Facebook with a boolean value indicating the presence (or not) of hate speech. In particular three tasks were proposed: **HaSpeeDe-FB** where only the Facebook dataset could be used to classify Facebook comments, **HaSpeeDe-TW** where just Twitter data could be used to classify tweets and **Cross-HaspeeDe** where only the Facebook dataset could be used to classify the Twitter test set and vice versa (**Cross-HaspeeDe_FB**, **Cross-HaspeeDe_TW**).

1.1 Lexical Resources

1.1.1 Automatically Generated Sentiment Polarity Lexicons for Social Media

For the purpose of modeling the word usage in generic, positive and negative contexts of social media texts, we developed three lexicons which we named $TW_{GEN}, TW_{NEG}, TW_{POS}$. Each lexicon reports the relative frequency of a word in three different corpora. The main idea behind building these lexicons is that positive and negative words should present a higher relative frequency in TW_{POS} and TW_{NEG} respectively. The three corpora were generated by first downloading approximately 50,000,000 tweets and then applying some filtering rules to the downloaded tweets to build the positive and negative corpora (no filtering rules were applied to build the generic corpus). In order to build a corpus of positive tweets, we constrained the downloaded tweets to contain at least one positive emoji among heart and kisses. Since emojis are rarely used in negative tweets, to build the negative tweets corpus we created a list of commonly used words in negative language and constrained these tweets to contain at least one of these words.

1.1.2 Automatically translated Sentiment Polarity Lexicons

The Multi–Perspective Question Answering (hereafter referred to as $MPQA$) Subjectivity Lexicon (Wilson et al., 2005). This lexicon consists of approximately 8,200 English words with their associated polarity. To use this resource for the Italian language, we translated all the entries through the Yandex translation service[1].

1.1.3 Word Embedding Lexicons

We generated four word embedding lexicons using the word2vec[2] toolkit (Mikolov et al., 2013). As recommended in (Mikolov et al., 2013), we used the CBOW model that learns to predict the word in the middle of a symmetric window based on the sum of the vector representations of the words in the window. For our experiments, we considered a context window of 5 words. The Word Embedding Lexicons starting from the following corpora which were tokenized and postagged by the PoS tagger for Twitter described in (Cimino and Dell'Orletta, 2016):

- The first lexicon was built using the itWaC corpus[3]. The itWaC corpus is a 2 billion word corpus constructed from the Web limiting the crawl to the .it domain and using medium-frequency words from the Repubblica corpus and basic Italian vocabulary lists as seeds.

- The second lexicon was built using the set of the 50,000,000 tweets we downloaded to build the sentiment polarity lexicons previously described in subsection 1.1.1

[1] http://api.yandex.com/translate/
[2] http://code.google.com/p/word2vec/
[3] http://wacky.sslmit.unibo.it/doku.php?id=corpora

- The third and the fourth lexicon were built using a corpus consisting of 538,835 Booking reviews scraped from the web. Since each review in the Booking site is split in a positive secion (indicated by a plus mark) and negative section (indicated by a minus mark), we split these reviews obtaining in 338,494 positive reviews and 200,341 negative reviews. Starting from the positive and the negative reviews, we finally obtained two different word embedding lexicons.

Each entry of the lexicons maps a pair (word, POS) to the associated word embedding, allowing to mitigate polisemy problems which can lead to poorer results in classification. In addition, both the corpora where preprocessed in order to 1) map each url to the word "URL" 2) distinguish between all uppercased words and non-uppercased words (eg.: "mai" vs "MAI"), since all uppercased words are usually used in negative contexts. Since each task has its own characteristics in terms of information that needs to be captured from the classifiers, we decided to use a subset of the word embeddings in each task. Table 1 sums up the word embeddings used in each shared task.

Task	Booking	ITWAC	Twitter
ABSITA	✓	✓	✗
GxG	✗	✓	✓
HaSpeeDe	✗	✓	✓
IronITA	✗	✓	✓

Table 1: Word embedding lexicons used by our system in each shared task (✓used; ✗not used).

1.2 The Classifier

The classifier we built for our participation to the tasks was designed with the aim of testing different learning algorithms and learning strategies. More specifically our classifier implements two workflows which allow testing SVM and recurrent neural networks as learning algorithms. In addition, when recurrent neural networks are chosen as learning algorithms, our classifier allows to perform neural network multi-task learning (MTL) using an external dataset in order to share knowledge between related tasks. We decided to test the MTL strategy since, as demonstrated in (De Mattei et al., 2018), it can improve the performance of the classifier on emotion recognition tasks. The benefits of this approach were investigated also by Søgaard and Goldberg (2016), which showed that MTL is appealing since it allows to incorporate previous knowledge about tasks hierarchy into neural networks architectures. Furthermore, Ruder et al. (2017) showed that MTL is useful to combine even loosely related tasks, letting the networks automatically learn the tasks hierarchy.

Both the workflows we implemented share a common pattern used in machine learning classifiers consisting of a document feature extraction and a learning phase based on the extracted features, but since SVM and Bi-LSTM take input 2-dimensional and 3-dimensional tensors respectively, a different feature extraction phase is involved for each considered algorithm. In addition, when the Bi-LSTM workflow is selected the classifier can take as input an extra file which will be used to exploit the MTL learning approach. Furthermore, when the Bi-LSTM workflow is selected, the classifier performs 5-fold training approach. More precisely we build 5 different models using different training and validation sets. These models are then exploited in the classification phase: the assigned labels are the ones that obtain the majority among all the models. The 5-fold approach strategy was chosen in order to generate a global model which should less be prone to overfitting or underfitting w.r.t. a single learned model.

1.2.1 The SVM classifier

The SVM classifier exploits a wide set of features ranging across different levels of linguistic description. With the exception of the *word embedding combination*, these features were already tested in our previous participation at the EVALITA 2016 SENTIPOLC edition (Cimino et al., 2016). The features are organised into three main categories: *raw and lexical text features*, *morpho-syntactic features* and *lexicon features*. Due to size constraints we report only the feature names.

Raw and Lexical Text Features number of tokens, character n-grams, word n-grams, lemma n-grams, repetition of n-grams chars, number of mentions, number of hashtags, punctuation.

Morpho-syntactic Features coarse grained Part-Of-Speech n-grams, Fine grained Part-Of-Speech n-grams, Coarse grained Part-Of-Speech distribution

Lexicon features Emoticons Presence, Lemma sentiment polarity n-grams, Polarity modifier, PMI score, sentiment polarity distribution, Most frequent sentiment polarity, Sentiment polarity in text sections, Word embeddings combination.

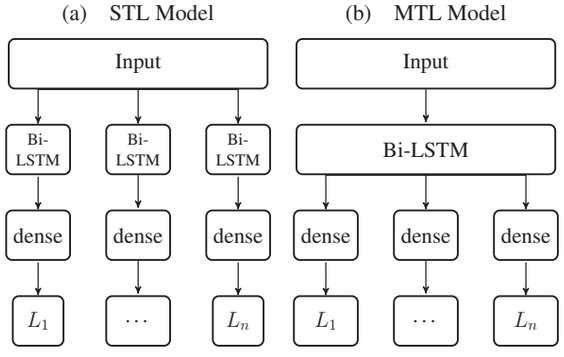
1.2.2 The Deep Neural Network classifier

We tested two different models based on Bi-LSTM: one that learns to classify the labels without sharing information from all the labels in the training phase (Single task learning - STL), and the other one which learns to classify the labels exploiting the related information through a shared Bi-LSTM (Multi task learning - MTL). We employed Bi-LSTM architectures since these architectures allow to capture long-range dependencies from both directions of a document by constructing bidirectional links in the network (Schuster et al., 1997). We applied a dropout factor to both input gates and to the recurrent connections in order to prevent overfitting which is a typical issue in neural networks (Galp and Ghahramani, 2015). We have chosen a dropout factor value of 0.50.

For what concerns GxG, as we had to deal with longer documents such as news, we employed a two layer Bi-LSTM encoder. The first Bi-LSTM layer served us to encode each sentence as a token sequence, the second layer served us to encode the sentences sequence. For what concerns ironITA we added a task-specifici Bi-LSTM for each substask before the dense layer.

Figure 1 shows a graphical representation of the STL and MTL architectures we employed. For what concerns the optimization process, the binary cross entropy function is used as a loss function and optimization is performed by the rmsprop optimizer (Tieleman and Hinton, 2012).

Figure 1: STL and MTL architectures.



Each input word is represented by a vector which is composed by:

Word embeddings: the concatenation of the word embeddings extracted by the available Word Embedding Lexicons (128 dimensions for each word embedding), and for each word embedding an extra component was added to handle the "unknown word" (1 dimension for each lexicon used).

Word polarity: the corresponding word polarity obtained by exploiting the Sentiment Polarity Lexicons. This results in 3 components, one for each possible lexicon outcome (negative, neutral, positive) (3 dimensions). We assumed that a word not found in the lexicons has a neutral polarity.

Automatically Generated Sentiment Polarity Lexicons for Social Media: The presence or the absence of the word in a lexicon and the relative presence if the word is found in the lexicon. Since we built the TW_{GEN}, TW_{POS} and TW_{NEG} 6 dimensions are needed, 2 for each lexicon.

Coarse Grained Part-of-Speech: 13 dimensions.

End of Sentence: a component (1 dimension) indicating whether the sentence was totally read.

2 Results and Discussion

Table 2 reports the official results obtained by our best runs on all the task we participated. As it can be noted our system performed extremely well, achieving the best scores almost in every single subtask. In the following subsections a discussion of the results obtained in each task is provided.

2.1 ABSITA

We tested five learning configurations of our system based on linear SVM and DNN learning algorithms using the features described in section 1.2.1 and 1.2.2. All the experiments were aimed at testing the contribution in terms of f-score of MTL vs STL, the k-fold technique and the external resources. For what concerns the Bi-LSTM learning algorithm we tested Bi-LSTM both in the STL and MTL scenarios. In addition, to test the contribution of the Booking word embeddings, we created a configuration which uses a shallow Bi-LSTM in MTL setting without using these embeddings (MTL NO BOOKING-WE). Finally, to test the contribution of the k-fold technique we created a configuration which does not use the k-fold technique (MTL NO K-FOLD). To obtain fair comparisons in the last case we run all the experiments 5 times and averaged the scores of the runs. To test the proposed classification models, we created

Task	Our Score	Best Score	Rank
ABSITA			
ACD	0.811	0.811	1
ACP	0.767	0.767	1
GxG IN-DOMAIN			
CH	0.640	0.640	1
DI	0.676	0.676	1
JO	0.555	0.585	2
TW	0.595	0.595	1
YT	0.555	0.555	1
GxG CROSS-DOMAIN			
CH	0.640	0.640	1
DI	0.595	0.635	2
JO	0.510	0.515	2
TW	0.609	0.609	1
YT	0.513	0.513	1
HaSpeeDe			
TW	0.799	0.799	1
FB	0.829	0.829	1
C_TW	0.699	0.699	1
C_FB	0.607	0.654	5
IronITA			
IRONY	0.730	0.730	1
SARCASM	0.516	0.520	3

Table 2: Classification results of our best runs on the ABSITA, GxG, HaSpeeDe and IronITA test sets.

an internal development set by randomly selecting documents from the training sets distributed by the task organizers. The resulting development set is composed by approximately the 10% (561 documents) of the whole training set.

Configuration	ACD	ACP
baseline	0.313	0.197
linear SVM	0.797	0.739
STL	0.821	0.795
MTL	**0.824**	**0.804**
MTL NO K-FOLD	0.819	0.782
MTL NO BOOKING-WE	0.817	0.757

Table 3: Classification results (micro f-score) of the different learning models on our ABSITA development set.

Table 3 reports the overall accuracies achieved by the models on the internal development set for all the tasks. In addition, the results of baseline system (baseline row) which emits always the most probable label according to the label distribu-

Configuration	ACD	ACP
baseline	0.338	0.199
linear SVM	0.772*	0.686*
STL	**0.814**	0.765
MTL	0.811*	**0.767***
MTL NO K-FOLD	0.801	0.755
MTL NO BOOKING-WE	0.808	0.753

Table 4: Classification results (micro f-score) of the different learning models on the ABSITA official test set.

tions in the training set is reported. The accuracy is calculated as the micro f–score obtained using the evaluation tool provided by the organizers. For what concerns the ACD task it is worth noting that the models based on DNN always outperform linear SVM, even though the difference in terms of f-score is small (approximately 2 f-score points). The MTL configuration was the best performing among all the the models, but the difference in term of f-score among all the DNN configuration is not evident.

When analyzing the results obtained on the ACP task we can notice remarkable differences among the performances obtained by the models. Again the linear SVM was the worst performing model, but this time with a difference in terms of f-score of 6 points with respect to MTL, the best performing model on the task. It is interesting to notice that the results achieved by the DNN models have bigger difference between them in terms of f-score with respect to the ACD task: this suggests that the external resources and the k-fold technique contributed significantly to obtain the best result in the ACP task. The configuration that does not use the k-fold technique scored 2 f-score points w.r.t. the MTL configuration. We can also notice that the Booking word emebeddings were particularly helpful in this task: the MTL NO BOOOKING-WE configuration in fact scored 5 points less than the best configuration. The results obtained on the internal development set lead us to choose the models for the official runs on the provided test set. Table 4 reports the overall accuracies achieved by all our classifier configurations on the official test set, the official submitted runs are starred in the table.

As it can be noticed the best scores both in the ACD and ACP tasks were obtained by the DNN

models. Surprisingly the difference in terms of f-score were reduced in both the tasks, with the exception of linear SVM, which performed 4 and 8 f-score points less in the ACD and ACP tasks respectively when compared to the best DNN model systems. The STL model outperformed the MTL models the ACD task, even though the difference in term of f-score is not relevant. When the results on the ACP are considered, the MTL model outperformed all the other models, even though the the difference in terms of f-score with respect to the STL model is not noticeable. Is it worth to notice that the k-fold technique and the Booking word embeddings seemed to again contribute in the final accuracy of the MTL system. This can be seen by looking at the results achieved by the MTL NO BOOKING-WE model and the MTL NO K-FOLD model that scored 1.2 and 1.5 f-score points less than the MTL system.

2.2 GxG

We tested three different learning configurations of our system based on linear SVM and DNN learning algorithms using the features described in section 1.2.1 and 1.2.2. For what concerns the Bi-LSTM learning algorithm we tested both the STL and MTL approaches. We tested the three configurations for each of the 5 five in-domain subtasks and for each of the 5 five cross-domain subtasks. To test the proposed classification models, we created internal development sets by randomly selecting documents from the training sets distributed by the task organizers. The resulting development sets are composed by approximately 10% of the each data sets. For what concern the in-domain task, we tried to train the SVM classifier on in-domain-data only and and on both in-domain and cross-domain data.

Model	CH	DI	JO	TW	YT
SVMa	0.667	0.626	0.485	0.582	0.611
SVM	**0.701**	0.737	0.560	0.728	0.619
STL	0.556	0.545	0.500	0.724	0.596
MTL	0.499	**0.817**	**0.625**	**0.729**	**0.632**

Table 5: Classification results of the different learning models on development set in terms of accuracy for the **in-domain** tasks.

Table 5 and 6 report the overall accuracy, computed as the average accuracy for the two classes (male and female), achieved by the models on the development data sets for the in-domain and the cross-domain tasks respectively. For the in-domain tasks we observe that the SVM performs well on the smaller datasets (Children and Diary), while MTL neural network has the best overall performances. When trained on all the datasets, in- and cross-domain, the SVM (SVMa) perform worst than when trained on in-domain data only (SVM). For what concerns the cross-domain datasets we observe poor performances over all the subtasks with all the employed models, implying that the models have difficulties in cross-domain generalization.

Model	CH	DI	JO	TW	YT
SVM	0.530	0.565	0.580	0.588	0.568
STL	0.550	0.535	0.505	0.625	0.580
MTL	0.523	0.549	0.538	0.500	0.556

Table 6: Classification results of the different learning models on development set in terms of accuracy for the **cross-domain** tasks

Model	CH	DI	JO	TW	YT
SVMa	0.545	0.514	0.475	0.539	**0.585**
SVM	0.550	0.649	**0.555**	0.567	0.555*
STL	0.545	0.541	0.500	**0.595***	0.512
MTL	**0.640***	**0.676***	0.470	0.561	0.546

Table 7: Classification results of the different learning models on the official test set in terms of accuracy for the **in-domain** tasks (* marks runs that outperformed all the systems that participated to the task).

Model	CH	DI	JO	TW	YT
SVM	0.540	0.514	**0.505**	0.586	**0.513***
STL	**0.640***	0.554	0.495	**0.609***	0.510
MTL	0.535	**0.595**	0.510	0.500	0.500

Table 8: Classification results of the different learning models on the official test set in terms of accuracy for the **cross-domain** tasks. (* marks runs that outperformed all the systems that participated to the task).

Table 7 and 8 report the overall accuracy, computed as the average accuracy for the two classes (male and female), achieved by the models on the official test sets for the in-domain and the cross-domain tasks respectively (* marks the running that obtain the best results in the competition). For what concerns the in-domain subtasks the perfor-

mances appear to be not in line with the ones obtained on the development set, but still our models outperform the other participant's systems in four out of five subtasks. The MTL model provided the best results for the Children and Diary test sets, while on the other test sets all the models performed quite poorly. Again when trained on all the datasets, in and cross-domain, the SVM (SVMa) perform worst then when trained on in-domain data only (SVM). For what concerns the cross-domain subtasks, while our model gets the best performances on three out of five subtasks, the results confirm poor performances over all the subtasks, again indicating that the models have difficulties in cross-domain generalization.

2.3 HaSpeeDe

We tested seven learning configurations of our system based on linear SVM and DNN learning algorithms using the features described in section 1.2.1 and 1.2.2. All the experiments were aimed at testing the contribution in terms of f-score of the number of layers, MTL vs STL, the k-fold technique and the external resources. For what concerns the Bi-LSTM learning algorithm we tested one and two layers Bi-LSTM both in the STL and MTL scenarios. In addition, to test the contribution of the sentiment lexicon features, we created a configuration which uses a 2-layer Bi-LSTM in MTL setting without using these features (1L MTL NO SNT). Finally, to test the contribution of the k-fold technique we created a configuration which does not use the k-fold technique (1 STL NO K-FOLD). To obtain fair results in the last case we run all the experiments 5 times and averaged the scores of the runs. To test the proposed classification models, we created two internal development sets, one for each dataset, by randomly selecting documents from the training sets distributed by the task organizers. The resulting development sets are composed by the 10% (300 documents) of the whole training sets.

Table 9 reports the overall accuracies achieved by the models on our internal development sets for all the tasks. In addition, the results of baseline system (baseline row) which emits always the most probable label according to the label distribution in the training set is reported. The accuracy is calculated as the f–score obtained using the evaluation tool provided by the organizers. For what concerns the Twitter in–domain task (TW in the table) it is worth noting that linear SVM outperformed all the configurations based on Bi-LSTM. In addition, the MTL architecture results are slightly better than the STL ones (+1 f-score point with respect to the STL counterparts). External sentiment resources were not particularly helpful in this task, as shown by the result obtained by the 1L MTL NO SNT row. In the FB task, Bi-LSTMs sensibly outperformed linear SVMs (+5 f-score points in average); this is most probably due to longer text lengths that are found in this dataset with respect to the Twitter one. For what concerns the out–domain tasks, when testing models trained on Twitter and tested on Facebook (C_TW column), we can notice an expected drop in performance with respect to the models trained on the FB dataset (15-20 points f-score points). The best result was achieved by the 2L MTL configuration (+4 points w.r.t. the STL counterpart). Finally, when testing the models trained on Facebook and tested on Twitter (C_FB column), linear SVM showed a huge drop in terms of accuracy (-30 f-score points), while all the models trained with Bi-LSTM showed a performance drop of approximately 12 f-score points. Also in this

Configuration	TW	FB	C_TW	C_FB
baseline	0.378	0.345	0.345	0.378
linear SVM	**0.800**	0.813	0.617	0.503
1L STL	0.774	**0.860**	0.683	0.647
2L STL	0.790	**0.860**	0.672	0.597
1L MTL	0.783	**0.860**	0.672	**0.663**
2L MTL	0.796	0.853	**0.710**	0.613
1L MTL NO SNT	0.793	0.857	0.651	0.661
1L STL NO K-FOLD	0.771	0.846	0.657	0.646

Table 9: Classification results of the different learning models on our HaSpeeDe development set in terms of F1-score.

Configuration	TW	FB	C_TW	C_FB
baseline	0.403	0.404	0.404	0.403
best official system	0.799	0.829	0.699	0.654
linear SVM	0.798*	0.761	0.658	0.451
1L STL	0.793	0.811*	0.669*	0.607*
2L STL	0.791	0.812	0.644	0.561
1L MTL	0.788	0.818	**0.707**	**0.635**
2L MTL	0.799*	**0.829***	0.699*	0.585*
1L MTL NO SNT	**0.801**	0.808	0.709	0.620
1L STL NO FOLD	0.785	0.806	0.652	0.583

Table 10: Classification results of the different learning models on the official HaSpeeDe test set in terms of F1-score.

setting the best result was achieved by a MTL configuration (1L MTL), which performed better with respect to the STL counterpart (+2 f-score points). For what concerns the k-fold learning strategy, we can notice that the results achieved by the model not using the k-fold learning strategy (1 STL NO K-FOLD) are always lower than the counterpart which used the k-fold approach (+2.5 f-score points gained in the C_TW task), showing the benefits of using this technique.

These results lead us to choose the models for the official runs on the provided test set. Table 10 reports the overall accuracies achieved by all our classifier configurations on the official test set, the official submitted runs are starred in the table. The *best official system* row reports, for each task, the best official results submitted by the participants of the EVALITA 2018 HaSpeeDe shared task. As we can note the best scores in each task were obtained by the Bi-LSTM in the MTL setting, showing that MTL networks seem to be more effective with respect to STL networks. For what concerns the Twitter in–domain task, we obtained similar results to the development set ones. A sensible drop in performance is observed in the FB task w.r.t the development set (-5 f-score points in average). Still Bi-LSTMs models outperformed the linear SVM model by 5 f-score points. In the out-domain tasks, all the models performed similarly to what observed in the development set. It is worth observing that linear SVM performed almost as a baseline system in the C_FB task. In addition, in the same task the model exploiting the sentiment lexicon (1L MTL) showed a better performance (+1.5 f-score points) w.r.t to the 1L MTL NO SNT model. It is worth to notice that the k-fold learning strategy was beneficial also on the official test set: the 1L STL model obtained better results (approximately +2 f-score points in each task) w.r.t. the model that did not use the k-fold learning strategy.

2.4 IronITA

We tested the four designed learning configurations of our system based on linear SVM and deep neural network (DNN) learning algorithms using the features described in section 1.2.1 and 1.2.2. To select the proposed classification models, we used k-cross validation (k=4).

Table 11 reports the overall average f-score achieved by the models on the k-cross validation sets for both the irony and sarcarsm detection tasks.

Configuration	Irony	Sarcasm
linear SVM	0.734	0.512
MTL	0.745	0.530
MTL+Polarity	0.757	**0.562**
MTL+Polarity+Hate	**0.760**	0.557

Table 11: Classification results of the different learning models on k-cross validation terms of average F1-score.

Configuration	Irony	Sarcasm
baseline-random	0.505	0.337
baseline-mfc	0.334	0.223
best participant	0.730	0.52
linear SVM	0.701	0.493
MTL	**0.736**	**0.530**
MTL+Polarity	0.730*	0.516*
MTL+Polarity+Hate	0.713*	0.503*

Table 12: Classification results of the different learning models on the official test set in terms of F1-score (* submitted run).

We can observe that the SVM obtains good results on irony detection but the MTL neural approach overperforms sensibly the SVM. Also we note that the usage of additional Polarity and Hate Speech datasets lead to better performances. These results lead us to choose the MTL models trained with the additional dataset for the two official run submissions.

Table 12 reports the overall accuracies achieved by all our classifier configurations on the official test set, the official submitted runs are starred in the table. The accuracies has been computed in terms of F-Score using the official evaluation script. We submitted the runs MTL+Polarity and MTL+Polarity+Hate. The run MTL+Polarity ranked first in the subtask A, and third in the subtask B on the official leaderboard. The run MTL+Polarity ranked second in the subtask A, and fourth in the subtask B on the official leaderboard.

The results on the test set confirm the good performances of the SVM classifier on irony detection task and that the MTL neural approaches overperform the SVM. The model trained on the IronITA and SENTIPOLC datasets outperformed all the systems that participated to the subtask A, while on the subtask B it slightly underperformed the best participant system. The model trained on

the IronITA, SENTIPOLC and HaSpeeDe datasets overperformed all the systems that participated to the subtask A but our model trained on IronITA and SENTIPOLC datasets only. Although the best scores in both tasks were obtained by the MTL network trained on IronITA data set only. The MTL model trained on IronITA dataset only would have outperformed all the systems submitted to both the subtasks by all participants. Seems that for these tasks the usage of additional datasets leads to overfitting issues.

3 Conclusions

In this paper we reported the results of our participation to the ABSITA, GxG, HaSpeeDe and IronITA shared tasks of the EVALITA 2018 conference. By resorting to a system which used Support Vector Machines and Deep Neural Networks (DNN) as learning algorithms, we achieved the best scores almost in every task, showing the effectivness of our approach. In addition, when DNN was used as learning algorithm we introduced a new multi-task learning approach and a majority vote classification approach to further improve the overall accuracy of our system. The proposed system resulted in an very effective solution achieving the first position in almost all sub-tasks for each shared task.

Acknowledgments

We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Tesla K40 used for this research.

References

Francesco Barbieri, Valerio Basile, Danilo Croce, Malvina Nissim, Nicole Novielli, Viviana Patti. 2016. Overview of the EVALITA 2016 SENTiment POLarity Classification Task. In *Proceedings of EVALITA '16, Evaluation of NLP and Speech Tools for Italian*. December, Naples, Italy.

Pierpaolo Basile, Valerio Basile, Danilo Croce, Marco Polignano. 2018. Overview of the EVALITA Aspect-based Sentiment Analysis (ABSITA) Task. T. Caselli, N. Novielli, V. Patti, P. Rosso, (eds). In *Proceedings of the 6th evaluation campaign of Natural Language Processing and Speech tools for Italian (EVALITA'18)*. December, Turin, Italy.

Cristina Bosco, Felice dell'Orletta, Fabio Poletto, Manuela Sanguinetti and Maurizio Tesconi. 2018. Overview of the Evalita 2018 Hate Speech Detection Task. T. Caselli, N. Novielli, V. Patti, P. Rosso, (eds). In *Proceedings of EVALITA '18, Evaluation of NLP and Speech Tools for Italian*. December, Turin, Italy.

Franois Chollet. 2016. Keras. *Software available at https://github.com/fchollet/keras/tree/master/keras*.

Alessandra Cignarella and Simona Frenda and Valerio Basile and Cristina Bosco and Viviana Patti and Paolo Rosso. 2018. Overview of the Evalita 2018 Task on Irony Detection in Italian Tweets (IronITA). T. Caselli, N. Novielli, V. Patti, P. Rosso, (eds). In *Proceedings of the 6th evaluation campaign of Natural Language Processing and Speech tools for Italian (EVALITA'18)*

Andrea Cimino and Felice dell'Orletta. 2016. Tandem LSTM-SVM Approach for Sentiment Analysis. In *Proceedings of EVALITA '16, Evaluation of NLP and Speech Tools for Italian*. December, Naples, Italy.

Andrea Cimino and Felice dell'Orletta. 2016. Building the state-of-the-art in POS tagging of Italian Tweets. In *Proceedings of Third Italian Conference on Computational Linguistics (CLiC-it 2016) & Fifth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2016), Napoli, Italy, December 5-7, 2016.*

Lorenzo de Mattei, Andrea Cimino and Felice dell'Orletta. 2018. Multi-Task Learning in Deep Neural Network for Sentiment Polarity and Irony classification. In *Proceedings of the 2nd Workshop on Natural Language for Artificial Intelligence, Trento, Italy, November 22-23, 2018.*

Felice Dell'Orletta and Malvina Nissim. 2018. Overview of the EVALITA Cross-Genre Gender Prediction in Italian (GxG) Task. T. Caselli, N. Novielli, V. Patti, P. Rosso, (eds). In *Proceedings of the 6th evaluation campaign of Natural Language Processing and Speech tools for Italian (EVALITA'18)*.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang and Chih-Jen Lin 2008. LIBLINEAR: A Library for Large Linear Classification *Journal of Machine Learning Research*. Volume 9, 1871–1874

Yarin Gal and Zoubin Ghahramani. 2015. A theoretically grounded application of dropout in recurrent neural networks. *arXiv preprint arXiv:1512.05287*

Sepp Hochreiter and Jurgen Schmidhuber. 1997. Long short-term memory. *Neural computation*

Tomas Mikolov, Kai Chen, Greg Corrado and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. arXiv preprint arXiv1:1301.3781.

Preslav Nakov, Alan Ritter, Sara Rosenthal, Fabrizio Sebastiani and Veselin Stoyanov. 2016. SemEval-2016 task 4: Sentiment analysis in Twitter. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*

Sebastian Ruder, Joachim Bingel, Isabelle Augenstein and Anders Søgaard. 2017. Document modeling with gated recurrent neural network for sentiment classification. arXiv preprint arXiv:1705.0814 1422-1432, Lisbon, Portugal.

Mike Schuster and Kuldip K. Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing* 45(11):2673–2681

Anders Søgaard and Yoav Goldberg. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. 231–235, Berlin, Portugal.

Duyu Tang, Bing Qin and Ting Liu. 2015. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of EMNLP 2015*. 1422-1432, Lisbon, Portugal.

Tijmen Tieleman and Geoffrey Hinton. 2012. Lecture 6.5-RmsProp: Divide the gradient by a running average of its recent magnitude. In *COURSERA: Neural Networks for Machine Learning.*

Theresa Wilson, Zornitsa Kozareva, Preslav Nakov, Sara Rosenthal, Veselin Stoyanov and Alan Ritter. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of HLT-EMNLP 2005*. 347-354, Stroudsburg, PA, USA. ACL.

XingYi Xu, HuiZhi Liang and Timothy Baldwin. 2016. UNIMELB at SemEval-2016 Tasks 4A and 4B: An Ensemble of Neural Networks and a Word2Vec Based Model for Sentiment Classification. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016).*

ItVENSES - A Symbolic System for Aspect-Based Sentiment Analysis

Rodolfo Delmonte
Dipartimento di Studi Linguistici e Culturali Comparati
Ca' Bembo – Dorsoduro 1075
Università Ca' Foscari – 30123 VENEZIA
delmont@unive.it

Abstract

English. ItVENSES is a system for syntactic and semantic processing that is based on the parser for Italian called ItGetaruns to analyse each sentence. ItVenses receives the output of ItGetaruns and decides which terms may be used as keywords or features for aspect identification. This is done at first by a simple lookup in a list created on the basis of a quantitative analysis of the training corpus. The result is sifted by activating a set of syntactic and semantic SIEVES that act upon the output constituency structure, the lemmatized and classified list of words, the predicate-argument structure(s) of the sentence. After this step, the aspect(s) associated to each sentence are enriched by the sentiment and polarity components computed on the output of ItGetaruns. Finally negation, factuality and subjectivity are considered in relation to each aspect. Results have been at first fairly low – 61% F1-score -, but after a series of ablation experiments two components of the algorithm have been reduced and the evaluation has suddenly soared reaching 83% F1-score, a value close to the one obtained for training data.

Italiano. *ItVENSES è un sistema per analisi sintattico-semantiche basato sul parser per l'italiano chiamato ItGetaruns per analizzare ogni frase. ItVenses riceve l'output di ItGetaruns e decide quali termini possono essere usati come feature o semi per identificare l'aspetto. Questo passo viene compiuto dapprima con una semplice operazione di lookup in una lista creata precedentemente sulla base di un'analisi quantitativa del corpus di training. Il risultato viene quindi vagliato attivando un insieme di filtri che agiscono sulla costituenza sintattica, la lista lemmatizzata e classificata delle parole e le strutture predicato-argomentali della frase. Dopo questo passaggio, l'aspetto associato a ciascuna frase viene arricchito dalle componenti di polarità e sentiment calcolate sull'output di ItGetaruns. Infine, vengono considerate negazione, fattualità e soggettività in relazione a ciascun aspetto. I risultati sono stati dapprima alquanto bassi – attorno al 61% di F1, ma successivamente, dopo aver eseguito una serie di esperimenti di sottrazione in cui sono stati ridotti due componenti dell'algoritmo, la valutazione ha improvvisamente avuto un'impennata raggiungendo l'83% di F1, valore simile a quello ottenuto per il training corpus.*

1 Introduction

In this paper we present work carried out to analyze aspect and polarity in a corpus of Italian tweets collected and annotated at the University of Turin (Basile et al. (2018)). The final system produced is fully symbolic, is made up by different modules and takes advantage of previous work for similar challenges presented at Clic-It 2014 (Delmonte (2014b)). In particular, the underlying parser for the semantic analysis of each text provides a full processing pipeline including tokenization, multiwords creation, morpho-syntactic analysis, POS tagging, Named Entity Detection, chunking and finally, extraction of dependency relations such as subject, object and modifiers. It also provides for pronominal binding and coreference resolution, and propositional level semantics related to negation, factuality and subjectivity. In the sections below we present in detail the method

used in the main modules, the general features of the dataset and the problems with some of its inconsistencies, the results.

2 The System and the Modules

One important step in the creation of the system ItVenses has been adaptation and contextualization which has acted on ItGetaruns – the semantic parser - at almost all levels of analysis. ItGetarun receives as input a string – the sentence to be analysed - which is then tokenized into a list. The list is then fully tagged, disambiguated and chunked. Chunks are then put together into a full sentence structure which is passed to the Island-Based predicate-argument structure (hence PAS) analyzer.

One important part of the adaptation of the system ItGetarun has been constituted by all requirements imposed by the domain at lexical, tagging, syntactic and semantic levels. Reviews on holiday resorts, hotels, touristic places have a specialized vocabulary which requires certain choices to be imposed by the components of the parser already from the start. In particular, we imposed a specific tag – here a Noun - to a set of otherwise lexically ambiguous words, as for instance in the following set of examples:

(1) torta, tavolo, fermata, pianta, insegna

where, each word could be tagged both as Noun and as PastParticiple or simply as Verb[1], A certain number of multiwords have been created in order again to reduce ambiguity of a set of words. In ItGetarun, the creation of multiwords is carried out during tokenization, thus before tagging takes place. Here are some examples,

(2) deposito bagagli, camera da letto, presa di corrente, ricevuta fiscale, sala colazione, centro storico

where again each first component could be analyzed as noun but also as verb or pastParticiple. Finally, since a great number of texts are simple fragments, made up of a list of nouns and adjectives and no verbs, we introduced a dummy verb ESSERE and marked the first noun phrase as Subject to be able to compute propositional level semantics. At semantic and pragmatic level, specific words acquire a meaning determined by the context: consider the adjective “piccolo” which is only used to mark negative polarity when predicative and as a modifier when attributive, together with a number of downtoners like ”poco”, as in the example below:

(3) 1240348699;1;1;0;1;0;1;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;"Stanza piccola ma pulita."

The problem in this case is represented by the implicit presence of ”stanza” in the elliptical portion of the text preceded by the adversative marker ”ma” which allows exclusive reference. Consider also example (4),

(4) 1240350017;0;0;0;0;0;0;1;0;1;1;1;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;"Qualche difficoltà col parcheggio nonostante la disponibilità del personale"

where aspect feature terms are not the most relevant items, syntactically and semantically speaking, but are included as modifiers in a noun phrase (”del personale”) or are treated as adjuncts prepositional phrases (”col parcheggio”). Inclusive semantic interpretation is associated to examples like (5) below,

(5) 1240347398;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;1;1;1;0;0;0;"La posizione della struttura è un po’ fuori dal centro, ma in compenso è vicina al capolinea degli autobus"

There are also idiomatic expressions which are taken into consideration and computed from PAS, as for instance ”lasciare a desiderare”, meaning ”being insufficient” rather than its literal meaning ”leave to desire”[2].

(6) 1240347831;1;0;1;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;"La pulizia lascia a desiderare per un hotel da 4 stelle."

[1]Tagging is very important to tell apart homographs like ”personale” in this example, which would be wrongly classified by a bag-of-words approach: 1240342728,"L unico difetto è che, a differenza di altri ostelli, l armadietto personale è molto piccolo."

[2]or in the example below where, however, the annotated aspect has been wrongly marked as “other” in slot 8: as will be shown in a section below, ”palazzo”, ”struttura” and ”hotel” have both been usually associated to location, slot 7. (7) 1240351211;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;1;0;1;"il palazzo dove è posta la struttura e l’accesso dalla via lascia un po’ a desiderare; tutto sembra ma non un hotel."

ItVenses, the algorithm for aspect classification and sentiment analysis – we took part in SENTIPOLC, (see Basile et al. (2014)) - takes as input the previous analysis and input files, including a list of all entities fully semantically classified and a list of all PAS with up to 4 arguments/adjuncts. Every PAS is preceded by one among three possible labels: a label NEG for negation (including negative sentiment of the lexical predicate), a label OPNSG for subjective propositions,and UNREAL for nonfactual ones; in addition to a label for speech act, STATEMENT, EXCLAMATIVE or QUESTION.

3 Sifting Aspect and Polarity with Semantic Sieves

Seeds or features present in each text are searched both at word and lemma level. This is done at first by a simple look-up operation which matches each token and corresponding lemma in the text with the list of possible seeds created by a quantitative analysis of the training corpus. In fact, this list only includes most frequent terms found and collected – amounting to 300. To reinforce this match, synonyms are added when present again by matching with the synonyms lexicon available in ItGetaruns. This was done in view of the need to enlarge the number of features to make available for the test corpus, as well as to generalize the procedure. No such lists exist for polarity items which are searched for and matched with the lexica available in ItGetaruns[3]

All these operations are subject to local filtering actions. If we consider example (6) above we see that there are three possible seeds: ”pulizia”, ”hotel”, ”4 stelle”, but the focus of attention is on ”pulizia” which also is the Subject of the main predicate. So one filter is determined by grammatical functions (for a similar approach see Brun and Raux (2016)) extracted by constituency and dependency analysis and made available one by one with the corresponding head. ”Pulizia” and related lexemes are then regarded more relevant than the simple seed ”camera” or ”stanza” and the choice is to delete aspect 2 in favour of aspect 1 when this is verified. Also consider positive words like MIGLIORARE which are usually associated to negative evaluation. Aspect and sentiment polarities (negative and positive), are then checked together one by one, in order to verify whether polarity has to be attenuated, shifted or inverted (see Polanyi and Zaenen (2006)) as a result of the presence of intensifiers, maximizers, minimizers, diminishers, or simply negations at a higher than constituent level (see Ohana and Delany (2006)). Consider now the presence of focalizers like ”solo, soltanto”, which is mostly used to focus on the insufficient presence of a given aspect related feature, as for instance in this case:

> (7) 1240344222;0;0;0;0;0;0;1;0;1;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;"In bagno 1 solo schampo e 1 solo bagno doccia per 2 persone."

Other important components are privative markers like ”senza, eccetto” but also specific words indicating ”lack of”, ”mancanza, assenza, privo di, in cerca di”. Specific markers are related to Aspect 7 – location one, and are markers for negative evaluation: ”lontano da, fuori da ”usually referring to city center. Eventually, all aspects plus polarities are collapsed in one single list for each sentence and passed to a final SIEVE that acts on more than one aspect at a time in order to establish preferences for pairs of aspects and erase redundantly assigned ones. Additional work on preferences will be discussed in the section below. Here it is important to note that in some cases presence of a specific feature to identify the corresponding aspect may be implicit, i.e. not linguistically expressed. Consider for instance the following example:

> (8) 1240347807;1;1;0;1;1;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;"Pulita, spaziosa e soprattutto funzionale."

where the word ”camera” is missing, or the following example where ”gradini” implies ”scala”. [4]

> (9) 1240345322;0;0;0;0;0;0;1;0;1;0;0;0;0;0;0;0;0;0;0;0;0;1;0;1;"Molti gradini 93 Ascensore piccolo."

Another set of interesting examples are constituted by those reviews declaring their total appraisal or sometimes their total lack of appraisal of the place:

[3]The list has been derived - checking and re-elaborating the data - from a number of previous lexica as for instance the one by Esuli and Sebastiani (2006).

[4]But in (9) the annotation is not consistent: both ”scala” and ”ascensore” are mainly annotated with aspect 2, rather than aspect 3 present here. None of them apart for a small number of cases are annotated with 8, “Other”.

(10) 1240346791;1;1;0;1;1;0;1;1;0;1;1;0;1;1;0;1;1;0;1;1;0;1;1;0;"niente da reclamare; tutto perfetto"

(11) 1240344015;0;0;0;1;1;0;1;1;0;1;1;0;1;1;0;0;0;0;0;0;0;0;0;0;"tutto molto bello e professionale"

These examples are redundantly marked with positive evaluation associated to all, or almost all aspects. However the next set of examples has no such marking and is contrasted by example (14) with almost the same text:

(12) 1240345792;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;1;1;0;"Mi è piaciuto praticamente tutto."

(13) 1240344497;1;1;0;1;1;0;1;1;0;1;1;0;1;1;0;1;1;0;1;1;0;1;1;0;"Mi è piaciuto tutto"

Eventually, ItVenses takes into account negation and non-factuality usually marked by unreal mood, information available at propositional level, used to modify previously assigned polarity from negative to positive, on the basis of PAS and their semantics. Consider the example below where double and triple negations are used to produce a positive effect:

(14) 1240345153;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;1;1;0;"Non c'è niente che non mi è piaciuto."

Non-factuality and subjectivity are used to mark negative polarity for the simple reason that unreal mood is usually associated to criticism for some service or comfort missing, as for instance in the following example,

(15) 1240344698;0;0;0;0;0;0;0;1;0;1;0;0;0;0;0;0;0;0;0;0;0;0;0;0;"Tuttavia sarebbe comodo un servizio navetta per il centro città soprattutto la sera."

4 Defining Preferences from Persistence

Telling seeds or features from terms is the difficult part of the task of aspect identification which is done here on the basis of preferences. Preferences have been partly determined by the quantitative analysis of the training corpus. After tagging and lemmatizing each word or term contained in each review text, we associated the numerical value/s of the aspect/s present in the annotation for the current text to each nominal expression. The idea was to come up with a list of frequency values for each term that allowed us to choose a "majority class" aspect to associate to the seeds. We did not expect a complete uniform distribution, i.e. that every seed had one and only one majority class and the rest of the frequency distribution were flat. There are three possible cases to consider: a. the text contains only one term which is annotated accordingly; b. the text contains more than one term, but only one is annotated; c. the text contains more than one term and they are all annotated. In case c. we associate each value to each seed thus producing a redundant annotation since we don't know which term has been associated to which aspect value. Taking into account each of the most frequent possible seeds or majority class lemmata associated to each aspect class, we computed indices for their persistence in that particular class, thus measuring level of dispersion, corresponding to ambiguity or uncertainty when choosing it. To produce such indices we considered the number of times in which a seed was associated to a given aspect class in texts with a unique aspect identifier (case a. above) rather than as part of a cluster of seeds for the same text. At first we consider general results of the quantitative analysis in Table 1. Seeds collected amount to a total of 25468 occurrences of nominal expressions which collapse down to 2678 unique types, distributed over the 8 aspect classes with a fairly unequal share, with aspect 2 and 3 covering almost half of all occurrences, followed by aspect 4, 7 and eventually but much lower, aspect 1, 5, 8 and 6 at the lowest. The majority of all nominal expressions, 16594, are contained in review texts with a unique aspect identifier, and this should make automatic assignment easier, but as can be seen from Table 3 below, this is not the case.

Aspect	**2**	**3**	**4**	**7**
Frequency	10623	8386	5488	5927

Table 1: Most frequent quantitative data of aspects and aspect clusters in training corpus

Multiple aspect annotation for the same text amounts to 8974 and they are mainly made by Doubles, then Triples, and finally Quadruples, and Quintuples. There are a few cases of empty evaluation[5], and also a few cases of all slots

[5]Here two examples of empty evaluation:

filled up – see below. Going back now to the question of uniformity of annotation and/or consistency of aspect/feature association, organizers have computed the usual inter-annotator agreement per class and reported a lower boundary of 85% of agreed sentences. However, we note the following situation: if all unique aspect identifiers in texts were consistently assigned to the same term or seed, then there would be no ambiguity and the algorithm would be easily working. But even unique identifiers do not show such persistence.[6] Most persistent seeds are the following ones in a graded scale order in list (1): WIFI, POSIZIONE, PERSONALE, STAZIONE, STAFF, CENTRO, PULIZIA; and in a lesser degree in list (2): METRO, PARCHEGGIO, DOCCIA, PREZZO. All these seeds have a persistence ratio respectively over 90% - the first list – and over 80%, the second list. In Table 3 we report some of the remaining high frequency seeds, where percentages in column three register the ratio between total of occurrences of the seed with respect to majority class; and in column four, the ratio between majority class and unique aspect identifiers. These seeds have a much lower level of persistence – well below 80% -, ”struttura” being the lowest. A low level of persistence indicates the fact that, for instance, the seed ”struttura” has been associated to a great variety of aspect class markers of which two are however paramount, and are marked in column two, with aspect 8 (Other) the most frequent, and aspect 2 (comfort), the less frequent. In two cases, we have high values associated with column 4, for ”rapporto” and ”materasso” indicating that these two seeds when found in unique aspect texts, have always or almost always persisted in the same annotated aspect marker.

Seeds / Aspect	Asp1 - Asp2	% MajCl Tot.Occs	% MajCl Tot.Uniqs
zona	**7 - 2**	**72,44**	**66,29**
vista	**7 - 3**	**62,09**	**49,46**
struttura	**8 - 2**	**18,57**	**30,74**
servizi/o	**3 - 2**	**56,97**	**38,46**
qualità	**5 - 3**	**52,72**	**45,84**
notte	**2 - 3**	**50,28**	**38,94**
hotel	**7 - 3**	**32,99**	**27,27**
arredo / amento	**2 - 2**	**58,46**	**45,57**
albergo	**7 - 2**	**36,56**	**30,97**
4_stelle	**3 - 2**	**32,77**	**28,57**
camera	**2 - 1**	**69,32**	**59,84**
ascensore	**3 - 2**	**69**	**64,18**
materasso	**2 - 3**	**71,01**	**100**
rapporto	**5 - 7**	**64,82**	**98,68**

Table 2: 19 less persistent majority class aspect seeds

If we consider aspect 8 or “Other”, in the majority of the cases, we take it to be a case of failure to annotate the text with the correct aspect class rather than a case of lack of aspect seed. As shown in table 2 above, the number of such associations is fairly high, amounting to 1586 cases. Seeds marked with 8, in order of their frequency, are: ”struttura”, ”camera”, ”hotel”, ”stanza”, ”arredo”, ”bagno”, ”4 stelle”. All cases of ”struttura” marked with class 8 are cases of unique aspect identifier, followed by ”hotel”, ”arredo”, and ”4 stelle”. The remaining cases are scattered among all seeds.

5 Results and Discussion

Results obtained and delivered in due time are not particularly satisfactory as shown in Table 4 below. When compared to results obtained for the training set, we notice a great difference. In order to understand the reason for that difference we started a set of ablation experiments, removing at first syntactic filtering and then lexical resources one by one. This was done also to evaluate what contribution was obtained from lexical resources we were using. After ablation test removing free

(i)1240349964;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;"Decisamente meglio la deluxe che ho già occupato altre volte e con costi inferiori."
(ii)1240350466;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;"Alle 9 45 la colazione era piuttosto scarsa per il prezzo di 15 Euro." If example ii. may be still regarded as a case of implicit association of aspect 2 due to presence of attribute ”deluxe” which is usually referrable to ”camera”; no such situation is found in example i. where ”colazione” and ”prezzo” are clear seeds for aspects 3 and 5.

[6] This may be simply due to the fact that a given seed may be less ”relevant” than another present in the same text. Some examples below:

(16) 1240344314;0;0;0;1;1;0;1;1;0;1;1;0;0;0;0;1;1;0;1;1;0;0;0;0;"Ottima posizione, hotel in rifacimento per cui c’è una cura verso l’ammodernamento, ottima colazione, favoloso wifi internet ho fatto una video call senza problemi ed ero in camera, camera e bagno molto ampi, personale gentile"

(17) 1240343916;0;0;0;0;0;0;1;1;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;"La vasta camera ed il bagno confortevole, il servizio WIFI e la piacevole colazione"

In (16) all aspects have been correctly annotated, not so in (17). In this examples there are many ”relevant” aspects to be annotated: aspect 2, 3 and 6.

synonym search for both aspect feature items and polarity items we discovered that a fully different result has been obtained and from the evaluation algorithm it was by far the best result – see Table 5 - in line with what we obtained in training data, which we report here below in Table 4.

Tasks/Results	ACD	ACP
Macro-Precis	0.8414	0.7705
Macro-Recall	0.8493	0.7916
Macro-F1score	0.8453	0.7809
Micro-Precis	0.8074	0.7076
Micro-Recall	0.8290	0.7766
Micro-F1score	0.8181	0.7405

Table 3: Results for Training Dataset

We discovered later on that that was due to a redundancy at a semantic level caused by polysemous synonyms present in our dictionary and used to enrich the list derived from training data. In developing the system with training data, we extracted synonym lists in order to adapt and contextualize them to the domain. Consider an important and frequent seed like POSIZIONE: it has a set of five different meanings in IWN (ItalianWordNet), (see Roventini et al. (2000)) which are then reflected in the extension of synonym lists covering all of them. So what we did was creating sublists adapted and limited to our domain. However, when we turned to analyse test data we decided to tune the seeds we regarded semantically unique, to the synonym lists without any previous adjustment. The result was a dramatic drop in performance when compared to training data.

Tasks /Results	ACD Run1	ACP Run1	ACD Run2	ACP Run2
Macro-P	0.5887	0.5277	0.5856	0.5241
Macro-R	0.6089	0.5661	0.6140	0.5699
Macro-F1	0.5986	0.5463	0.5994	0.5461
Micro-P	0.6232	0.5209	0.6164	0.5144
Micro-R	0.6093	0.5659	0.6134	0.5692
Micro-F1	0.6162	0.5425	0.6149	0.5404

Table 4: Published Results for Test Dataset

Tasks /Results	ACD Run1	ACP Run1	ACD Run2	ACP Run2
Macro-P	0.8222	0.7590	0.8258	0.7603
Macro-R	0.8458	0.7932	0.8564	0.8009
Macro-F1	0.8339	0.7757	0.8408	0.7801
Micro-P	0.7975	0.7033	0.7951	0.6986
Micro-R	0.8348	0.7880	0.8430	0.7938
Micro-F1	0.8157	0.7432	0.8183	0.7431

Table 5: Results for Test Dataset after Ablation Experiments

References

P. Basile, V. Basile, D. Croce, M. Polignano. 2018. *Overview of the EVALITA 2018 Aspect-based Sentiment Analysis (ABSITA) Task*. T. Caselli, N. Novielli, V. Patti, P. Rosso, (eds). In *Proceedings of the 6th evaluation campaign of Natural Language Processing and Speech tools for Italian (EVALITA'18)*. CEUR.org, Turin.

V. Basile, A. Bolioli, M. Nissim, V. Patti, P. Rosso. 2014. *Overview of the Evalita 2014 SENTIment POLarity Classification Task, Proceedings of EVALITA'14*. Edizioni PLUS, Pisa University Press, Pisa.

Brun, C. J. Perez, and C. Raux 2016. *XRCE at SemEval-2016 Task 5: Feedbacked Ensemble Modeling on Syntactico-Semantic Knowledge for Aspect Based Sentiment Analysis*. Proceedings of SemEval-2016, 277–281.

Rodolfo Delmonte. 2014a. *A Linguistic Rule-Based System for Pragmatic Text Processing*. Proceedings of Fourth International Workshop EVALITA 2014, 64-69. Edizioni PLUS, Pisa University Press, Pisa.

Rodolfo Delmonte. 2014b. *A Reevaluation of Dependency Structure Evaluation*. Proceedings of CLiC-it 2014 - The First Italian Conference on Computational Linguistics, 151-157. Edizioni PLUS, Pisa University Press, Pisa.

Esuli, A. and F. Sebastiani. 2006. *SentiWordnet: a publicly available lexical resource for opinion mining*. Proceedings of the 5th Conference on Language Resources and Evaluation LREC, 417–422.

Ohana, B. and B. Tierney and S.J. Delany. 2016. *Sentiment Classification Using Negation as a Proxy for Negative Sentiment*. Proceedings of 29th FLAIRS Conference, AAAI, 316-321.

Polanyi, Livia and Zaenen, Annie. 2006. *Contextual valence shifters*. Janyce Wiebe, editor, Computing Attitude and Affect in Text: Theory and Applications. Springer, Dordrecht, 1–10.

Roventini A., Alonge A., Calzolari N., Magnini B., Bertagna F.. 2014. *ItalWordNet: a Large Semantic Database for Italian*. Proceedings of LREC II, ELRA, 783-790.

APPENDIX
English Translation of all Italian examples in the paper

(1) cake/twisted, table, stop, plant, teaches/sign

(2) luggage storage, bed room, socket, fiscal receipt, breakfast room, historical center

(3) Small room but clean

(4) Some difficulties with parking notwithstanding staff helpfulness

(5) The position of the structure is a little out of the center, but in return it is close to the main bus stops

(6) Cleaning has a lot to be desired for a 4 star hotel

(7) In the bathroom only 1 schampoo and only 1 shower gel for 2 people

(8) Clean, spacious and what's more functional

(9) Many steps 93 lift small

(10) nothing to complain; all perfect

(11) all very nice and professional

(12) I liked practically all

(13) I liked all

(14) There is nothing that I didn't like

(15) However, a shuttle service for the city center would be convenient especially in the evening

(16) Great position, hotel in remaking this is why there's care for modernizing, great breakfast, fabulous wifi internet I made a video call with no problems and I was in my room, room and bath very spacious, gentle staff

(17) The large room and the comfortable bath, the WIFI service and the pleasant breakfast

(i) Definitely better the deluxe one that I have already taken other times and at lower costs

(ii) At 9 45 breakfast was rather scarse for the price of 15 Euros

Aspect-based Sentiment Analysis: X2Check at ABSITA 2018

Emanuele Di Rosa
Chief Technology Officer
App2Check s.r.l.
emanuele.dirosa
@app2check.com

Alberto Durante
Research Scientist
App2Check s.r.l.
alberto.durante
@app2check.com

Abstract

English. In this paper we describe and present the results of the two systems, called here X2C-A and X2C-B, that we specifically developed and submitted for our participation to ABSITA 2018, for the Aspect Category Detection (ACD) and Aspect Category Polarity (ACP) tasks. The results show that X2C-A is top ranker in the official results of the ACD task, at a distance of just 0.0073 from the best system; moreover, its post deadline improved version, called X2C-A-s, scores first in the official ACD results. About the ACP results, our X2C-A-s system, which takes advantage of our ready-to-use industrial Sentiment API, scores at a distance of just 0.0577 from the best system, even though it has not been specifically trained on the training set of the evaluation.

Italiano. *In questo articolo descriviamo e presentiamo i risultati dei due sistemi, chiamati qui X2C-A e X2C-B, che abbiamo specificatamente sviluppato per partecipare ad ABSITA 2018, per i task Aspect Category Detection (ACD) e Aspect Category Polarity (ACP). I risultati mostrano che X2C-A si posiziona ad una distanza di soli 0.0073 dal miglior sistema del task ACD; inoltre, la sua versione migliorata, chiamata X2C-A-s, realizzata successivamente alla scadenza, mostra un punteggio che lo posiziona al primo posto nella classifica ufficiale del task ACD. Riguardo al task ACP, il sistema X2C-A-s che utilizza il nostro standard Sentiment API, consente di ottenere un punteggio che dista solo 0.0577 dal miglior sistema, nonostante il classificatore di sentiment non sia stato specificamente addestrato sul training set della evaluation.*

1 Introduction

The traditional task of sentiment analysis is the classification of a sentence according to the positive, negative, or neutral classes. However, such task in this simple version is not enough to detect when a sentence contains a mixed sentiment, in which a positive sentiment is referred to one aspect and a negative sentiment to another aspect. Aspect-based sentiment analysis is focused on the sentiment classification (negative, neutral, positive) for a given aspect/category in a sentence. In nowadays world, reviews became an important tool widely used by consumers to evaluate services and products. Given the large amount of reviews available online, systems allowing to automatically classify reviews according to different categories, and assign a sentiment to each of those categories, are gaining more and more interest in the market. The former task is called Aspect Category Detection (ACD) since detects whether a review speaks about one of the categories under evaluation; the latter task, called Aspect Category Polarity (ACP) tries to assign a sentiment independently for each aspect. In this paper, we present X2C-A and X2C-B, two different implementations for dealing with the ACD and ACP tasks, specifically developed for the ABSITA evaluation (Basile et al., 2018). In particular, we describe the models used to participate to the ACD competition together with some post deadline results, in which we had the opportunity to improve our ACD results and evaluate our systems also on the ACP task. The resuls show that our X2C-A system is top ranking in the official ACD competition and scores first, in its X2C-A-s version. Moreover, by testing our ACD models on the ACP tasks, with the help of our standard X2Check sentiment API, the X2C-A-s system scores fifth at a

distance of just 0.057 from the best system, even if the other systems have a sentiment classifier specifically trained on the training set of the competition. This paper is structured as follow: after the introduction we present the descriptions of our two systems submitted to ABSITA and the results on the development set; then we show and discuss the results on the official testset of the competition for both ACD and ACP, finally we provide our conclusions.

2 Systems description

The official training dataset has been split into our internal training set (80% of the documents) and development set (the remaining 20%). We randomly sampled the examples for each category, thus obtaining different sets for training/test set, by keeping the per category distribution of the samples through the three sets. We submitted two runs, as the results of the two different systems we developed for each category, called X2C-A and X2C-B. The former has been developed on top of the Scikit-learn library in Python language (Pedregosa et al., 2011), and the latter on top of the WEKA library (Frank et al., 2016) in JAVA language. In both cases, the input text has been cleaned with a typical NLP pipeline, involving punctuation, numbers and stopwords removal. The two systems have been developed separately, but the best algorithms obtained by both the model selections are different implementations of Support Vector Machine. More details in the following sections.

2.1 X2C-A

The X2C-A system has been created by applying an NLP pipeline including a vectorization of the collection of reviews to a matrix of token counts of the bi-grams; then, the count matrix has been transformed to a normalized tf-idf representation (term-frequency times inverse document-frequency). As machine learning algorithm, an implementation of the Support Vector Machine has been used, specifically the LinearSVC. Such algorithm has been selected as the best performer on such dataset compared to other common implementations available in the sklearn library.

Table 1 shows the F1 score on the positive label in the development set for each category, where the average value on all of the categories is 84.92%. X2C-A shows the lowest performance on the Value category, while shows the best performance on Location, and high score on Wifi and Staff.

2.2 X2C-B

In the model selection process, the two best algorithms have been Naive Bayes and SMO. We built a model with both algorithms for each category. We took into account the F1 score on the positive labels in order to select the best algorithm. In this implementation, SMO (Sequential Minimal Optimization) (Platt, 1998) (Keerthi et al., 2001) (Hastie et al., 1998) has been the best performing algorithm on all of the categories, and showed an average F1 score across all categories of 85.08%. Its scores are reported in Table 1, where we also compare its performance with the X2C-A one on the development set.

The two systems are built on different implementation of support vector machines, as previously pointed out, and differ on the features extraction process. In fact, X2C-B takes into account a vocabulary of the 1000 most mentioned words in the training set, according to the size limit parameter available in the StringToWordVector Weka function. Moreover, it uses unigrams instead of the bi-grams extraction performed in X2C-A. The two systems reach similar results, i.e. high scores on Location, Wifi and Staff, and low scores on the Value category. However, the overall weighted performance is very close, around 85% of F1 on the positive labels, and since for some categories is better X2C-A and for others X2C-B, we decided to submit both implementations, in order to understand which is the best one on the test set of the ABSITA evaluation.

Category	X2C-A	X2C-B
Cleanliness	0.8675	**0.8882**
Comfort	**0.8017**	0.7995
Amenities	**0.8041**	0.7896
Staff	0.8917	**0.8978**
Value	**0.7561**	0.7333
Wifi	0.9056	**0.9412**
Location	**0.9179**	0.9058

Table 1: F1 score per category on the positive labels on the development set. Best system in bold.

3 Results on the ABSITA testset

3.1 Aspect Category Detection

Table 2 shows the official results of the Aspect-based Category Detection task, with the addition of two post deadline results obtained by an additional version of X2C-A and X2C-B, called X2C-A-s and X2C-B-s.

The difference between the submitted results and the versions called X2C-A-s and X2C-B-s, is just at prediction time: X2C-A and X2C-B make a prediction at document-level, i.e. on the whole review, while X2C-A-s and X2C-B-s make a prediction at sentence-level, where each sentence has been obtained by splitting the reviews on some punctuation and key conjunction words. This makes more likely that each sentence contains one category and it seems to be easier for the models the category detection. For example, the review

> *The sight is beautiful, but the staff is rude*

is about Location and Staff, but since only a part of it is about Location, the location model of this category would receive a document containing "noise" from its point of view. In the post deadline runs, we reduce the "noise" by splitting this example review in *The sight is beautiful* which is only about Location, and *but the staff is rude* which is only about Staff. As we can see in Table 2, the performance of X2C-A increased significantly and reached a performance score that is better even than the first classified. However, the performance of X2C-B slighted decreased in its X2C-B-s version. This means that the model of this latter system is not helped by this kind of "noise" removal technique. This last result shows that such approach does not have a general applicability but it depends on the model; however, it shows to work very well on X2C-A.

In order to identify the categories where we perform better, we calculated the score of our systems on each category[1], as shown in Table 3 and Table 4. In Table 3 X2C-A is the best of our systems on all the categories except Cleanliness and Wifi, where X2C-B has reached the higher score. In Table 4, X2C-A-s shows the best performance on all of the categories. By comparing the results across tables 3 and 4, we can see that X2C-A-s is the best system on all of the categories, with the exception of Cleanliness, where X2C-B shows a slightly better performance. Comparing the results on development set (Table 1) and the ones on the ABSITA test set, Value is confirmed being the most difficult category to detect for our systems, with a score of 0.6168. Instead, concerning Wifi, which has been the easiest category in Table 1, in Table 4 shows a lower relative score, while the easiest category to detect overall was Location, on which X2C-A-split has reached a score of 0.8898.

[1]To obtain these scores, we modified the ABSITA evaluation script so that only one category is taken into account.

Team	Mic-Pr	Mic-Re	Mic-F1
X2C-A-s	0.8278	**0.8014**	**0.8144**
1	0.8397	0.7837	0.8108
2	0.8713	0.7504	0.8063
3	0.8697	0.7481	0.8043
X2C-A	0.8626	0.7519	0.8035
5	0.8819	0.7378	0.8035
X2C-B	**0.8980**	0.6937	0.7827
X2C-B-s	0.8954	0.6855	0.7765
7	0.8658	0.697	0.7723
8	0.7902	0.7181	0.7524
9	0.6232	0.6093	0.6162
10	0.6164	0.6134	0.6149
11	0.5443	0.5418	0.5431
12	0.6213	0.433	0.5104
baseline	0.4111	0.2866	0.3377

Table 2: ACD results

	X2C-A	X2C-B
Cleanliness	0.8357	**0.8459**
Comfort	**0.794**	0.7475
Amenities	**0.8156**	0.7934
Staff	**0.8751**	0.8681
Value	**0.6146**	0.6141
Wifi	0.8403	**0.8667**
Location	**0.8887**	0.8681

Table 3: X2Check per category results submitted to ACD

3.2 Aspect Category Polarity

In Table 5 we show the results of the Aspect-based Category Polarity task to which X2Check did not formally participate. In fact, after the evaluation deadline we had time to work on the ACP task.

In order to deal with the ACP task, we decided to take advantage of our ready-to-use, standard

	X2C-A-s	X2C-B-s
Cleanliness	**0.8445**	0.8411
Comfort	**0.8099**	0.739
Amenities	**0.8308**	0.7884
Staff	**0.8833**	0.8652
Value	**0.6168**	0.6089
Wifi	**0.8702**	0.8667
Location	**0.8898**	0.8584

Table 4: X2Check per category results submitted post deadline to ACD.

X2Check sentiment API (Di Rosa and Durante, 2017). In fact, since we do have an industrial perspective, we realized that in a real world setting, the fact of training an Aspect-based sentiment system through a specific training set has a high effort associated and cannot have a general purpose application. In fact, a very common case is the one in which new categories to predict have to be quickly added into the system. In this setting, a high effort activity of labeling examples for the training set would be required. Moreover, labeling a review according to the aspects mentioned and additionally assign a sentiment to each aspect requires a higher human effort than just labeling the category. For this reason, we decided to not specifically train a sentiment predictor specialized on the given categories/aspects in the evaluation. Thus, we performed an experimental evaluation in which after the prediction of the category in the review, our standard X2Check sentiment API has been called to predict the sentiment. Since we are aware that a review may, in general, speak about multiple aspects and having different sentiment associated, we decided to apply the X2C-A-s and X2C-B-s versions which use the splitting method described in section 3.1. More specifically:

1. each review document has been split into sentences

2. both the X2Check sentiment API and the X2C-A/X2C-B category classifiers were run on each sentence. The former gives as output the polarity of each sentence; our assumption is that each portion of the review has a high probability to have just one sentiment associated. The latter gives as output all of the detected categories in each sentence

3. the overall result of a review is given by the collection of all of the category-sentiment pairs found in the sentences

The results shown in Table 5 show that our assumption is valid. In fact, despite being a single sentiment model for all of the categories, we reach the fifth place in the official ranking with our X2C-A-s system, at a distance of just 0.057 from the best system specifically trained on such training set. Furthermore, the ACP performance depends on the ACD results, in fact the former task cannot reach a performance higher than the other. For this reason, we decided to evaluate the sentiment performance reached on the reviews whose categories have been correctly predicted. Thus, we created a score capturing the relationship between the two results: it is the ratio between the micro F1 score obtained in the ACP task and the one obtained in the ACD task. This hand crafted score shows the quality of the sentiment model, by removing the influence of the performance on the ACD task. The overall sentiment score obtained is 88.0% for X2C-B and 87.1% for X2C-A, showing that even if a specific train has not been made, the general purpose X2Check sentiment API shows very good results (recall that, according to (Wilson et al., 2009) humans agree in the sentiment classification in the 82% of cases).

Team	**Mic-Pr**	**Mic-Re**	**Mic-F1**
1	0.8264	0.7161	**0.7673**
2	0.8612	0.6562	0.7449
3	0.7472	0.7186	0.7326
4	0.7387	**0.7206**	0.7295
X2C-A-s	0.7175	0.7019	0.7096
5	**0.8735**	0.5649	0.6861
X2C-B-s	0.7888	0.6025	0.6832
6	0.6869	0.5409	0.6052
7	0.4123	0.3125	0.3555
8	0.5452	0.2511	0.3439
baseline	0.2451	0.1681	0.1994

Table 5: ACP results.

Tables 6 and 7 show for each category the micro-F1 and the sentiment score of the ACP task, calculated like in Table 4, and the relationship between ACP and ACD scores per category. We can see that the sentiment model has reached a very good performance on Cleanliness, Comfort, Staff and Location since it is close or over the 90%. However, like noticed for the ACD results, it is

difficult to handle reviews about the Value category.

	Micro-F1	SS
Cleanliness	0.7739	91.6%
Comfort	0.7165	88.5%
Amenities	0.6618	79.7%
Staff	0.8086	91.5%
Value	0.4533	73.5%
Wifi	0.6615	76.0%
Location	**0.8168**	**91.8%**

Table 6: X2C-A ACP results and sentiment score by category.

	Micro-F1	SS
Cleanliness	0.7626	90.7%
Comfort	0.671	90.8%
Amenities	0.6276	79.6%
Staff	0.7948	91.9%
Value	0.4581	75.2%
Wifi	0.6441	74.3%
Location	**0.7969**	**92.8%**

Table 7: X2C-B ACP results and sentiment score by category.

4 Conclusions

In this paper we presented a description of two different implementations for dealing with the ACD and ACP tasks at ABSITA 2018. In particular, we described the models used to participate to the ACD competition together with some post deadline results, in which we had the opportunity to improve our ACD results and evaluate our systems also on the ACP task. The resuls show that our X2C-A system is top ranking in the official ACD competition and scores first, in its X2C-A-s version. Moreover, by testing our ACD models on the ACP tasks, with the help of our standard X2Check sentiment API, the X2C-A-s system scores fifth at a distance of just 0.057 from the best system, even if the other systems have a sentiment classifier specifically trained on the training set of the competition.

References

Pierpaolo Basile, Valerio Basile, Danilo Croce and Marco Polignano. 2018. *Overview of the EVALITA 2018 Aspect-based Sentiment Analysis task (ABSITA)* in Tommaso Caselli, Nicole Novielli, Viviana Patti, and Paolo Rosso, editors, Proceedings of the 6th evaluation campaign of Natural Language Processing and Speech tools for Italian (EVALITA'18), CEUR.org, Turin

Eibe Frank, Mark A. Hall, and Ian H. Witten. 2016. The WEKA Workbench. Online Appendix for "Data Mining: Practical Machine Learning Tools and Techniques", Morgan Kaufmann, Fourth Edition, 2016.

Pedregosa, F. and Varoquaux, G. and Gramfort, A. and Michel, V. and Thirion, B. and Grisel, O. and Blondel, M. and Prettenhofer, P. and Weiss, R. and Dubourg, V. and Vanderplas, J. and Passos, A. and Cournapeau, D. and Brucher, M. and Perrot, M. and Duchesnay, E. 2011. *Scikit-learn: Machine Learning in Python* in Journal of Machine Learning Research, pp. 2825–2830.

Emanuele Di Rosa and Alberto Durante. LREC 2016 *App2Check: a Machine Learning-based system for Sentiment Analysis of App Reviews in Italian Language* in Proc. of the 2nd International Workshop on Social Media World Sensors, pp. 8-11.

Emanuele Di Rosa and Alberto Durante. 2017. *Evaluating Industrial and Research Sentiment Analysis Engines on Multiple Sources* in Proc. of AI*IA 2017 Advances in Artificial Intelligence - International Conference of the Italian Association for Artificial Intelligence, Bari, Italy, November 14-17, 2017, pp. 141-155.

Sophie de Kok, Linda Punt, Rosita van den Puttelaar, Karoliina Ranta, Kim Schouten and Flavius Frasincar. 2018. *Review-aggregated aspect-based sentiment analysis with ontology features* in Prog Artif Intell (2018) 7: 295.

Theresa Wilson, Janyce Wiebe and Paul Hoffmann. 2009. *Recognizing Contextual Polarity: An Exploration of Features for Phrase-Level Sentiment Analysis* in Computational Linguistic, pp. 399–433.

Seth Grimes. 2010. *Expert Analysis: Is Sentiment Analysis an 80% Solution?* http://www.informationweek.com/software/information-management/expert-analysis-is-sentiment-analysis-an-80–solution/d/d-id/1087919

J. Platt. 1998. *Fast Training of Support Vector Machines using Sequential Minimal Optimization* in Advances in Kernel Methods - Support Vector Learning

S.S. Keerthi and S.K. Shevade and C. Bhattacharyya and K.R.K. Murthy. 2001. *Improvements to Platt's SMO Algorithm for SVM Classifier Design* in Neural Computation, volume 13, pp. 637-649.

Trevor Hastie and Robert Tibshirani 1998. *Classification by Pairwise Coupling* in Advances in Neural Information Processing Systems, volume 10.

Bidirectional Attentional LSTM for Aspect Based Sentiment Analysis on Italian

Giancarlo Nicola
University of Pavia
giancarlo.nicola01@universitadipavia.it

Abstract

English. This paper describes the SentITA system that participated to the ABSITA task proposed in Evalita 2018. The system is based on a Bidirectional Long Short Term Memory network with attention that exploits word embeddings and sentiment specific polarity embeddings. The model also leverages grammatical information from POS tagging and NER tagging. The system participated in both the Aspect Category Detection (ACD) and Aspect Category Polarity (ACP) tasks achieving the 5^{th} place in the ACD task and the 2^{nd} in the ACD task.

Italiano. *Questo paper descrive il sistema SentITA valutato nel task ABSITA proposto all'interno di Evalita 2018. Il sistema è basato su una rete nuerale ricorrente con celle di memoria di tipo Long Short Term Memory e con implementato un meccanismo d'attenzione. Il modello sfrutta sia word embeddings generali sia polarity embeddings specifici per la sentiment analysis ed inoltre fa uso delle informazioni derivanti dal POS-tagging e dal NER-tagging. Il sistema ha partecipato sia nella sfida di Aspect Category Detection (ACD) sia in quella di Aspect Category Polarity (ACP) posizionandosi al quinto posto nella prima e al secondo posto nella seconda.*

1 Introduction

This paper describes the SentITA system that participated to the ABSITA task (Basile et al. 2018) proposed in Evalita 2018. In ABSITA the task consists in performing Aspect Based Sentiment Analysis (ABSA) on self-reliant sentences scraped from the "booking.com" website. The aspects are related to the accommodation reviews and comprehend topics like cleanliness, comfort, location, etc. The task is divided in two subtasks Aspect Category Detection (ACD) and Aspect Category Polarity (ACP). The fist, ACD consists in identifying the aspects mentioned in the sentence, while the second requires to associate a sentiment polarity label to the aspects evoked in the sentence. Both the tasks are addressed with the same architecture and the same data preprocessing. The system is based on a deep learning model, a Bidirectional Long Short Term Memory network with attention. The model exploits word embeddings, sentiment specific polarity embeddings and it leverages also grammatical and information from POS tagging and NER tagging.

Recently, deep learning has emerged as a powerful machine learning technique achieving state-of-the-art results in many application domains, including sentiment analysis. Among the deep learning frameworks applied to sentiment analysis, many employ a combination of semantic vector representations (Mikolov et al. 2013), (Pennington et al. 2014) and different deep learning architectures. Long Short-Term Memory (LSTM) networks (Hochreiter and Schmidhuber 1997), (Socher et al. 2013), (Cho et al. 2014) have been applied to model complex and long term non-local relationships in both word level and character level text sequences. Recursive Neural Tensor Networks (RNTN) have shown great results for semantic compositionality (Socher et al. 2011), (Socher et al. 2013) and also convolutional networks (CNN) for both sentiment analysis (Collobert et al 2011) and sentence modelling (Kalchbrenner et al. 2014) have performed better than previous state of the art methodologies. All these methods in most of the applications receive in input a vector representation of words called word embeddings. (Mikolov 2012), (Mikolov et

al. 2013) and (Pennignton et al. 2014), further expanding the work on word embeddings (Bengio et al 2003), that grounds on the idea of distributed representations for symbols (Hinton et al 1986), have introduced unsupervised learning methods to create dense multidimensional spaces where words are represented by vectors. The position of such vectors is related to their semantic meaning and grammatical properties and they are widely used in all modern NLP. In fact, they allow for a dimensionality reduction compared to traditional sparse vectors space models and they are often used as pre-trained initialization for the first embedding layers of the neural networks in NLP tasks. In (Le and Mikolov 2014), expanding the previous work on word embeddings, is developed a model capable of representing also sentences in a dense multidimensional space. In this case too, sentences are represented by vectors whose position is related to the semantic content of the sentence with similar sentences represented by vectors that are close to each other.

When working with isolated and short sentences, often with a specific writing style, like tweets or phrases extracted from internet reviews many long term text dependencies are lost and not exploitable. In this situation it is important that the model learns both to pay attention to specific words that have key roles in determining the sentence polarity like negations, magnifiers, adjectives and to model the discourse but with less focus on long term dependencies (due to the text brevity). For this reason, deep learning word embedding based models augmented with task specific gazettes (dictionaries) and features, represent a solid baseline when working with these kind of datasets (Nakov et al. 2016)(Attardi et al. 2016)(Castellucci et al. 2016)(Cimino et al. 2016)(Deriu et al. 2016). In this system, a polarity dictionary for Italian has been included as feature to the model in addition to the word embeddings. Moreover every sentence during preprocessing is augmented with its NER tags and POS tags which then are used as features in the model. Thanks to the inclusion of these features relevant for the considered task in combination with word embeddings and an attentional bidirectional LSTM recurrent neural network, the model achieves useful results with some thousands labelled examples.

The remainder of the paper presents the model and the experiments on the ABSITA task. in Section 2 the model and its features are explained; in Section 3 the model training and its performances are discussed; in Section 4 a conclusion with the next improvement of the model is given.

2 Description of the system

The model implemented is an Attentional Bidirectional Recurrent Neural Network with LSTM cells. It operates at word level and therefore each sentence is represented as a sequence of words representations that are sequentially fed to the model one after another until the sequence has been entirely used up. One sentence sequence coupled with its polarity scores represent a single datapoint for the model.

The input to the model are sentences, represented as sequence of word representations. The maximum sequence length has been set to 35, with shorter sentences left-padded to this length and longer sentences cut to this length. Each word of the sequence is represented by five vectors corresponding to 5 different features that are: high dimensional word embedding, word polarity, word NER tag, word POS tag, custom low dimensional word embedding. The high dimensional word embeddings are the pretrained Fastext embeddings for Italian (Grave et al. 2018). They are 300-dimensional vectors obtained using the skip-gram model described in (Bojanowski et al. 2016) with default parameters. The word polarity is obtained from the OpeNER Sentiment Lexicon Italian (Russo et al. 2016). This freely available Italian Sentiment Lexicon contains a total of 24.293 lexical entries annotated for positive/negative/neutral polarity. It was semi-automatically developed using a propagation algorithm starting from a list of seed key-words and manually reviewing the most frequent.

Both the NER tags and POS tags are obtained from the Spacy library Tagger model for Italian (Spacy 2.0.11 - https://spacy.io/). The custom low dimensional word embeddings are generated by random initialization and are inserted to provide an embedding representation of the words that are missing from the Fastext embeddings, which otherwise would all be represented by the same out of vocabulary token (OOV token). In general, it could be possible to train and fine-tune these custom embeddings on specific datasets to let the model learn the usage of words in specific cases. The information extracted from the OpeNER Sen-

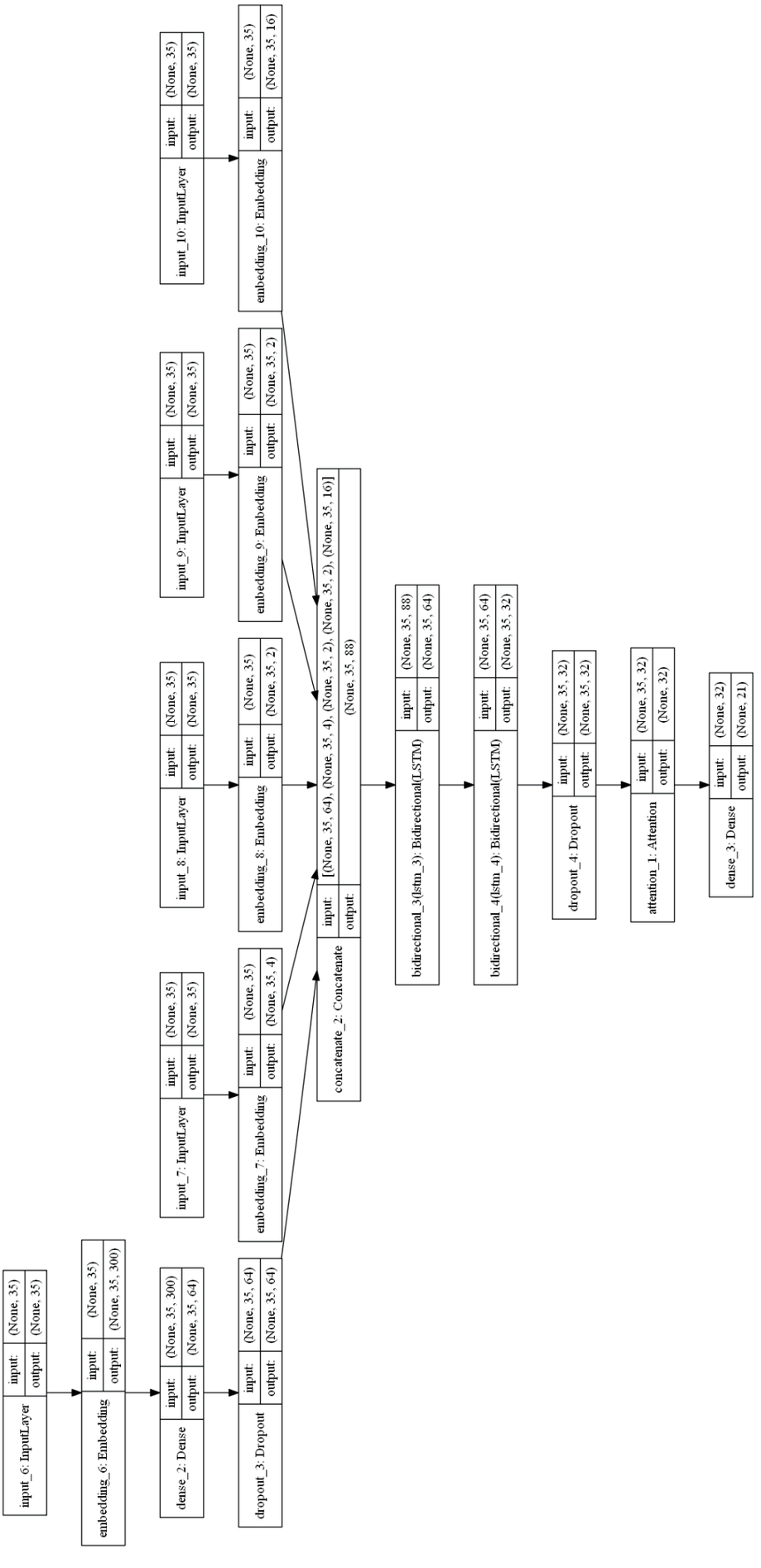


Figure 1: Model architecture

timent Lexicon Italian are the word polarity with its confidence and they are concatenated in a vector of length 2 that is one of the input to the first layer of the network. The NER tags and POS tags instead are mapped to randomly initialized embeddings of dimensionality respectively 2 and 4 that are not trained during the model training for the task submission. With more data available it would probably be beneficial to train all the NER, POS and custom embeddings but for this specific dataset the results were comparable and slightly better when not training the embeddings.

The model, whose architecture is schematized in fig. 1, performs in its initial layer a dimensionality reduction on the Fastext embeddings and then concatenates them with the rest of the embeddings (polarity, NER tag, POS tag, and custom word embeddings) for each each timestep (word) of the sequence. The concatenation of the embeddings is fed in a sequence of two bidirectional recurrent layers with LSTM cells. The result of these recurrent layers is passed to the attention mechanism presented in (Raffel et al. 2016) and finally to the dense layers that outputs the aspect detection and aspect polarity signals. The attention mechanism in this formulation, produces a fixed-length embedding of the input sequence by computing an adaptive weighted average of the sequence of states (normally denoted as "h") of the RNN. This form of integration is similar to the "global temporal pooling" described in (Sander 2014), which is based on the "global average pooling" technique of (Min et al. 2014). The non linear activations used in the model are Rectified Linear Units (ReLU) for the internal dense layers, hyperbolic tangent (tanh) in the recurrent layers and sigmoid in the output dense layer. In order to contrast overfitting the dropout mechanism has been used after the Fastext embedding dimensionality reduction with rate 0.5, in both the recurrent layers between each timestep with rate 0.5 and on the output of the recurrent layers with rate 0.3.

The model has 61,368 trainable parameters and a total of 45,233,366 parameters, the majority of them representing the Fastext embedding matrix (45,000,300). Compared to many NLP models used today the number of trainable parameters is quite small to reduce the possibility of overfitting the training dataset (6,337 examples is small compared to many English sentiment datasets) and also because is compensated by the addition of engineered features like polarity dictionary, NER tag and POS tag that help in classifying the examples.

3 Training and results

The only preprocessing applied to the text is the conversion of each character to its lower case form. Then, the vocabulary of the model is limited to the first 150,000 words of the Fastext embeddings trough a cap on the max number of embeddings, due to memory constraints of the GPU used for training the model. The Fastext embeddings are sorted by descending frequency of appearance in their training corpus, thus the vocabulary comprises the 150,000 most frequent words in Italian. The other words that remain out of this cut are represented in the model high dimensional embeddings (Fastext embeddings) by an out of vocabulary token. However, all the training set words are anyhow included in the custom low dimensional word embeddings; this is done since both our training text and in general users text (specially when working with reviews, tweets, social network platforms) could be quite different from the one on which Fastext embeddings are trained. In addition the NER-tagging and POS-tagging models for Italian included in the Spacy library are applied to the text to compute the additional NER-tags and POS-tags features for each word.

To train the model and generate the challenge submission a k-fold cross validation strategy has been applied. The dataset has been divided in 5 folds and 5 different instantiations of the same model (with the same architecture) have been trained picking each time a different fold as validation set (20%) and the remaining 4 folds as training set (80%). The number of training epochs is defined with the early stopping technique with patience parameter equal to 7. Once the training epochs are completed, the model snapshot that achieved the best validation loss is loaded. At the end the predictions from the 5 models have been averaged together and thresholded at 0.5. The training of five different instantiations of the same model and the averaging of their predictions overcomes the fact that in each k^{th}-fold the model selection based on the best validation loss is biased on the validation fold itself.

Each of the five models is trained minimizing the crossentropy loss on the different classes with the Nesterov Adam (Nadam) optimizer (Dozat

Ranking	Micro Precision	Micro Recall	Micro F1-score
1	0.8397	0.7837	0.8108
2	0.8713	0.7504	0.8063
3	0.8697	0.7481	0.8043
4	0.8626	0.7519	0.8035
5	0.8819	0.7378	0.8035
6	0.898	0.6937	0.7827
7	0.8658	0.697	0.7723
8	0.7902	0.7181	0.7524
9	0.6232	0.6093	0.6162
10	0.6164	0.6134	0.6149
11	0.5443	0.5418	0.5431
12	0.6213	0.433	0.5104
baseline	0.4111	0.2866	0.3377

Table 1: Task ACD (Aspect Category Detection) ranking. This system score is reported between dashed lines

2015) with default parameters ($\lambda = 0.002$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, schedule_decay $= 0.004$). The Nesterov Adam optimizer is similar to the Adam optimizer (Kingma et al. 2014) but were momentum is replaced with Nesterov momentum (Nesterov 1983). Adam in fact, combines two algorithms known to work well for different reasons: momentum, which points the model in a better direction, and RMSProp, which adapts how far the model goes in that direction on a per-parameter basis. However, Nesterov momentum which can be viewed as a simple modification of the former, increases stability, and can sometimes provide a distinct improvement in performance, superior to momentum (Sutskever et al. 2013). For this reason the two approaches are combined in the Nadam optimizer.

This system obtained the 5^{th} place in the ACD and the 2^{nd} place in the ACP task as reported respectively in Table 1 and Table 2. In these tables the performances of the systems participating to the challenge have been ranked by F1-score from the task organizers. In particular, it is interesting the second place in the ACP since the model is more oriented towards polarity classification for which it has specific dictionaries more than aspect detection. This is confirmed also from the high precision score obtained from the model in the ACP task, the 2^{nd} highest among the participating systems.

4 Discussion

The results obtained by the SentITA system at ABSITA 2018 are promising, as the system placed 2^{nd} in the ACP and 5^{th} in the ACD task but not very far from the 1^{st} in terms of F1-score. The model in general shows a high precision but in general a lower recall compared to the other systems. The proposed architecture makes use of different features that is easy to obtain through other models like POS and NER tags, polarity and word embeddings, for this reason, the human effort in the data preprocessing is very limited. One important direction to further improve the model would be to rely more on unsupervised learning, which at the moment is used only for the word embeddings. It could be possible to integrate in the model features based on language models or encoder-decoder networks, for example. More unsupervised learning would better ensure the model generalization to cover most of the argument and lexical content of the Italian language due to the large quantity of text available and thus improving also the model recall.

Ranking	Micro Precision	Micro Recall	Micro F1-score
1	0.8264	0.7161	0.7673
2	0.8612	0.6562	0.7449
3	0.7472	0.7186	0.7326
4	0.7387	0.7206	0.7295
5	0.8735	0.5649	0.6861
6	0.6869	0.5409	0.6052
7	0.4123	0.3125	0.3555
8	0.5452	0.2511	0.3439
baseline	0.2451	0.1681	0.1994

Table 2: Task ACP (Aspect Category Polarity) ranking. This system score is reported between dashed lines

References

Giuseppe Attardi, Daniele Sartiano, Chiara Alzetta, Federica Semplici. 2016. Convolutional Neural Networks for Sentiment Analysis on Italian Tweets. CLiC-it/EVALITA (2016).

Pierpaolo Basile and Valerio Basile and Danilo Croce and Marco Polignano. 2018. Overview of the EVALITA 2018 Aspect-based Sentiment Analysis task (ABSITA). Proceedings of the 6th evaluation campaign of Natural Language Processing and Speech tools for Italian (EVALITA'18), CEUR.org, Turin

Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin (2003) A neural probabilistic language model. The Journal of Machine Learning Research, 3:1137–1155, 2003.

P. Bojanowski, E. Grave, A. Joulin, T. Mikolov (2016) Enriching Word Vectors with Subword Information. arXiv:1607.04606v2

Giuseppe Castellucci, Danilo Croce, Roberto Basili. 2016. Context–aware Convolutional Neural Networks for Twitter Sentiment Analysis in Italian. CLiC-it/EVALITA (2016).

K. Cho, B. van Merrienboer, C. Gulcehre, F. Bougares, H. Schwenk, and Y. Bengio. (2014) Learning phrase representations using RNN encoder-decoder for statistical machine translation. In EMNLP, 2014.

Andrea Cimino, Felice Dell'Orletta. 2016. Tandem LSTM–SVM Approach for Sentiment Analysis. Castellucci, Giuseppe et al. CLiC-it/EVALITA (2016).

R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu and P. Kuksa. Natural Language Processing (Almost) from Scratch. Journal of Machine Learning Research, 12:2493- 2537, 2011.

Jan Deriu, Mark Cieliebak. 2016. Sentiment Detection using Convolutional Neural Networks with Multi–Task Training and Distant Supervision. CLiC-it/EVALITA (2016).

Timothy Dozat (2015) Incorporating Nesterov Momentum into Adam. http://cs229.stanford.edu/proj2015/054_report.pdf.

E. Grave*, P. Bojanowski*, P. Gupta, A. Joulin, T. Mikolov (2018) Learning Word Vectors for 157 Languages. Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)

G. E. Hinton, J. L. McClelland, and D. E. Rumelhart (1986) Distributed representations. In Rumelhart, D. E. and McClelland, J. L., editors, Parallel Distributed Processing: Explorations in the Microstructure of Cognition. 1986. Volume 1: Foundations, MIT Press, Cambridge, MA. pp 77-109.

S. Hochreiter, J. Schmidhuber. Long Short-Term Memory. Neural Computation 9(8):1735-1780, 1997

N. Kalchbrenner, E. Grefenstette, P. Blunsom. (2014) A Convolutional Neural Network for Modelling Sentences. In Proceedings of ACL 2014.

Kingma, Diederik and Ba, Jimmy. (2014). Adam: A Method for Stochastic Optimization. International Conference on Learning Representations. https://arxiv.org/pdf/1412.6980.pdf

Q. Le, T. Mikolov. Distributed Representations of Sentences and Documents. Proceedings of the 31 st International Conference on Machine Learning, Beijing, China, 2014. JMLR: W&CP, volume 32.

T. Mikolov. (2012) Statistical Language Models Based on Neural Networks. PhD thesis, PhD Thesis, Brno University of Technology, 2012.

T. Mikolov, K. Chen, G. Corrado, and J. Dean. (2013) Efficient estimation of word representations in vector space. In Proceedings of Workshop at International Conference on Learning Representations, 2013.

Min Lin, Qiang Chen, and Shuicheng Yen. Network in network. arXiv preprint arXiv:1312.4400, 2014.

Preslav Nakov, Alan Ritter, Sara Rosenthal, Fabrizio Sebastiani, Veselin Stoyanov. 2016. SemEval-2016 Task 4: Sentiment Analysis in Twitter. Proceedings of SemEval-2016, pages 1–18, San Diego, California, June 16-17, 2016.

Y. Nesterov (1983) A method of solving a convex programming problem with convergence rate o (1/k2). In Soviet Mathematics Doklady, volume 27, pages 372-376, 1983.

J. Pennington, R. Socher, and C. Manning. (2014) Glove: Global vectors for word representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1532–1543, Doha, Qatar, October. Association for Computational Linguistics.

Colin Raffel, Daniel P. W. Ellis (2016) Feed-Forward Networks with Attention Can Solve Some Long-Term Memory Problems. https://arxiv.org/abs/1512.08756

Russo, Irene; Frontini, Francesca and Quochi, Valeria, 2016, OpeNER Sentiment Lexicon Italian - LMF, ILC-CNR for CLARIN-IT repository hosted at Institute for Computational Linguistics "A. Zampolli", National Research Council, in Pisa, http://hdl.handle.net/20.500.11752/ILC-73.

Sander Dieleman. Recommending music on Spotify with deep learning. http://benanne.github.io/2014/08/05/spotify-cnns.html, 2014.

R. Socher, J. Pennington, E. H. Huang, A. Y. Ng, and Christopher D. Manning. (2011) Semi-Supervised Recursive Autoencoders for Predicting Sentiment Distributions. In Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP).

R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and Christopher Potts. (2013) Recursive deep models for semantic compositionality over a sentiment treebank. In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1631–1642, Stroudsburg, PA, October. Association for Computational Linguistics.

Ilya Sutskever, James Martens, George Dahl, Geoffrey Hinton (2013) Proceedings of the 30th International Conference on Machine Learning, PMLR 28(3):1139-1147, 2013.

Ensemble of LSTMs for EVALITA 2018 Aspect-based Sentiment Analysis task (ABSITA)

Mauro Bennici
You Are My GUide
mauro@youaremyguide.com

Xileny Seijas Portocarrero
You Are My GUide
xileny@youaremyguide.com

Abstract

English. In identifying the different emotions present in a review, it is necessary to distinguish the single entities present and the specific semantic relations. The number of reviews needed to have a complete dataset for every single possible option is not predictable.

The approach described starts from the possibility to study the aspect and later the polarity and to create an ensemble of the two models to provide a better understanding of the dataset.

Italiano. Nell'identificazione delle diverse emozioni presenti in una recensione è necessario distinguere le singole entità presenti e le singole relazioni semantiche. Il numero di recensioni necessarie per avere un dataset completo per ogni singola opzione possibile non è predicibile.

L'approccio descritto parte dalla possibilità di creare due modelli diversi, uno per la parte di categorizzazione, e l'altro per la parte di polarità. E di unire i due modelli per ottenere una maggiore comprensione del dataset.

1 Introduction

With the increase in interactions between users and businesses across different channels and different languages, it becomes increasingly difficult for businesses to respond promptly and effectively in an effective manner. Not all activities can have a team dedicated to public relations and often rely on external agencies that do not know the internal operations of the company.

Automating the correct recognition of the various problems can lead to the timely addressing of the same to the persons appointed to solve them.

The research was carried out with the dataset provided within the task called ABSITA, Aspect-based Sentiment Analysis at EVALITA 2018[1] (Basile et al., 2018). The task was a combination of two tasks, Aspect Category Detection (ACD) and Aspect Category Polarity (ACP).

The dataset is a selection of hotel reviews taken in Italian from the portal Booking.com.

2 Description of the system

Each review has been cleaned up by special characters, lemmatized and brought to lowercase with the SpaCy[2] framework.
Generic Italian texts have been used, instead of reviews in the accommodation context to be sure that the model will be suitable for more business models, to generate vectors in fastText[3]. The best one has a dimension of 200, with character n-grams of length 5, a window of size 5 and 10 negatives.

The system is the ensemble of two different models to improve the ability to discover hidden properties (Akhtar et al., 2018).

The first model is a bi-directional Long Short-Term Memory (BI-LSTM).

[1] http://sag.art.uniroma2.it/absita/
[2] https://spacy.io
[3] https://fasttext.cc/

This model is used for the discernment of the ASPECT.

Layer (type)	Output Shape	Param #
e (Embedding)	(None, 100, 200)	1420400
b (Bidirection)	(None, 512)	935936
d (Dense)	(None, 7)	3591

A second BI-LSTM model is used for the discernment of POLARITY.

Layer (type)	Output Shape	Param #
e (Embedding)	(None, 100, 200)	1420400
b (Bidirection)	(None, 512)	935936
d (Dense)	(None, 14)	7182

A dropout and a recurrent_dropout of 0.1.
The optimizer for both is the RMSProp.
The loaded embedding is trainable.
Both the systems use Keras[4] to create the RNN models.

The models were trained and tested with a 5-fold cross-validation with a ratio of 80% training and 20% testing. The best model was automatically saved at each iteration.

A threshold of 0.5 was used on the first model to activate the result of the last layer. In the second model, the threshold was of 0.43.

Aspect Category Detection (ACD)		
micro precision	micro recall	micro F1 score
0.8397	0.8050	0.8204

Table 1: micro precision, micro recall and micro F1 score with the gold dataset.

Aspect Category Polarity (ACP)		
micro precision	micro recall	micro F1 score
0.8138	0.6593	0.7172

Table 2: micro precision, micro recall and micro F1 score with the gold dataset.

The results show that the models are useful to understand the category of a review better than its polarity.

After that we ensemble the two models (Choi et al., 2018) to obtain a system able to overcome the results of every single model in the ACP task reducing the result on the ACD task (table 3).

The ensemble has been created in cascade making sure that a system acts as Attention to the underlying system.
The threshold of activation was a range between 0.45 and 0.55.

A third model, a LightGBM[5] (Bennici and Portocarrero, 2018) was also tested, where the following properties are extracted from the reviews text:

- length of the review
- percentage of special characters
- the number of exclamation points
- the number of question marks
- the number of words
- the number of characters
- the number of spaces
- the number of stop words
- the ratio between words and stop words
- the ratio between words and spaces

and they are joined to the vector created by the bigram and trigram of the text itself at word and character level.
The number of leaves is 250, the learner set as 'Feature', and a the learning rate at 0.04.

The result of the union between the three models could not be submitted to the final evaluation, due to the limit of 2 possible submissions, but reported results higher than 83% in the tests car-

[4] https://keras.io

[5] https://github.com/Microsoft/LightGBM

ried out after the release of the complete dataset for ASPECT and 75% for POLARITY.
Also, the inference is faster than the RNN models.

3 Results

Aspect Category Detection (ACD)			
Runs	micro precision	micro recall	micro F1
Run 1	0.8713	0.7504	0.8063
Run 2	0.8697	0.7481	0.8043

Table 3: micro precision, micro recall and micro F1 score for the submitted ACD subtasks.

Aspect Category Polarity (ACP)			
Runs	micro precision	micro recall	micro F1
Run 1	0.7387	0.7206	0.7295
Run 2	0.7472	0.7186	0.7326

Table 4: micro precision, micro recall and micro F1 score for the submitted ACP subtask.

In the evaluation phase, we can see how the results have given reason to the ensemble of the two results.

It is clear that the ACP task (table 4) is the beneficiary of this process, instead of the ACD one (table 3) that lost more than one point.

The study of the dataset is influenced by the little extension of the training dataset and by the specificity of some terms that could refer to different categories such as the comfort of the room and the quality/price ratio.

Various types of data preparation have also been used, including the preservation of special characters, the shape of words (to better identify cities or places written in capital letters), and some SMOTE functions to increase the number of entries but with poor results and noticeable overfitting.

4 Conclusion

Creating an ensemble of models to bring out various properties of a review gave better results than using a single model in the polarity identification.

The terms used in the review are sometimes misleading and can be used both positively or negatively, and to identify different categories of the hotel.

In the near future, we are ready to create a system to split the text of the review to categorize only a single sentence, or less a single subject or object. In this way, we will be ready to evaluate also the polarity of the single object or subject, and only the terms single related to it to improve the result of the ACP task.

The performance of the system will also be evaluated by replacing all the possible entities with variables known as:

- City
- Museum
- Panoramic Point
- Railway station
- Street

and with a pre-category knew a priori as Breakfast for words like Coffee, Cornetto, and Jam.

The expected result is to reduce the variance of the dataset, to improve the ACD result, and to be able to use the system in production.

Finally, we will evaluate the speed and effectiveness of a CNN model in which the tasks, ASPECT, and POLARITY, can be studied separately and then merged.

Reference

Basile, P., Basile, V., Croce, D., & Polignano, M. (2018). Overview of the EVALITA 2018 Aspect-based Sentiment Analysis task (ABSITA). Proceedings of the 6th evaluation campaign of Natural Language Processing and Speech tools for Italian (EVALITA'18)

Akhtar, M., Ghosal, D., Ekbal, A., Bhattacharyya, P., & Kurohashi, S. (2018, October 15). A Multi-task Ensemble Framework for Emotion, Sentiment and

Intensity Prediction. Retrieved from https://arxiv.org/abs/1808.01216

Choi, J. Y. and Bumshik, L. (2018).“Combining LSTM Network Ensemble via Adaptive Weighting for Improved Time Series Forecasting,” Mathematical Problems in Engineering, vol. 2018, Article ID 2470171, 8 pages. doi: https://doi.org/10.1155/2018/2470171.

Bennici, M. and Seijas Portocarrero, X. (2018). The validity of dictionaries over the time in Emoji prediction. In Tommaso Caselli, Nicole Novielli, Viviana Patti, and Paolo Rosso, editors, Proceedings of the 6th evaluation campaign of Natural Language Processing and Speech tools for Italian (EVALITA’18), Turin, Italy. CEUR.org.

Fully Convolutional Networks for Text Classification

Jacob Anderson
Sentim LLC
Columbus, OH, USA
papers@sentimllc.com

Abstract

English. In this work I propose a new way of using fully convolutional networks for classification while allowing for input of any size. I additionally propose two modifications on the idea of attention and the benefits and detriments of using the modifications. Finally, I show suboptimal results on the ITAmoji 2018 tweet to emoji task and provide a discussion about why that might be the case as well as a proposed fix to further improve results.

Italian. In questo lavoro viene presentato un nuovo approccio all'uso di fully convolutional network per la classificazione, adattabile a dati di input di qualsiasi dimensione. In aggiunta vengono proposte due modifiche basate sull'uso di meccanismi di attention, valutandone benefici e svantaggi. Infine, sono presentati i risultati della partecipazione al Task ITAmoji 2018 relativo alla predizione di emoji associate al testo di tweets, discutendo il perché delle performance non ottimali del sistema sviluppato e proponendo possibili migliorie.

1 Introduction

The dominant approach in many natural language tasks is to use recurrent neural networks or convolutional neural networks (CNN) (Conneau et al., 2017). For classification tasks, recurrent neural networks have a natural advantage because of their ability to take in any size input and output a fixed size output. This ability allows for greater generalization as no data is removed nor added in order for the inputs to match in length. While convolutional neural networks can also support input of any size, they lack the ability to generate a fixed size output from any sized input. In text classification tasks, this often means that the input is fixed in size in order for the output to also have a fixed size.

Other recent work in language understanding and translation uses a concept called attention. Attention is particularly useful for language understanding tasks as it creates a mechanism for relating different position of a single sequence to each other (Vaswani et al., 2017).

In this work I propose a new way of using fully convolutional networks for classification to allow for any sized input length without adding or removing data. I also propose two modifications on attention and then discuss the benefits and detriments of using the modified versions as compared to the unmodified version.

2 Model Description

The overall architecture of my fully convolutional network design is shown in Figure 1. My model begins with a character embedding where each character in the input maps to a vector of size 16. I then first apply a causal convolution with 128 filters of size 3. After which, I apply a stack of 9 layers of residual dilated convolutions with skip connections, each of which use 128 filters of size 7. The size of 7 here was chosen by inspection, as it converged faster than size 3 or 5 while not consuming too much memory. Additionally, the dilation rate of each layer of the stack doubles for every layer, so the first layer has rate 1, then the second layer has rate 2, then rate 4, and so on.

All of the skip connections are combined with a summation immediately followed by a ReLU to increase nonlinearity. Finally, the output of the network was computed using a convolution with 25 filters each of size 1, followed by a global max pool operation. The global max pool operation reduces the 3D tensor of size (batch size, input length, 25) to (batch size, 25) in order to match the expected output.

I implemented all code using a combination of Tensorflow (Abadi et al., 2016) and Keras (Chollet, 2015). During training I used softmax cross-entropy loss with an l2 regularization penalty with a scale of .0001. I further reduced overfitting by adding spatial dropout (Tompson et al., 2015) with a drop probability of 10% in the residual dilated convolution layers.

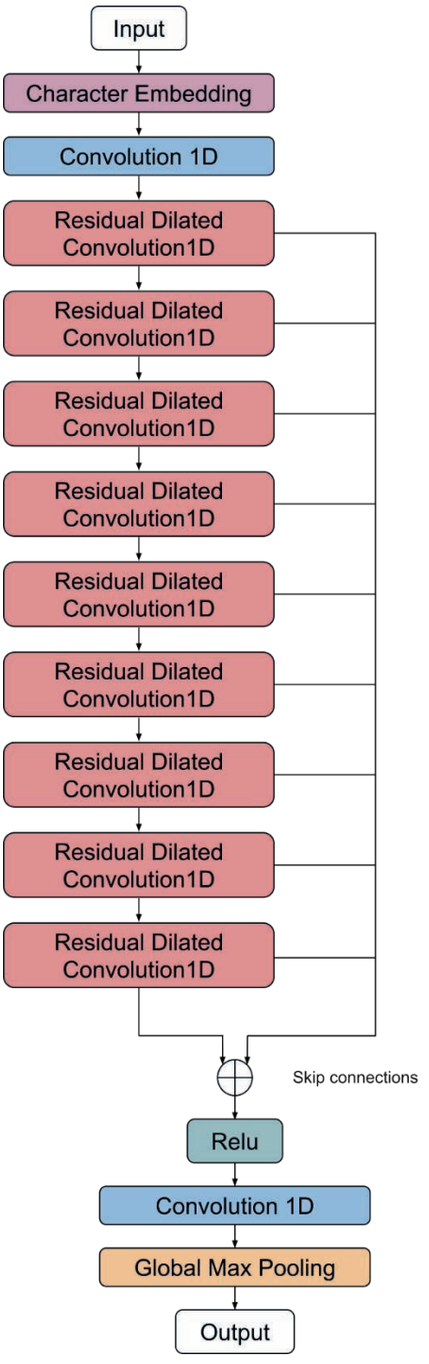


Figure 1: Model Architecture

[1] They have since changed this limitation to 13 GB.

2.1 Hardware Limitations

At the time of creating the models in this paper, I was limited to only a Google Colab GPU, which comes with a runtime restriction of 12 hours per day and a half a GB of GPU memory[1]. While it is possible to continue training again after the restriction is reset, in order to maximize GPU usage, I tried to design each iteration of the model so that it would finish training within a 12 hour time period.

2.2 Residual Block

A residual connection is any connection which maps the input of one layer to the output of a layer further down in the network. Residual connections decrease training error, increase accuracy, and increase training speed (He et al., 2015).

2.3 Dilated Convolution

A dilated convolution is a convolution where the filter is applied over a larger area by skipping input values according to a dilation rate. This rate usually exponentially scales with the numbers of layers of the network, so you would look at every input for the first layer and then every other input for the second, and then every fourth and so on (van den Oord, 2016).

In this paper, I use dilated convolutions similar to Wavenet (van den Oord, 2016), where each convolution has both residual and skip connections. However, instead of the gated activation function from the Wavenet paper, I used local response normalization followed by a ReLU function. This activation function was proposed by Krizhevsky, Sutskever, and Hinton (2012), and I used it because I found this method to achieve equal results but faster convergence.

2.4 Residual Dilated Convolution

A residual dilated convolution is a dilated convolution with a residual connection. First, I take a dilated convolution on the input and a linear projection on the input. The dilated convolution and the linear projection are added together and then outputted. The dilated convolution also outputs as a skip connection, which is eventually summed together with every other skip connection later in the network.

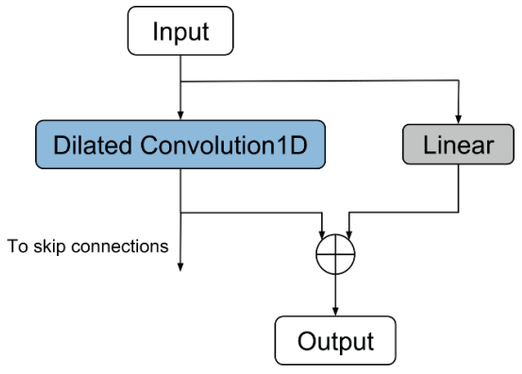


Figure 2: Residual Dilated Convolution

2.5 Skip Connections

In this paper, I also use the idea of skip connections from Long, Shelhamer, and Darrell (2015). Skip connections simply connect previous layers with the layer right before the output in order to fuse local and global information from across the network. In this work, the connections are all fused together with a summation followed by a ReLU activation to increase nonlinearity.

2.6 Attention and Self-Attention

Attention can be described as mapping a query and a set of key value pairs to an output (Vaswani et al., 2017). Specifically, when I say attention or 'normal' attention, I am referring to Scaled Dot-Product Attention. Scaled Dot-Product Attention is computed as:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (1)$$

Where Q, K, and V are matrices representing the queries, keys, and values respectively (Vaswani et al., 2017).

Self-Attention then is where Q, K, and V all come from the same source vector after a linear projection. This allows each position in the vector to attend to every other position in the same vector.

2.7 Simplified and Local Attention

Simplified and local attention can both be thought of as trying to reinforce the mapping of a key to value pair by extracting extra information from the key. I compute a linear transformation followed by a softmax to get the weights on the values. These weights and the initial values are multiplied together element-wise in order to highlight which of the values are the most important for solving the problem. Simplified attention can also be thought of as reinforcing a one-to-one correspondence between the key and the value.

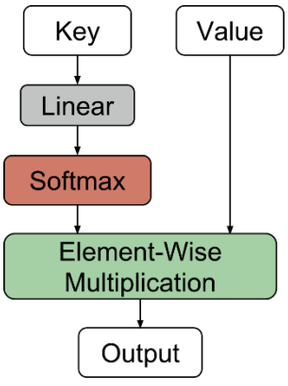


Figure 3: Simplified Attention

Local attention is like simplified attention except instead of performing a linear projection on the keys, local attention performs a convolutional projection on the keys. This allows for the network to use local information in the keys to attend to the values.

2.8 Multi-Head Attention

In multi-head attention, attention is performed multiple times on different projections of the input (Vaswani et al., 2017). In this paper, I either use one or eight heads in every experiment with attention, in order to get the best results and to compare the different methods accurately.

2.9 Model Modifications for Attention

In this paper, I tested seven different models, six of which extend the base model using some type of attention. In the models with attention, self-attention is used right after the final convolution and right before the global pooling operation.

2.10 Global Max Pooling

While CNN's support input of any size, they lack the ability to generate a fixed size input and instead output a tensor that is proportional in size to the input size. In order for the output of the network to have a fixed size of 25, I use max pooling (Scherer et al., 2010) along the time dimension of the last convolutional layer. I perform the max pooling globally, which simply means that I take the maximum value of the whole time dimension instead of from a sliding window of the time dimension.

3 Experiment and Results

In this section, I go over the ITAmoji task description and limitations, as well as my results on the task.

3.1 ITAmoji Task

This model was initially designed for the ITAmoji task in EVALITA 2018 (Ronzano et al., 2018). The goal of this task is to predict which of 25 emojis (shown in Table 1) is most likely to be in a given Italian tweet. The provided dataset is 250,000 Italian tweets with one emoji label per tweet, and no additional data is allowed for training the models. However, it is allowed to use additional data to train unsupervised systems like word embeddings. All results in the coming subsections were tested on the dataset of 25,000 Italian tweets provided by the organizers.

Emoji	Label	% Samples
❤️	red heart	20.28
😂	face with tears of joy	19.86
😍	smiling face with heart eyes	9.45
💋	kiss mark	1.12
😉	winking face	5.35
😊	smiling face with smiling eyes	5.13
😁	beaming face with smiling eyes	4.11
😀	grinning face	3.54
😘	face blowing a kiss	3.34
😎	smiling face with sunglasses	2.80
👍	thumbs up	2.57
🤣	rolling on the floor laughing	2.18
🤔	thinking face	2.16
💙	blue heart	2.02
😜	winking face with tongue	1.93
😱	face screaming in fear	1.78
💪	flexed biceps	1.67
😋	face savoring food	1.55
😅	grinning face with sweat	1.52
😭	loudly crying face	1.49
🔝	top arrow	1.39
💕	two hearts	1.36
☀️	sun	1.28
🌹	rose	1.06
✨	sparkles	1.06

Table 1: Each of the 25 different emojis used in the ITAmoji task, their labels, and the corresponding percent of samples in the test dataset.

3.2 Results

Table 2 shows my official results from the ITAmoji competition, as well as the first and second group scores. Table 3 shows the best result (evaluated after the competition was complete) according to the macro f1 score of the seven different models I trained during the competition. It also shows the micro f1 score at the same run of the best macro f1 score for comparison. Table 4 shows the upper and lower bounds of the f1 scores after the scores have stopped increasing and have plateaued.

Model	Macro F1	Micro F1
1st Place Group	0.365	0.477
2nd Place Group	0.232	0.401
Run 3: Simplified Attention	0.106	0.294
Run 2: 1 Head Attention	0.102	0.313
Run 1: No Attention[2]	0.019	0.064

Table 2: Official results from the ITAmoji competition, as compared to the first and second place groups.

Model	Macro F1	Micro F1
8 Head Attention	0.113	0.316
1 Head Attention	0.105	0.339
Local Attention	0.106	0.341
8 Head Local	0.106	0.337
Simplified Attention	0.106	0.341
8 Head Simplified	0.109	0.308
No Attention	0.11	0.319

Table 3: The best results from the different models on the dataset, run after the competition was over.

[2] Due to an off-by-one error in the conversion from network output to emoji, the official results for the no attention network are much worse than in actuality.

Model	Macro F1	Micro F1
8 Head Attention	[.10, .11]	[.30, .36]
1 Head Attention	[.09, .11]	[.30, .36]
Local Attention	[.10, .11]	[.30, .35]
8 Head Local	[.10, .11]	[.34, .36]
Simplified Attention	[.10, .11]	[.32, .36]
8 Head Simplified	[.10, .11]	[.31, .36]
No Attention	[.10, .11]	[.30, .36]

Table 4: The upper and lower bounds of the f1 scores of the different model types after the scores have plateaued in training and start oscillating.

While 8 head attention did outperform the 8 head local and simplified models, it's interesting to note that that isn't the case for the 1 head versions. Additionally, the bounds for the scores significantly overlap so there is no statistically significant gains for one method over the other. This result, along with my comparatively worse scores is probably because the max pooling at the end of my model was throwing away too much information in order to make the size consistent.

4 Discussion

In the upcoming sections, I discuss a possible problem with the design of my models and propose a few solutions for that problem. I further discuss the two new modifications on attention that I proposed and their possible uses.

4.1 Loss of Information While Pooling

For the problem of throwing away too much information during the pooling or downsampling phase, there are three main approaches that could be explored, each with their positives and negatives.

The first approach is to just fix the size of the input and use fully connected layers or similar approaches to find the correct output. This is the current approach by most researchers, and has shown good results. The main negative here is that the input size must be fixed, and fixing the input size could mean throwing away or adding information that isn't naturally there.

The second approach is to use a recurrent neural network neuron like an LSTM or a GRU with size equal to the output size to parse the result and output singular values for the final sequence. This would probably lead to better results but is going to be slower than the other approaches.

The last approach is to use convolutional layers with a large kernel size and stride (e.g. stride equal to the size of the kernel). This would allow the network to shrink the output size naturally, and would be faster than using an LSTM. The issue here is that in order to maintain the property that the network can have any input size, pooling or some other method of downsampling has to be used, potentially throwing away useful data.

4.2 Potential Uses of Simplified and Local Attention

While the original idea behind simplifying attention in such a manner as presented in this paper was to reduce computational cost and encourage easier learning by enforcing a softmax distribution of data, there didn't seem to be any benefit in doing so. In most cases the computational cost of a couple of matrix multiplications versus an element-wise product is negligible, so it would usually be better to just apply normal attention in those cases as it already covers the case of simplified attention in its implementation.

Similar to simplified attention, it doesn't necessarily make sense to use local attention instead of normal attention for small input sizes. Instead, it might make sense to switch out the linear projection on the queries and keys in normal attention with a convolutional projection but otherwise perform the scaled-dot product attention normally. This could be potentially useful if the problem being approached needs to map patterns to values instead of mapping values to values. One could of course extend this even further by also performing a convolutional projection on the values in order to map local patterns to other local patterns, and so on.

On the other hand, the local attention suggested in this paper could be useful in neural nets used for images and other large data, where it might not make sense to attend over the whole input. This is especially true in the initial layers of such neural networks where the neurons are only looking at a small section of the input in the first place. Beyond the smaller memory demands compared to normal attention, local attention could be useful in these layers because it provides a method to naturally figure out which patterns are important at these early layers.

Of course an alternative to local attention is to just take small patches of the image and apply the original formulation of scaled-dot product attention to get similar results. This idea was originally suggested as future work in Vaswani et al. (2017).

5 Conclusion

In this work I present simplified and local attention and test the methods in comparison to similar

models with normal attention and without any kind of attention at all. I also introduced a new strategy for classifying data with fully convolutional networks with any sized input.

The new model design was not without its own flaws, as it showed poor results for all modifications of the method. The poor results were probably due to the final pooling layer throwing away too much information. A better method would be to use LSTMs or specially designed convolutions in order to shrink the output to the correct size.

Future work will include further explorations of simplified and local attention to really get a grasp of what tasks they are good at and where, if anywhere, they show better efficiency or results than normal attention. In the future I will also further explore the new strategy for classification on any sized input with fully convolutional model and see what I can change and update in order to improve the results of the model.

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M. and Kudlur, M., 2016, November. Tensorflow: a system for large-scale machine learning. In *OSDI* (Vol. 16, pp. 265-283).

Conneau, A., Schwenk, H., Barrault, L. and Lecun, Y., 2016. Very deep convolutional networks for text classification. *arXiv preprint arXiv:1606.01781*.

Chollet, F., 2015. Keras.

He, K., Zhang, X., Ren, S. and Sun, J., 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).

Krizhevsky, A., Sutskever, I. and Hinton, G.E., 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097-1105).

Long, J., Shelhamer, E. and Darrell, T., 2015. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 3431-3440).

Ronzano, Francesco and Barbieri, Francesco and Wahyu Pamungkas, Endang and Patti, Viviana and Chiusaroli, Francesca. 2018. Overview of the EVALITA 2018 Italian Emoji Prediction (ITAMoji). *Proceedings of Fifth Italian Conference on Computational Linguistics (CLiC-it 2018) & Sixth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2018).*

Scherer, D., Müller, A. and Behnke, S., 2010. Evaluation of pooling operations in convolutional architectures for object recognition. In *Artificial Neural Networks–ICANN 2010* (pp. 92-101). Springer, Berlin, Heidelberg.

Tompson, J., Goroshin, R., Jain, A., LeCun, Y. and Bregler, C., 2015. Efficient object localization using convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 648-656).

Van Den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A.W. and Kavukcuoglu, K., 2016, September. WaveNet: A generative model for raw audio. In *SSW* (p. 125).

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł. and Polosukhin, I., 2017. Attention is all you need. In *Advances in Neural Information Processing Systems* (pp. 5998-6008).

Zhang, X., Zhao, J. and LeCun, Y., 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems* (pp. 649-657).

ITAmoji 2018: Emoji Prediction via Tree Echo State Networks

Daniele Di Sarli, Claudio Gallicchio, Alessio Micheli
Department of Computer Science
University of Pisa, Pisa, Italy
d.disarli@studenti.unipi.it, {gallicch,micheli}@di.unipi.it

Abstract

English. For the "ITAmoji" EVALITA 2018 competition we mainly exploit a Reservoir Computing approach to learning, with an ensemble of models for trees and sequences. The sentences for the models of the former kind are processed by a language parser and the words are encoded by using pretrained FastText word embeddings for the Italian language. With our method, we ranked 3rd out of 5 teams.

Italiano. *Per la competizione EVALITA 2018 sfruttiamo principalmente un approccio Reservoir Computing, con un ensemble di modelli per sequenze e per alberi. Le frasi per questi ultimi sono elaborate da un parser di linguaggi e le parole codificate attraverso degli embedding FastText preaddestrati per la lingua italiana. Con il nostro metodo ci siamo classificati terzi su un totale di 5 team.*

1 Introduction

Echo State Networks (Jaeger and Haas, 2004) are an efficient class of recurrent models under the framework of Reservoir Computing (Lukoševičius and Jaeger, 2009), where the recurrent part of the model ("reservoir") is carefully initialized and then left untrained (Gallicchio and Micheli, 2011). The only weights that are trained are part of a usually simple readout layer[1]. Echo State Networks were originally designed to work on sequences, however it has been shown how to extend them to deal with recursively structured data, and trees in particular, with Tree Echo State Networks (Gallicchio and Micheli, 2013), also referred to as TreeESNs.

We follow this approach for solving the ITAmoji task in the EVALITA 2018 competition (Ronzano et al., 2018). In particular, we parse the input texts into trees resembling the grammatical structure of the sentences, and then we use multiple TreeESN models to process the parse trees and make predictions. We then merge these models by using an ensemble to make our final predictions.

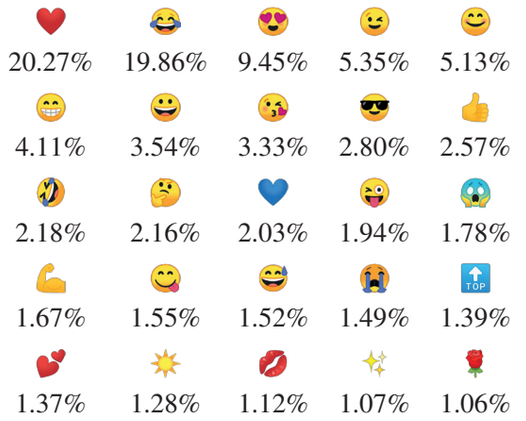


Figure 1: Emojis under consideration and their frequency within the dataset.

2 Task and Dataset

Given a set of Italian tweets, the goal of the ITAmoji task is to predict the most likely emoji associated with each tweet. The dataset contains 250,000 tweets in Italian, each of them originally containing only one (possibly repeated) of the 25 emojis considered in the task (see Figure 1). The emojis are removed from the sentences and used as targets.

The test dataset contains 25,000 tweets similarly processed.

[1] Trained in closed form, e.g. by Moore-Penrose pseudo-inversion, or Ridge Regression.

3 Preprocessing

The provided dataset has been shuffled and split into a training set (80%) and a validation set (20%).

We preprocessed the data by first removing any URL from the sentences, as most of them did not contain any informative content (e.g. "`https://t.co/M3StiVOzKC`"). We then parsed the sentences by using two different parsers for the Italian language: Tint[2] (Palmero Aprosio and Moretti, 2016) and spaCy (Honnibal and Johnson, 2015). This produced two sets of trees, both including information about the dependency relations between the nodes of each tree. We finally replace each word with its corresponding pretrained FastText embedding (Joulin et al., 2016).

4 Description of the system

Our ensemble is composed by 13 different models, 12 of which are TreeESNs and the other one is a Long Short-Term Memory (LSTM) over characters. Different random initializations ("trials") of the model parameters are all included in the ensemble in order to enrich the diversity of the hypotheses. We summarize the entire configuration in Table 1.

4.1 TreeESN models

The TreeESN that we are using is a specialization of the description given by Gallicchio and Micheli (2013), and the reader can refer to that work for additional details. Here, the state corresponding to node n of an input tree $\mathbf{t}$ is computed as:

$$\mathbf{x}(n) = f\left(\mathbf{W}_{in}\mathbf{u}(n) + \frac{1}{k}\sum_{i=1}^{k}\hat{\mathbf{W}}_i^n\mathbf{x}(ch_i(n))\right), \quad (1)$$

where $\mathbf{u}(n)$ is the label of node n in the input tree, k is the number of children of node n, $ch_i(n)$ is the i-th child of node n, $\mathbf{W}_{in}$ is the input-to-reservoir weight matrix, $\hat{\mathbf{W}}_i^n$ is the recurrent reservoir weight matrix associated to the grammatical relation between node n and its i-th child, and f is the element-wise applied activation function of the reservoir units (in our case, it is a $\tanh$). All matrices in Equation 1 are left untrained.

[2]Emitting data in the CoNLL-U format (Nivre et al., 2016), a revised version of the CoNLL-X format (Buchholz and Marsi, 2006).

Note that Equation 1 determines a recursive application (bottom-up visit) over each node of the tree $\mathbf{t}$ until the state for all nodes is computed, which we can express in structured form as $\mathbf{x}(\mathbf{t})$. The resulting tree $\mathbf{x}(\mathbf{t})$ is then mapped into a fixed-size feature representation via the χ state mapping function. We make use of *mean* and *sum* state mapping functions, respectively yielding the mean and the sum of all the states. The result, $\chi(\mathbf{x}(\mathbf{t}))$, is then projected into a different space by a matrix $\mathbf{W}_\phi$:

$$\hat{\mathbf{y}} = f_\phi\left(\mathbf{W}_\phi\,\chi(\mathbf{x}(\mathbf{t}))\right), \quad (2)$$

where f_ϕ is an activation function.

For the readout we use both a linear regression approach with L2 regularization known as Ridge regression (Hoerl and Kennard, 1970) and a multilayer perceptron (MLP):

$$\mathbf{y} = \text{readout}(\hat{\mathbf{y}}), \quad (3)$$

where $\mathbf{y} \in \mathbb{R}^{25}$ is the output vector, which represents a score for each of the classes: the index with the highest value corresponds to the most likely class.

4.2 CharLSTM model

The CharLSTM model uses a bidirectional LSTM (Hochreiter and Schmidhuber, 1997; Graves and Schmidhuber, 2005) with 2 layers, which takes as input the characters of the sentences expressed as pretrained character embeddings of size 300. The LSTM output is then fed into a linear layer with 25 output units.

Similar models have been used in recent works related to emoji prediction, see for example the model used by Barbieri et al. (2017), or the one by Baziotis et al. (2018), which is however a more complex word-based model.

4.3 Ensemble

We take into consideration two different ensembles, both containing the models in Table 1, but with different strategies for weighting the N_P predictions. In the following, let $\mathbf{Y} \in \mathbb{R}^{N_P \times 25}$ be the matrix containing one prediction per row.

The weights for the first ensemble (corresponding to the run file `run1.txt`) have been produced by a random search: at each iteration we compute a random vector $\mathbf{w} \in \mathbb{R}^{N_P}$ with entries sampled from a random variable W^2, $W \sim \mathcal{U}[0, 1]$. The square increases the probability of sampling

#	Class	Reservoir units	$\mathbf{f}_\phi$	Readout	Parser	Trials
1	TreeESN	1000	ReLU	MLP	Tint	10
2	TreeESN	1000	Tanh	MLP	Tint	10
3	TreeESN	5000	Tanh	MLP	Tint	1
4	TreeESN	5000	Tanh	MLP	spaCy	2
5	TreeESN	5000	ReLU	MLP	Tint	1
6	TreeESN	5000	ReLU	MLP	spaCy	1
7	TreeESN	5000	Tanh	Ridge regression	Tint	1
8	TreeESN	5000	Tanh	Ridge regression	spaCy	3
9	TreeESN	5000	ReLU	Ridge regression	Tint	1
10	TreeESN	5000	ReLU	Ridge regression	spaCy	3
11	TreeESN	5000	Tanh	Ridge regression	Tint	1
12	TreeESN	5000	Tanh	Ridge regression	spaCy	2
13	CharLSTM	–	–	–	–	1

Table 1: Composition of the ensemble, highlighting the differences between the models.

near-zero weights. After selecting the best configuration on the validation set, the predictions from each of the models are merged together in a weighted mean:

$$\bar{\mathbf{y}} = \mathbf{w}\mathbf{Y} \tag{4}$$

For the second type of ensemble (corresponding to the run file `run2.txt`) we adopt a multilayer perceptron. We feed as input the N_P predictions concatenated into a single vector $\mathbf{y}^{(1\ldots N_P)} \in \mathbb{R}^{25N_P}$, so that the model is:

$$\bar{\mathbf{y}} = \tanh\left(\mathbf{y}^{(1\ldots N_P)}\mathbf{W}_1 + \mathbf{b}_1\right)\mathbf{W}_2 + \mathbf{b}_2, \tag{5}$$

where the hidden layer has size 259 and the output layer is composed by 25 units.

In both types of ensemble, as before, the output vector contains a score for each of the classes, providing a way to rank them from the most to the least likely. The most likely class $\tilde{c}$ is thus computed as $\tilde{c} = \arg\max_i \bar{\mathbf{y}}_i$.

5 Training

The training algorithm differs based on the kind of model taken under consideration. We address each of them in the following paragraphs.

Models 1-6 The first six models are TreeESNs using a multilayer perceptron as readout. Given the fact that the main evaluation metric for the competition is the Macro F-score, each of the models has been trained by rebalancing the frequencies of the different target classes. In particular, the sampling probability for each input tree has been skewed so that the data extracted during training follows a uniform distribution with respect to the target class. For the readout part we use the Adam algorithm (Kingma and Ba, 2015) for the stochastic optimization of the multi-class cross entropy loss function.

Models 7-10 Models from 7 to 10 are again TreeESNs, but with a Ridge Regression readout. In this case, 25 classifiers are trained with a 1-vs-all method, one for each class, using binary targets.

Models 11-12 Models 11 and 12 are again TreeESNs with a Ridge Regression readout, but they are trained to distinguish only between the most frequent class, the second most frequent class and all the other classes aggregated together. This is done to try to improve the ensemble precision and recall for the top two classes.

Model 13 The last model is a sequential LSTM over character embeddings. Like in the first 6 models, the Adam algorithm is used to optimize the cross entropy loss function.

6 Results

The ensemble seems to bring a substantial improvement to the performance on the validation set, as highlighted in Table 2. This is possible thanks to the number and diversity of the different models, as we can see in Figure 2 where we show the Pearson correlation coefficients between the predictions of the models in the ensemble.

On the test set we scored substantially lower,

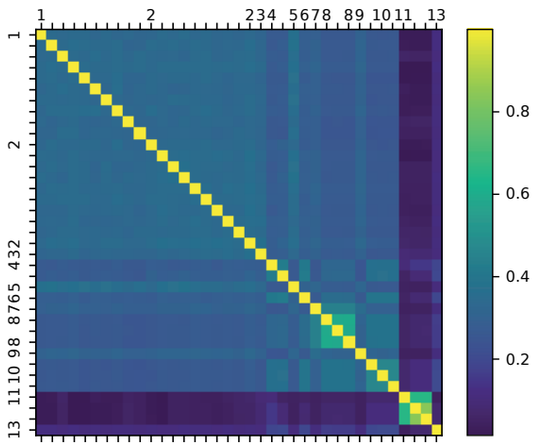


Figure 2: Plot of the correlation between the predictions of the models in the ensemble. For reasons of space, not all labels are shown on the axes.

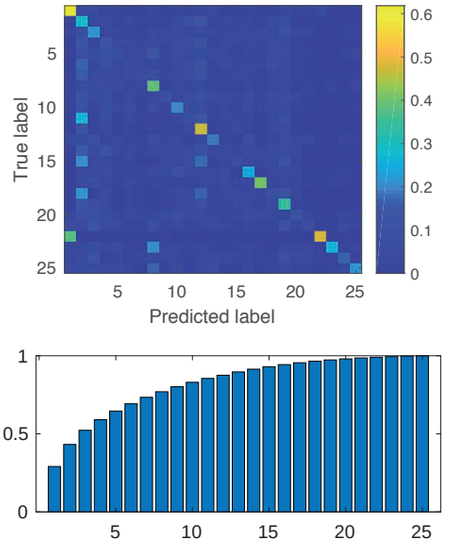


Figure 3: Confusion matrix (top) and accuracy at top-N (bottom) on the test set. Labels are ordered by frequency.

Run	Avg F1	Max F1	Ens. F1	CovE
run1	14.4	18.5	24.9	4.014
run2	14.4	18.5	26.7	3.428

Table 2: Performance obtained on the validation set for the two submitted runs. The columns are, in order, the average and maximum Macro-F1 over the models in the ensemble, and the Macro-F1 and Coverage Error of the ensemble.

Run	Macro-F1	Coverage Error
run1	**19.24**	5.4317
run2	18.80	5.1144

Table 3: Performance on the test set. These values have been obtained by retraining the models over the whole dataset (training set *and* validation set) after the final model selection phase.

with the Macro-F1 and Coverage Errors reported in Table 3. These numbers are close to those obtained by the top two models applied to the Spanish language in the "Multilingual Emoji Prediction" task of the SemEval-2018 competition (Barbieri et al., 2018), with F1 scores of 22.36 and 18.73 (Çöltekin and Rama, 2018; Coster et al., 2018). In Figure 3 we report the confusion matrix (with values normalized over the columns to address label imbalance) and the accuracy over the top-N classes.

An interesting characteristic of this approach, though, is computation time: we were able to train a TreeESN with 5000 reservoir units over 200,000 trees in just about 25 minutes, and this is without exploiting parallelism between the trees.

In ITAmoji 2018, our team ranked 3rd out of 5. Detailed results and rankings are available at `http://bit.ly/ITAmoji18`.

7 Discussion and conclusions

Different authors have highlighted the difference in performance between SVM models and (deep) neural models for emoji prediction, and more in general for text classification tasks, suggesting that simple models like SVMs are more able to capture the features which are most important for generalization: see for example the reports of the SemEval-2018 participants Çöltekin and Rama (2018) and Coster et al. (2018).

In this work, instead, we approached the problem from the novel perspective of reservoir computing applied to the grammatical tree structure of the sentences. Despite a significant performance drop on the test set[3] we showed that, paired with a rich ensemble, the method is comparable to the results obtained in the past by other participants in similar competitions using very different models.

[3]Probably due to overtraining: we observed that Macro-F1 overcame 0.40 in training.

References

Francesco Barbieri, Miguel Ballesteros, and Horacio Saggion. 2017. Are Emojis Predictable? *arXiv preprint arXiv:1702.07285*.

Francesco Barbieri, Jose Camacho-Collados, Francesco Ronzano, Luis Espinosa Anke, Miguel Ballesteros, Valerio Basile, Viviana Patti, and Horacio Saggion. 2018. SemEval 2018 Task 2: Multilingual Emoji Prediction. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 24–33.

Christos Baziotis, Nikos Athanasiou, Georgios Paraskevopoulos, Nikolaos Ellinas, Athanasia Kolovou, and Alexandros Potamianos. 2018. NTUA-SLP at SemEval-2018 Task 2: Predicting Emojis using RNNs with Context-aware Attention. *arXiv preprint arXiv:1804.06657*.

Sabine Buchholz and Erwin Marsi. 2006. CoNLL-X shared task on Multilingual Dependency Parsing. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*, pages 149–164. Association for Computational Linguistics.

Çağrı Çöltekin and Taraka Rama. 2018. Tübingen-Oslo at SemEval-2018 Task 2: SVMs perform better than RNNs in Emoji Prediction. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 34–38.

Joël Coster, Reinder Gerard Dalen, and Nathalie Adriënne Jacqueline Stierman. 2018. Hatching Chick at SemEval-2018 Task 2: Multilingual Emoji Prediction. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 445–448.

Claudio Gallicchio and Alessio Micheli. 2011. Architectural and Markovian factors of echo state networks. *Neural Networks*, 24(5):440–456.

Claudio Gallicchio and Alessio Micheli. 2013. Tree Echo State Networks. *Neurocomputing*, 101:319–337.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional LSTM networks. In *Neural Networks, 2005. IJCNN'05. Proceedings. 2005 IEEE International Joint conference on*, volume 4, pages 2047–2052. IEEE.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Arthur E Hoerl and Robert W Kennard. 1970. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67.

Matthew Honnibal and Mark Johnson. 2015. An Improved Non-monotonic Transition System for Dependency Parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1373–1378, Lisbon, Portugal, September. Association for Computational Linguistics.

Herbert Jaeger and Harald Haas. 2004. Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *Science*, 304(5667):78–80.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of Tricks for Efficient Text Classification. *arXiv preprint arXiv:1607.01759*.

Diederik P Kingma and Jimmy Lei Ba. 2015. Adam: Amethod for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*.

Mantas Lukoševičius and Herbert Jaeger. 2009. Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, 3(3):127–149.

Joakim Nivre, Marie-Catherine De Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D Manning, Ryan T McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, et al. 2016. Universal Dependencies v1: A Multilingual Treebank Collection. In *LREC*.

A. Palmero Aprosio and G. Moretti. 2016. Italy goes to Stanford: a collection of CoreNLP modules for Italian. *ArXiv e-prints*, September.

Francesco Ronzano, Francesco Barbieri, Endang Wahyu Pamungkas, Viviana Patti, and Francesca Chiusaroli. 2018. Overview of the EVALITA 2018 Italian Emoji Prediction (ITAMoji) Task. In Tommaso Caselli, Nicole Novielli, Viviana Patti, and Paolo Rosso, editors, *Proceedings of the 6th evaluation campaign of Natural Language Processing and Speech tools for Italian (EVALITA'18)*, Turin, Italy. CEUR.org.

The UNIBA System at the EVALITA 2018 Italian Emoji Prediction Task

Lucia Siciliani and **Daniela Girardi**
Department of Computer Science, University of Bari Aldo Moro
Via, E. Orabona, 4 - 70125 Bari (Italy)
{lucia.siciliani,daniela.girardi}@uniba.it

Abstract

English. This paper describes our participation in the ITAmoji task at EVALITA 2018 (Ronzano et al., 2018). Our approach is based on three sets of features, i.e. micro-blog and keyword features, sentiment lexicon features and semantic features. We exploit these features to train and combine several classifiers using different libraries. The results show how the selected features are not appropriate for training a linear classifier to properly address the emoji prediction task.

Italiano. *Questo articolo descrive l'approccio utilizzato per la partecipazione al task ITAmoji di EVALITA 2018 (Ronzano et al., 2018). Il nostro metodo si basa su tre insiemi di features: il primo rappresenta le informazioni intrinseche dei messaggi all'interno dei microblog, il secondo riguarda le informazioni derivanti dal lessico ed infine un terzo creato usando i principi di semantica distribuzionale. Queste features sono state utilizzate per addestrare diversi classificatori attraverso diverse librerie. I risultati ottenuti mostrano come le features selezionate non sono appropriate per addestrare un classificatore lineare nel task di predizione delle emoji.*

1 Introduction

Nowadays, emojis are widely used to express sentiments and emotions in written communication, which is becoming more and more popular due to the increasing use of social media. In fact, emojis can help the user to express and codify many different messages which can be also easily interpreted by a great audience since they are very intuitive. However, sometimes happens that their meaning is misleading, resulting in the misunderstanding of the entire message. The emoji detection has captured the interest of research since they could be relevant to improve sentiment analysis and user profiling tasks as well as the retrieval of social network material.

In particular, in the context of the International Workshop on Semantic Evaluation (SemEVAL 2018), the Multilingual Emoji Prediction Task (Barbieri et al., 2018) has been proposed for challenging the research community to automatically model the semantics of emojis occurring in English and Spanish Twitter messages. During this challenge, (Barbieri et al., 2017) created a model which outperforms humans in predicting the most probable emoji associated with a given tweet.

Twitter supports more than 1.000 emojis[1], belonging to different categories (e.g.: smiley and people, animals, fruits, etc.) and this number seems to grow.

In this paper, we used a set of features which showed promising results in predicting sentiment polarity in tweets (Basile and Novielli, 2014) in order to understand whether they could be used also to predict emoji or not. The paper is organized as follow: Section 2 describes the system and the exploited features, while in Section 3 we report the obtained results using different classifiers and their ensemble. Finally, in Section 4 we discuss our findings and Section 5 reports the conclusions.

2 System Description

In this section, we describe the approach used for solving the ITAmoji challenge. This task is structured as a multi-class classification since for each tweet it is possible to assign one of 25 emoji which however are mutually exclusive.

[1] https://it.piliapp.com/twitter-symbols/

The feature extraction was performed entirely using the language Java. First of all, each tweet was tokenized and stop-words were removed exploiting the "Twitter NLP and Part-of-Speech Tagging" API[2] developed by the Carnegie Mellon University. No other NLP steps, like stemming or PoS-tagging where considered since those features were considered not relevant for this particular kind of task.

Then we moved to the extraction of the features from the training data. These features can be categorized into three sets: one addressing the keywords and micro-blog features, the second one exploiting the polarity of each word in a semantic lexicon and the third one using their representation obtained through a distributional semantic model. A description of the different sets of features will be provided in Section 2.1.

After the features extraction, we obtained a total set of 342 features to be used to train a linear classifier. For classification, we decided to exploit the Weka API[3] and use an ensemble of three different classifiers to obtain better predictive results. The three classifiers that have been used are: the L2-regularized_L2-loss_support_vector_classification, the L2-regularized_logistic_regression, and the random forest classifier. The first two algorithms are based on the WEKA wrapper class for the Liblinear classifier (Fan et al., 2008) and were trained on the whole set of features, while the random forest was trained only over the keyword and micro-blog features. All the classifiers were combined using the soft-voting technique, which averages the sum of the output of each classifier over their overall number.

In the light of the results of the task given by the organizers, we conducted an in-depth analysis of our solution and discovered that due to a problem in the Liblinear WEKA wrapper, not all the classifiers returned a set of probability scores for multi-class classification thus compromising the results of all the ensemble. Therefore, even if out of the time scope of this challenge, we decided to try to use the scikit-learn (Buitinck et al., 2013) to build our classifiers and evaluate the impact of the selected features.

All the results will be summarized and discussed in Section 3 and Section 4.

[2]http://www.cs.cmu.edu/ ark/TweetNLP/

[3]9http://www.cs.waikato.ac.nz/ml/weka/

2.1 Features

As in the previous work of (Basile and Novielli, 2014), we defined three groups of features based on (i) keyword and micro-blogging characteristics, (ii) a sentiment lexicon and (iii) a Distributional Semantic Model (DSM). Keyword based features exploit tokens occurring in the tweets, considering only unigrams. During the tokenization phase user mentions, URLs and hash-tags are replaced with three meta-tokens: "USER", "URL", and "TAG", in order to count them and include their number as features. Other features connected to the micro-blogging environment are: the presence of exclamation and interrogative marks, adversative, disjunctive, conclusive, and explicative words, the use of uppercase and informal expressions of laughter, such as "ah ah". The list of micro-blogging features is reported in 1.

The second block of features consists of sentiment lexicon features. As Italian lexicon database, we used MultiWordNet (Pianta et al., 2002), where at each lemma is assigned a positive, negative and neutral score. In particular, we include features based on the prior polarity of words in the tweets. To deal with mixed polarity cases we defined two sentiment variation features so as to capture the simultaneous expression of positive and negative sentiment. We decided to include features related to the polarity of the tweets since emoji could be intuitively categorized into positive and negative and are usually used to enforce the sentiment expressed. The list of sentiment lexicon features is reported in 2. The last group of features is the semantic one, which exploits a Distributional Semantic Model. We used the vector embeddings for each word and the superposition operator (Smolensky, 1990) to compute an overall vector representation of the tweet. Analogously, we first computed a prototype vector for each polarity class (positive, negative, subjectivity and objectivity) as the sum of all the vector representations of each tweet to a certain class. Finally, we computed the element-wise minimum and maximum of the vectors representation of each word in the tweet and then the resulting vectors were then concatenated and used as features. This approach has been proved to work well and easy to compute for small texts like tweets and other micro-blog posts (De Boom et al., 2016). The list of sentiment lexicon features is reported in 3.

Microblog	Description
tag	total occurences of hashtags
url	total occurrences of URLs
user	total occurrences of user mentions
neg_count	total occurrences of ”non” word pt
exclamation	total occurrences of exclamation marks
interrogative	total occurrences of interrogative marks
adversative	total occurrences of adversative words
disgiuntive	total occurrences of disjunctive words
conclusive	total occurrences of conclusive words
esplicative	total occurrences of esplicative words
uppercase_ch	number of upper case characters
repeat_ch	number of consecutive repetitions of a character in a word
ahah_repetition	total occurrences of ”ahah” laughter expression

Table 1: Microblog Features.

3 Evaluation

The goal of the ITAmoji challenge is to evaluate the capability of each system to predict the right emoji associated with a tweet, regardless of its position in the text.

Organizers selected a subset of 25 emojis and provided 250,000 tweets for training, each tweet contains only one emoji which is extracted from text and given as a target feature. The training set is very unbalanced since three emojis (i.e.: read_heart ❤️, face_with_tears_of_joy 😂, and smiling_face_with_heart_eyes 😍) represent almost 50% of the whole dataset.

For the evaluation instead, the organizers created a test set made up of 25,000 tweets, keeping unchanged the ratio of the different classes over the whole set. The prediction for each tweet is composed by the list of all the 25 emojis ordered by their probability to be associated to the tweet: in this way, it is possible to evaluate the systems according to their accuracy up to a certain position in the rank. Nevertheless only the first emoji one was mandatory for the submission.

Systems were ranked according to the macro F-Measure but also other metrics have been calculated, i.e. the micro F-measure, the weighted F-measure, the coverage error and the accuracy (measured @5, @10, @15 and @20). The final results for the challenge are reported in table 5. We can see how while there is quite a difference between the results obtained for the macro-F1 score, the same does not happen with the micro F1 score. The same happens with the outcomes of the accuracy where, setting aside two runs, all the other obtain a result which is included between 0,5 and 0,8. In other words, even if the macro-F1 measure appears to be the most discriminating factor among all the runs, such a result is based on the presence of some classes which appear over a numerous amount of instances and this causes the classifiers to overfit over them.

Table 6 summarizes the results obtained using both WEKA (the one which was submitted, highlighted in italic) and scikit-learn. We used the scikit-learn library to perform a classification using the logistic regression and then adding, using a soft voting technique, a Naive-Bayes classifier and a Random Forest (rows 4 and 5 respectively). From these results we can see how, independently from the used classifier, the final results in terms of the metrics used for the evaluation over the test dataset stay quite similar among them. Specifically, these results depends on the fact that our system predicts only two label as first which are ”red_heart” and ”face whit tears”, resulting unable to classify correctly the other classes, as is shown in table 4. This outcome is then probably due to the set of features that we used, which does not manage to appropriately model the data in this task, even if it proved to be successful in another sentiment analysis context (Basile and Novielli, 2014). In the last column of table 6, we reported the average macro-F1 obtained performing 5-fold cross validation. The value for the first evaluation has not been calculated since the fault in the library described in section 2.

Sentiment Lexicon	Description
subjScore	sum of the positive and negative scores
objScore	sum of the neutral scores
hitSubj	number of tokens having the positive or negative score higher than zero
hitObj	number of tokens having the neutral score higher than zero
avgSubj	the ratio between subScore/hitSubj
avgObj	the ratio between objScore/hitObj
subObjDiff	difference
posScore	sum of positive scores for the tokens in the tweet
negScore	sum of negative scores for the tokens in the tweet
hitPos	number of tokens that have the positive score higher than zero
hitNeg	number of tokens that have the negative score higher than zero the
avgPos	ratio between posScore and hitPos
avgNeg	ratio between negScore and hitNeg
posnegScore	difference between avgPos and avgNeg
max_sum_subj_ratio	ratio between the maximum subjScore and number of token having positive and negative score higher than zero
max_obj_score_ratio	ratio between the maximum objScore and number of token having neutral score higher than zero
avgMaxPos	ratio between maxSumPos and hitMaxPos
avgMaxNeg	ration between maxSumNeg and hitMaxNeg
diff_avg_max_pos_neg	difference between avgMaxPos and avgMaxNeg
sentiment_variation	for each token occurring in the tweet a tag is assigned, according to the highest polarity score of the token in the Italian lexicon Tag values are in the set OBJ, POS, NEG The sentiment variation counts how many switches from POS to NEG, or vice versa, occur in the tweet
sentiment_variation_posneg	it is similar to the previous feature, but the OBJ tag is assigned only if both positive and negative scores are zero. Otherwise, the POS tag is assigned f the positive score is higher than the negative one, vice versa the NEG tag is assigned.
intensity	intensity of the tweet
polarity	polarity of the tweet

Table 2: Sentiment Lexicon Features.

4 Discussion

The overall results of the challenge show how this task is non-trivial and difficult to solve with high precision and the reason behind this is intrinsic to the task itself. First of all, there are several emojis which often differ only slightly from each other, furthermore, this meaning is deeply dependent on the single user and from the context. In fact, a single emoji (like 😂) could be used to convey both joy and fun or, on the contrary, it could also be used ironically with a negative meaning. To this extent, an interesting update for the task could be to leave the text of the tweet as it is so that the position could be also exploited to detect irony and other variations.

From the analysis of the overall results of the task emerged that there is a large gap between the macro-F1 scores which is not reflected by the micro-F1. For this particular task, where both training and testing dataset are heavily unbalanced, we think that the micro-F1 score is more suited to capture the performance of the submitted systems since it takes into account the support of each class.

There is a result which is particularly interesting that is, the value for the 5-fold using only the logistic regression as a classifier which is particularly high (0,358) and is opposing to the final score. This aspect surely needs further investiga-

Semantic	Description
vec	the sum of the vector representations of each word in the tweet
simNeg	the similarity between $\vec{t}$ and the negative prototype vector $\vec{p_s}$
simPos	the similarity between $\vec{t}$ and the positive prototype vector $\vec{p_s}$
simSubj	the similarity between $\vec{t}$ and the subjective prototype vector $\vec{p_s}$
simObj	the similarity between $\vec{t}$ and the objective prototype vector $\vec{p_s}$
vecMin	the element-wise minimum of the vectors representations of each word in the tweet
vecMax	the element-wise maximum of the vectors representations of each word in the tweet

Table 3: Semantic Features.

tions.

5 Conclusion

In this paper, we presented our contribution to the ITAmoji task of the EVALITA 2018 campaign. We tried to model the data by extracting features based on the keywords and micro-blogging characteristics, using a sentiment lexicon and finally using word embeddings. Apart from the characteristics of the different libraries available for machine learning purposes, the results show how, independently from the classifier, those features do not adapt to this problem. As future work, this analysis could also be extended with an ablation which would allow understanding if there are noisy features.

References

Francesco Barbieri, Miguel Ballesteros, and Horacio Saggion. 2017. Are emojis predictable? *arXiv preprint arXiv:1702.07285*.

Francesco Barbieri, Jose Camacho-Collados, Francesco Ronzano, Luis Espinosa Anke, Miguel Ballesteros, Valerio Basile, Viviana Patti, and Horacio Saggion. 2018. Semeval 2018 task 2: Multilingual emoji prediction. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 24–33.

Pierpaolo Basile and Nicole Novielli. 2014. Uniba at evalita 2014-sentipolc task: Predicting tweet sentiment polarity combining micro-blogging, lexicon and semantic features. *Proceedings of EVALITA*, pages 58–63.

Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. 2013. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122.

Cedric De Boom, Steven Van Canneyt, Thomas Demeester, and Bart Dhoedt. 2016. Representation learning for very short texts using weighted word embedding aggregation. *Pattern Recognition Letters*, 80:150–156.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Liblinear: A library for large linear classification. *Journal of machine learning research*, 9(Aug):1871–1874.

Emanuele Pianta, Luisa Bentivogli, and Christian Girardi. 2002. Multiwordnet: developing an aligned multilingual database. 1st gwc. *India, January*.

Francesco Ronzano, Francesco Barbieri, Endang Wahyu Pamungkas, Viviana Patti, and Francesca Chiusaroli. 2018. Overview of the EVALITA 2018 Italian Emoji Prediction (ITAMoji) Task. In *Proceedings of Sixth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2018)*, Turin, Italy. CEUR.org.

Paul Smolensky. 1990. Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial intelligence*, 46(1-2):159–216.

label	precision	recall	f1-score	support
beaming_face_with_smiling_eyes	0,000	0,000	0,000	1028
blue_heart	0,500	0,002	0,004	506
face_blowing_a_kiss	0,500	0,002	0,005	834
face_savoring_food	0,000	0,000	0,000	387
face_screaming_in_fear	0,000	0,000	0,000	444
face_with_tears_of_joy	0,313	0,448	0,369	4966
flexed_biceps	0,000	0,000	0,000	417
grinning_face	0,000	0,000	0,000	885
grinning_face_with_sweat	0,000	0,000	0,000	379
kiss_mark	0,000	0,000	0,000	279
loudly_crying_face	0,000	0,000	0,000	373
red_heart	0,259	0,909	0,403	5069
rolling_on_the_floor_laughing	0,000	0,000	0,000	546
rose	0,125	0,004	0,007	265
smiling_face_with_heart_eyes	0,135	0,004	0,008	2363
smiling_face_with_smiling_eyes	0,167	0,000	0,002	1282
smiling_face_with_sunglasses	0,000	0,000	0,000	700
sparkles	0,000	0,000	0,000	266
sun	0,000	0,000	0,000	319
thinking_face	0,000	0,000	0,000	541
thumbs_up	0,000	0,000	0,000	642
top_arrow	0,000	0,000	0,000	347
two_hearts	0,000	0,000	0,000	341
winking_face	0,000	0,000	0,000	1338
winking_face_with_tongue	0,000	0,000	0,000	483
avg / total	**0,164**	**0,274**	**0,156**	**25000**

Table 4: Classification report for each class.

teamName	macroF1	microF1	weightedF1	covErr	acc@5	acc@10	acc@t15	acc@t20
FBK_FLEXED	0,365	0,477	0,470	3,470	0,817	0,921	0,969	0,991
FBK_FLEXED	0,356	0,476	0,466	3,486	0,815	0,919	0,968	0,992
FBK_FLEXED	0,292	0,423	0,396	4,354	0,745	0,875	0,943	0,980
GW2017	0,233	0,401	0,378	5,662	0,672	0,815	0,894	0,930
GW2017	0,222	0,422	0,369	4,601	0,713	0,859	0,943	0,983
CIML-UNIPI	0,192	0,291	0,315	5,432	0,646	0,830	0,930	0,980
CIML-UNIPI	0,188	0,376	0,341	5,114	0,685	0,839	0,924	0,973
sentim	0,106	0,294	0,232	6,412	0,585	0,769	0,885	0,957
sentim	0,102	0,313	0,231	6,326	0,576	0,772	0,897	0,964
GW2017	0,038	0,119	0,110	13,489	0,279	0,430	0,560	0,663
UNIBA	0,032	0,274	0,156	6,697	0,588	0,760	0,864	0,935
sentim	0,019	0,065	0,040	12,458	0,292	0,488	0,644	0,740

Table 5: Final results of the challenge.

runName	macroF1	microF1	weightedF1	covErr	acc@t5	acc@10	acc@15	acc@20	K-fold
UNIBA_weka	*0,032*	*0,274*	*0,156*	*6,697*	*0,588*	*0,760*	*0,864*	*0,935*	-
UNIBA_sklearn_lr	0,039	0,257	0,156	6,459	0,610	0,765	0,873	0,947	0,358
UNIBA_sklearn_lr_nb	0,032	0,195	0,119	6,634	0,604	0,761	0,868	0,946	0,120
UNIBA_sklearn_lr_rf_nb	0,032	0,214	0,126	6,749	0,582	0,758	0,869	0,946	0,183

Table 6: Evaluation of the other classifiers using the same set of feature. In the second row are reported the results of our first submission. The last column reports the average macroF1 obtained performing a K-fold cross validation.

Predicting Emoji Exploiting Multimodal Data: FBK Participation in ITAmoji Task

Andrei Catalin Coman
Fondazione Bruno Kessler
coman@fbk.eu

Yaroslav Nechaev
Fondazione Bruno Kessler
nechaev@fbk.eu

Giacomo Zara
Fondazione Bruno Kessler
gzara@fbk.eu

Abstract

English. In this paper, we present our approach that has won the ITAmoji task of the 2018 edition of the EVALITA evaluation campaign[1]. ITAmoji is a classification task for predicting the most probable emoji (a total of 25 classes) to go along with the target tweet written by a given person in Italian. We demonstrate that using only textual features is insufficient to achieve reasonable performance levels on this task and propose a system that is able to benefit from the multimodal information contained in the training set, enabling significant F_1 gains and earning us the first place in the final ranking.

Italiano. *In questo articolo presentiamo l'approccio con cui abbiamo vinto la competizione ITAmoji dell'edizione 2018 di EVALITA[1]. ITAmoji è un task di classificazione per predire l'emoji più probabile (tra un totale di 25 classi) che possa essere associato ad un dato tweet scritto in italiano da uno specifico utente. Dimostriamo che utilizzare esclusivamente dati testuali non è sufficiente per ottenere un ragionevole livello di performance su questo task, e proponiamo un sistema in grado di beneficiare dalle informazioni multimodali contenute nel training set, aumentando significativamente lo score F_1 e guadagnando la prima posizione nella classifica finale.*

1 Introduction

Particularly over the last few years, with the increasing presence of social networks and instant messaging services in our lives, we have been witnessing how common it has become for average users to enrich natural language by means of emojis. An emoji is essentially a symbol placed directly into the text, which is meant to convey a simple concept or more specifically, as the name says, an emotion.

The emoji phenomenon has attracted considerable research interest. In particular, recent works have studied the connection between the natural language and the emojis used in a specific piece of text. The 2018 edition of EVALITA ITAmoji competition (Ronzano et al., 2018) is a prime example of such interest. In this competition, participants were asked to predict one of the 25 emojis to be used in a given Italian tweet based on a text, the date and the user that has written it. Differently from the similar SemEval (Barbieri et al., 2018) challenge, the addition of the user information significantly expanded the scope of potential solutions that could be devised.

In this paper, we describe our neural network-based system that exhibited the best performance among the submitted approaches in this task. Our approach is able to successfully exploit user information, such as the prior emoji usage history of a user, in conjunction with the textual features that are customary for this task. In our experiments, we have found that the usage of just the textual information from the tweet provides limited results: none of our text-based models were able to outperform a simple rule-based baseline based on prior emoji history of a target user. However, by considering all the modalities of the input data that were made available to us, we were able to improve our results significantly. Specifically, we combine into a single efficient neural network the typical Bi-LSTM-based recurrent architecture, that has shown excellent performance previously in this task, with the multilayer perceptron applied to user-based features.

[1] **EVALITA**: http://evalita.it/2018

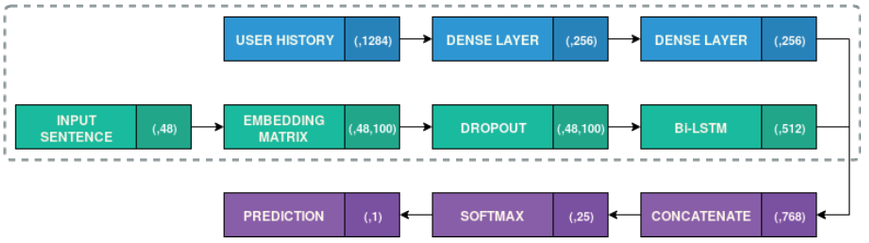


Figure 1: A diagram of the approach.

2 Description of the System

ITAmoji task is a classification task of predicting one of the 25 emojis to go along with the tweet. The training set provided by the organizers of the competition consists of 250 000 Italian tweets, including for each tweet the text (without the target emoji), the user ID and the timestamp as features. Participants were explicitly forbidden to expand the training set. Figure 1 provides an overview of our approach. In this section, we provide detailed descriptions of the methods we employed to solve the proposed task.

2.1 Textual features

In order to embed the textual content of the tweet, we have decided to apply a vectorization based on `fastText` (Bojanowski et al., 2017), a recent approach for learning unsupervised low-dimensional word representations. `fastText` sees the words as a collection of character n-grams, learning a representation for each n-gram. `fastText` follows the famous distributional semantics hypothesis utilized in other approaches, such as LSA, word2vec and GloVe. In this work, we exploit the Italian embeddings trained on text from Wikipedia and Common Crawl[2] and made available by the `fastText` authors[3]. Such embeddings include 300-dimensional vectors for each of 2M words in the vocabulary. Additionally, we have trained our own[4] embeddings using the corpus of 48M Italian tweets that were acquired from Twitter Streaming API. This yielded 1.1M 100-dimensional word vectors. Finally, we have also conducted experiments with word vectors suggested by the task organizers (Barbieri et al., 2016).

[2] `http://commoncrawl.org/`

[3] `https://github.com/facebookresearch/fastText/blob/master/docs/crawl-vectors.md`

[4] `https://doi.org/10.5281/zenodo.1467220`

2.2 User-based features

Rather than relying solely on a text of the target tweet, we exploit additional user-based features to improve performance. The task features many variations of the smiling face and three different heart emojis, making it impossible even for a human to determine the most suitable one just based on a tweet. One of the features we considered was the prior emoji distribution for a target author. The hypothesis was that the choice of a particular emoji is driven mainly by the personal user preferences exemplified by the previous emoji choices.

To this end, we have collected two different types of emoji history for each user. Firstly, we use labels in the training set to compute emoji distributions for each user yielding vectors of size 25. Users from the test set that were not present in the training set were initialized with zeroes. Secondly, we have gathered the last 200 tweets for each user using Twitter API[5], and then extracted and counted all emojis that were present in those tweets. This yielded a sparse vector of size 1284. At this step we took extra care to prevent data leaks: if a tweet from the test set ended up in the collected 200 tweets, it wasn't considered in the user history. The runs that used the former, training set-based approach had a "`_tr`" suffix in its name. The ones that used the full user history-based approach had a "`_ud`" suffix.

In addition to prior emoji distribution, we did preliminary experiments with user's social graph. Social graph, which is a graph of connections between the users, is shown to be an important feature for many tasks on social media, for example, user profiling. We followed the recently proposed approach (Nechaev et al., 2018a) to acquire 300-dimensional dense user representations based on a social graph. This feature, however, did not improve the performance of our approach and was excluded.

[5] `https://developer.twitter.com`

Layer	Parameter	Value
Textual Input	*seq. length*	48
Embedding	*input*	256
	output	100
	trainable	true
	l2 regularization	10^{-6}
Dropout	*probability*	0.4
Bi-LSTM	*output*	512

(a) Bi-LSTM model hyperparameters.

History Input	*input*	1284
Dense	*output*	256
	l2 regularization	10^{-5}
	activation	tanh
Dense	*output*	256
	l2 regularization	10^{-5}
	activation	tanh

(b) User history model hyperparameters.

Concatenate	*output*	768
Dense	*output*	25
	l2 regularization	–
	activation	softmax
Optimizer	*method*	$Adam$
	learning rate	0.001
	β_1	0.9
	β_2	0.999
	decay	0.001

(c) Joint model and optimizer parameters.

Table 1: Model hyperparameters.

2.3 RNN exploiting textual features

The Recurrent Neural Networks have turned out to be a powerful architecture when it comes to analyzing and performing prediction on sequential data. In particular, over the last few years, different variations of the RNN has shown to be the top performing approaches for a wide variety of tasks, including tasks in Natural Language Processing (NLP). RNN consumes the input sequence one element at the time, modifying the internal state along the way to capture relevant information from the sequence. When used for NLP tasks, RNN is able to consider the entirety of the target sentence, capturing even the longest dependencies within the text. In our system, we use the bi-directional long short-term memory (Bi-LSTM) variation of the RNN. This variation uses two separate RNNs to traverse the input sequence in both directions (hence bi-directional) and employs LSTM cells.

Input text provided by the organizers is split into tokens using a modified version of the Keras tokenizer (can be found in our repository). Then, the input tokens are turned into word vectors of fixed dimensionality using the embedding matrix of one of the approaches listed in Section 2.1. The resulting sequence is padded with zeroes to a constant length, in our case 48, and fed into the neural network.

2.4 Overall implementation

In order to accommodate both textual and user-based features, we devise a joint architecture that takes both types of features as input and produces probability distribution for the target 25 classes. The general logic of our approach is shown in Figure 1. The core consists of two main components:

- **Bi-LSTM.** The recurrent unit consumes the input sequence one vector at a time, modifying the hidden state (i.e., memory). After the whole sequence is consumed in both directions, the internal states of the two RNNs are concatenated and used as a tweet embedding. Additionally, we perform $l2$-regularization of the input embedding matrix and the dropout to prevent overfitting and fine-tune the performance. Table 1a details the hyperparameters we used for a textual part of our approach.

- **User-based features.** The emoji distribution we collected (as described in Section 2.2) was fed as input to a multilayer perceptron: two densely-connected layers with $tanh$ as activation and $l2$-regularization to prevent overfitting. Table 1b showcases the chosen hyperparameters for this component using the full user history as input.

The outputs of the two components are then concatenated and a final layer with the $softmax$ activation is applied to acquire the probability distribution of the 25 emoji labels. The network is then optimized jointly with cross entropy as the objective function using `Adam` optimizer. Table 1c includes all relevant hyperparameters we used for this step.

Since the runs are evaluated based on macro-F_1, in order to optimize our approach for this metric, we have introduced class weights into the objective function. Each class i is associated with the weight equal to:

$$w_i = \left(\frac{\max_i(N)}{N_i} \right)^{\alpha} \quad (1)$$

where N_i is the amount of samples in a particular class and $\alpha = 1.1$ is a hyperparameter we tuned

for this task. This way the optimizer is assigning a greater penalty for mistakes in rare classes, thus optimising for the target metric.

During the training of our approach, we employ an early stopping criteria to halt the training once the performance on the validation set stops improving. In order to properly evaluate our system, we employ 10-fold cross-validation, additionally extracting a small validation set from the training set for that fold to perform the early stopping. For the final submission we use a simple ensemble mechanism, where predictions are acquired independently from each fold and then averaged out to produce the final submission. Additionally, one of the runs was submitted using predictions from the random fold. Runs exploiting the ensemble approach have the "_10f" suffix, while runs using just one fold have the "_1f" suffix.

The code used to preprocess data, train and evaluate our approach is available on GitHub[6].

3 Evaluation setting

In this section, we provide details on some of the approaches we have tested during the development of our system, as well as the models we submitted for the official evaluation. In this paper, we report results for the following models:

- MF_HISTORY. A rule-based baseline that always outputs the most frequent emoji from the user history based on a training set.
- BASE_CNN. A basic Convolutional Neural Network (CNN) taking word embeddings as input without any user-based features.
- BASE_LSTM. A Bi-LSTM model described in Section 2.3 used with textual features only.
- BASE_LSTM_TR. The complete approach including both feature families with emoji distribution coming from the training set.
- BASE_LSTM_UD. The complete approach with emoji distribution coming from the most recent 200 tweets for each user.

For the other models tested during our local evaluation and complete experimental results, please refer to our GitHub repository.

Additionally, for the BASE_LSTM approach we report performance variations due to a choice of a particular word embedding approach. In particular, provided refers to the ones that were suggested by organisers, custom-100d indicates our fastText-based embeddings and common-300d refers to the ones available on fastText website. Table 2 details the performances of the mentioned models.

Finally, we submitted three of our best models for the official evaluation. All of the submitted runs use the Bi-LSTM approach with our custom-100d word embeddings along with some variation of user emoji distribution as detailed in Section 2.2. Two of the runs use the ensembling trick using all available cross-validation folds, while the remaining one we submitted ("_1f") uses predictions from just one fold.

4 Results

Here we report performances of the models benchmarked both during our local evaluation (Table 2) and the official results (Table 3). We started experiments with just the textual models testing different architectures and embedding combinations. Among those, the Bi-LSTM architecture was a clear choice, providing 1-2% F_1 over CNN, which led to us abandoning the CNN-based models. Among the three word embedding models we evaluated, our custom-100d embedding exhibited the best performance on Bi-LSTM, while common-300d showed the best performance using the CNN architecture.

After we have acquired the user emoji distributions, we have devised a simple baseline (MF_HISTORY), which, to our surprise, outperformed all the text-based models we've tested so far: 3% F_1 improvement compared to the best Bi-LSTM model. When we introduced the user emoji histories in our approach, we have gained a significant performance gain: 4% when using the scarce training set data and 12% when using the complete user history of 1284 emojis from recent tweets. During the final days of the competition, we have tried to exploit other user-based features to further bolster our results, for example, the social graph of a user. Unfortunately, such experiments did not yield performance gains before the deadline.

During the official evaluation, complete user history-based runs exhibited top performance with ensembling trick actually decreasing the final F_1. As we expected from our experiments, training set-based emoji distribution was much less per-

[6] **GitHub repository**: https://github.com/Remper/emojinet

Approach	Embedding	Accuracy	Precision	Recall	F1 macro
MF_HISTORY	–	0.4396	0.4076	0.2774	0.3133
BASE_CNN	common-300d	0.4351	0.3489	0.2464	0.2673
BASE_LSTM	common-300d	0.4053	0.3167	0.2534	0.2707
BASE_LSTM	provided	0.4415	0.3836	0.2408	0.2622
BASE_LSTM	custom-100d	0.4443	0.3666	0.2586	0.2809
BASE_LSTM_TR	custom-100d	0.4874	0.4343	0.3218	0.3565
BASE_LSTM_UD	custom-100d	0.5498	0.4872	0.4097	**0.4397**

Table 2: Performance of the approaches as tested locally by us.

Run	Accuracy@5	Accuracy@10	Accuracy@15	Accuracy@20	F1 macro
BASE_UD_1F	0.8167	0.9214	0.9685	0.9909	**0.3653**
BASE_UD_10F	0.8152	0.9194	0.9681	0.9917	0.3563
BASE_TR_10F	0.7453	0.8750	0.9434	0.9800	0.2920
gw2017_p.list	0.6718	0.8148	0.8941	0.9299	0.2329

Table 3: Official evaluation results for our three submitted runs and the runner-up model.

formant but still offered significant improvement over the runner-up team (gw2017_p.list) as shown in Table 3. Additionally, we detail the performance of our best submission (BASE_UD_1F) for each individual emoji in Table 4 and Figure 2.

5 Discussion and Conclusions

Our findings suggest that emojis are currently used mostly based on user preferences: the more prior user history we added, the more significant performance boost we have observed. Therefore, the emojis in a text cannot be considered independently from the person that has used them and textual features alone can not yield a sufficiently performant approach for predicting emojis. Additionally, we have shown that the task was sensitive to the choice of a particular neural architecture as well as to the choice of the word embeddings used to represent text.

An analogous task was proposed to the participants of the SemEval 2018 competition. The winners of that edition applied an SVM-based approach for the classification (Çöltekin and Rama, 2018). Instead, we have opted for a neural network-based architecture that allowed us greater flexibility to experiment with various features coming from different modalities: the text of the tweet represented using word embeddings and the sparse user-based history. During our experiments with the SemEval 2018 task as part of the NL4AI workshop (Coman et al., 2018), we have found the CNN-based architecture to perform better, while here the RNN was a clear winner. Such discrepancy might suggest that even within the emoji prediction task the effectiveness of different approaches may significantly vary based either on a language of the tweets or based on a way the dataset was constructed.

In the future, we would like to investigate this topic further by trying to study differences in

	Precision	Recall	F_1	Support
❤️	**0.7991**	0.6490	**0.7163**	5069
😂	0.4765	**0.7116**	0.5708	4966
💋	0.6402	0.4337	0.5171	279
😊	0.5493	0.4315	0.4834	387
🌹	0.4937	0.4453	0.4683	265
☀️	0.7254	0.3229	0.4469	319
😍	0.3576	0.5370	0.4293	2363
😘	0.4236	0.4089	0.4161	834
💙	0.4090	0.3775	0.3926	506
😊	0.4034	0.3354	0.3663	1282
😀	0.4250	0.3299	0.3715	885
😉	0.3743	0.3184	0.3441	1338
😁	0.3684	0.3239	0.3447	1028
✨	0.3854	0.2782	0.3231	266
🤣	0.3844	0.2711	0.3179	546
👍	0.3899	0.2648	0.3154	642
😎	0.3536	0.2743	0.3089	700
💪	0.3835	0.2566	0.3075	417
🤔	0.3525	0.1922	0.2488	541
💕	0.2866	0.2639	0.2748	341
😭	0.2280	0.2922	0.2562	373
🔝	0.2751	0.2133	0.2403	347
😅	0.2845	0.1741	0.2160	379
😜	0.3154	0.1822	0.2310	483
😱	0.2956	0.1824	0.2256	444

Table 4: Precision, Recall, F_1 of our best submission and the number of samples in test set for each emoji.

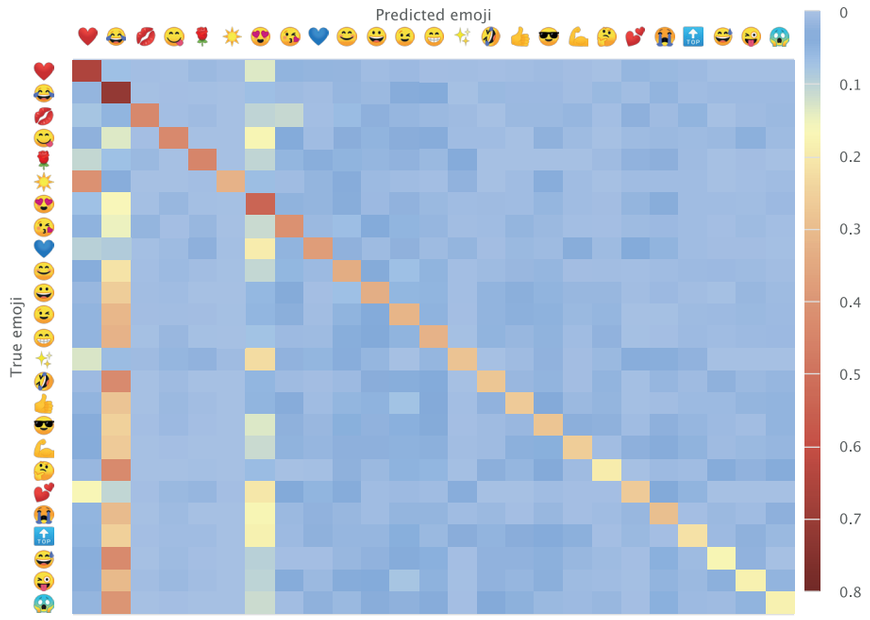


Figure 2: Confusion matrix for our best submission normalized by support size: each value in a row is divided by the row marginal. Diagonal values give recall for each individual class (see Table 4).

emoji usage between languages and communities. Additionally, we aim to further improve our approach by identifying more user-based features, for example, by taking into account the feature families suggested by Nechaev et al. (Nechaev et al., 2018b).

References

Francesco Barbieri, German Kruszewski, Francesco Ronzano, and Horacio Saggion. 2016. How cosmopolitan are emojis?: Exploring emojis usage and meaning over different languages with distributional semantics. In *Proceedings of the 2016 ACM on Multimedia Conference*, pages 531–535. ACM.

Francesco Barbieri, Jose Camacho-Collados, Francesco Ronzano, Luis Espinosa-Anke, Miguel Ballesteros, Valerio Basile, Viviana Patti, and Horacio Saggion. 2018. SemEval-2018 Task 2: Multilingual Emoji Prediction. In *Proc. of the 12th Int. Workshop on Semantic Evaluation (SemEval-2018)*, New Orleans, LA, United States. Association for Computational Linguistics.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Çagri Çöltekin and Taraka Rama. 2018. Tübingen-oslo at semeval-2018 task 2: Svms perform better than rnns in emoji prediction. In *Proc. of The 12th Int. Workshop on Semantic Evaluation, SemEval@NAACL-HLT, New Orleans, Louisiana*, pages 34–38.

Andrei Catalin Coman, Giacomo Zara, Yaroslav Nechaev, Gianni Barlacchi, and Alessandro Moschitti. 2018. Exploiting deep neural networks for tweet-based emoji prediction. In *Proc. of the 2nd Workshop on Natural Language for Artificial Intelligence co-located with 17th Int. Conf. of the Italian Association for Artificial Intelligence (AI*IA 2018), Trento, Italy*.

Yaroslav Nechaev, Francesco Corcoglioniti, and Claudio Giuliano. 2018a. Sociallink: Exploiting graph embeddings to link dbpedia entities to twitter profiles. *Progress in AI*, 7(4):251–272.

Yaroslav Nechaev, Francesco Corcoglioniti, and Claudio Giuliano. 2018b. Type prediction combining linked open data and social media. In *Proc. of the 27th ACM Int. Conf. on Information and Knowledge Management, CIKM 2018, Torino, Italy*, pages 1033–1042.

Francesco Ronzano, Francesco Barbieri, Endang Wahyu Pamungkas, Viviana Patti, and Francesca Chiusaroli. 2018. Overview of the evalita 2018 italian emoji prediction (itamoji) task. In Tommaso Caselli, Nicole Novielli, Viviana Patti, and Paolo Rosso, editors, *Proceedings of the 6th evaluation campaign of Natural Language Processing and Speech tools for Italian (EVALITA'18)*, Turin, Italy. CEUR.org.

The validity of word vectors over the time for the EVALITA 2018 Emoji prediction task (ITAmoji)

Mauro Bennici
You Are My GUide
mauro@youaremyguide.com

Xileny Seijas Portocarrero
You Are My GUide
xileny@youaremyguide.com

Abstract

English. This document describes the results of our system in the evaluation campaign on the prediction of Emoji in Italian, organized in the context of EVALITA 2018[1] (Ronzano et al., 2018). Given the text of a tweet in Italian, the task is to predict the emoji most likely associated with that tweet among the 25 emojis selected by the organizers. In this report, we describe the three proposed systems for evaluation. The approach described starts from the possibility of creating two different models, one for the part of categorization, and the other for the part of polarity. And to combine the two models to get a better understanding of the dataset.

Italiano. Questo documento descrive i nostri risultati del nostro sistema nella campagna di valutazione sulla predizione delle Emoji in italiano, organizzata nel contesto di EVALITA 2018.

Dato il testo di un tweet in italiano, il task consiste nel predire l'emoji più probabilmente associata a quel tweet tra le 25 emojis selezionate dagli organizzatori. In questo report descriviamo i tre sistemi proposti per la valutazione.

L'approccio descritto parte dalla possibilità di creare due modelli diversi, uno per la parte di categorizzazione, e l'altro per la parte di polarità. E di unire i due modelli per ottenere una maggiore comprensione del dataset.

1 Introduction

In the field of communication, the importance of addressing your audience with a common language in which the customer can recognize and identify with each other is fundamental. In social interactions, an increasing amount of communication occurs in a non-verbal way, as with emoji. Being able to predict the best emoji to use in a message can increase the perception of the same and give strength to the message itself.

In the context of the Italian Emoji Prediction task called ITAmoji, we have tried to predict one of 25 possible emojis from different tweets.

Despite the knowledge of how a system of SVM could be the best solution for the problem, as per the previous context SemEval 2018 (Rama & Çöltekin, 2018), a different approach was chosen to focus on the effectiveness of a Neural Network based model

2 Description of the system

We first started by cleaning the given data from all the noise information. All the punctuation marks were removed from the text of tweets, and we focused on cleaning the text and removing ambiguities such as shortened words and abbreviations. We substituted all the hyperlinks with a more generic word "LINK" and we did the same with the usernames preceded by '@' (users' tags), after seeing that it was not relevant in the prediction of the most likely emoji for the tweet.

We tried removing the stop words from the tweets' text to leave only the words with relevant meaning in it, but the results were poor.

[1] https://sites.google.com/view/itamoji/

Then we converted every word of the tweet's text into its lemma, and while doing the lemmatization, we saw that sometimes the username was misleading in the text, so we chose to remove it and substitute it with a more generic word 'USERNAME'.

We used two different fastText[2] vectors created in the 2016 and the other created in 2017, all with Italian tweets containing at least one emojis. The idea is to analyze if different fastText vectors created with tweets published in different periods could discover the use of the emojis and its evolution over the time.

The system created is an Ensemble of two different models to replicate the result obtained in the emotion classification (Akhtar at al., 2018).
The first model is a bi-directional Long Short-Term Memory (BI-LSTM) implemented in Keras[3].

Layer (type)	Output Shape	Param #
e (Embedding)	(None, 25, 200)	34978200
b (Bidirectional)	(None, 512)	935936
d (Dense)	(None, 25)	12825

A dropout and a recurrent_dropout of 0.9.
The optimizer is the RMSProp. The embedding is trainable.

The second is a LightGBM[4], where the following properties are extracted from the tweet text:

- length of the tweet
- percentage of special characters
- the number of exclamation points
- the number of question marks
- the number of words
- the number of characters
- the number of spaces
- the number of stop words
- the ratio between words and stop words
- the ratio between words and spaces
- the ratio between words and hashtags

and are joined to the vector created by the bi-gram and the trigram of the tweet itself at word and character level.
The number of leaves is 250, the learner set as 'Feature', and the learning rate at 0.04.

The ensemble is done in the weighted average when the BI_LSTM decide the 60% of the vote and the LightGBM the 40%.

It was also tried to add a linear classifier but the attempt did not provide any advantage. The cross-validation task to find a good weight was ineffectual and the provision was insignificant.

3 Results

The results of the Bi-LSTM were:

BI-LSTM with 2016 fastText		
precision	recall	F1 score
0.3595	0.2519	0.2715

Table 1: precision, recall, and F1 score with 2016 fastText vector.

BI-LSTM with 2017 fastText		
precision	recall	F1 score
0.3520	0.2577	0.2772

Table 2: precision, recall, and F1 score with 2017 fastText vector.

The model trained with the data published during the 2017 is quite similar to the model trained with the data published on the 2016.

The results of the LightGBM were:

[2] https://fasttext.cc
[3] https://keras.io
[4] https://github.com/Microsoft/LightGBM

LightGBM only text		
precision	recall	F1 score
0.2399	0.3094	0.2460

Table 3: precision, recall, and F1 score

The LightGBM model was also tested by adding to the already mentioned properties additional information such as the user ID and information extracted from the tweet date such as day, month, the day of the week and time.

The results obtained also indicate here that there is a correspondence between the use of emojis, the user, the time and the day. For example the Christmas tree in December or the heart emoji in the evening hours.

LightGBM with user and date		
precision	recall	F1 score
0.5044	0.2331	0.2702

Table 4: precision, recall, and F1 score

The level of Precision obtained in this way was very high even if the F1 score is still lower than the BI-LSTM model.

To avoid the unbalancing of the emojis present in the training dataset various undersampling and oversampling operations were performed without any appreciable results.
Turning to the result of the ensemble of the two models we had a marked increase in the F1 score thanks to the substantial growth of the Recall in both cases.

In the tables 5 and 6 there are the results from the minimum and the maximum F1 score obtained during the process of the ensemble.

BI-LSTM with 2016 fastText + LightGBM only text		
precision	recall	F1 score
0.4121	0.2715	0.2955

Table 5: precision, recall, and F1 score

BI-LSTM with 2017 fastText + LightGBM with user and date		
precision	recall	F1 score
0.3650	0.2917	0.3048

Table 6: precision, recall, and F1 score

The result of the validation was however very far from that obtained during the training phase. It will be necessary to evaluate if, as in the research Exploring Emoji Usage and Prediction Through a Temporal Variation Lens (Barbieri et al., 2018), it was the time of the publication of the tweets is to be distant from the date of the tweets analyzed.

If the tweets analyzed were too different from those of the training dataset, if the users in the test dataset have different behaviors, or if the system suffered from some kind of overfitting (visible in the third submission, gw2017_pe).

	gw2017_e	gw2017_p	gw2017_pe
Macro F1	0.222082	**0.232940**	0.037520
Micro F1	**0.421920**	0.400920	0.119480
Weighted F1	0.368996	**0.378105**	0.109664
Coverage error	**4.601440**	5.661600	13.489400
Accuracy at 5	**0.713000**	0.671840	0.279280
Accuracy at 10	**0.859040**	0.814880	0.430360
Accuracy at 15	**0.943080**	0.894160	0.560000
Accuracy at 20	**0.982520**	0.929920	0.662720

Table 7: macro F1, micro F1, weighted F1, coverage error, accuracy at 5, 10, 15 and 20 for the three runs submitted.

In table 8 we can observe the result of the three submissions split for each emoji.

Runs	gw2017_e			gw2017_p			gw2017_pe			
Label	precision	recall	f1-score	precision	recall	f1-score	precision	recall	f1-score	quantity

😁	0.2150	0.0224	0.0405	0.1395	0.0642	0.0879	0.0242	0.0107	0.0148	1028
💙	0.4429	0.1917	0.2676	0.3608	0.2075	0.2635	0.0215	0.0178	0.0195	506
😘	0.3142	0.3417	0.3274	0.2726	0.3681	0.3133	0.0343	0.0468	0.0396	834
😋	0.3624	0.3540	0.3582	0.3204	0.3850	0.3498	0.0107	0.0155	0.0127	387
😱	0.3137	0.0360	0.0646	0.1608	0.0518	0.0784	0.0077	0.0023	0.0035	444
😂	0.3533	**0.8357**	0.4967	0.4185	0.6104	0.4965	0.2024	**0.2648**	**0.2294**	4966
💪	0.3902	0.1535	0.2203	0.3257	0.2038	0.2507	0.0263	0.0264	0.0263	417
😀	0.2917	0.0554	0.0931	0.2190	0.0678	0.1035	0.0328	0.0102	0.0155	885
😅	0.0800	0.0053	0.0099	0.0581	0.0132	0.0215	0.0380	0.0079	0.0131	379
💋	0.5143	0.2581	0.3437	0.4464	0.2688	0.3356	0.0044	0.0036	0.0039	279
😭	0.3144	0.1635	0.2152	0.1895	0.2520	0.2163	0.0135	0.0134	0.0135	373
❤️	**0.7567**	0.7497	**0.7531**	**0.7803**	**0.7358**	**0.7574**	**0.2101**	0.2016	0.2058	5069
😉	0.1714	0.0110	0.0207	0.1053	0.0183	0.0312	0.0137	0.0018	0.0032	546
🌹	0.3769	0.1849	0.2481	0.3439	0.2038	0.2559	0.0142	0.0113	0.0126	265
😍	0.3137	0.4109	0.3558	0.2952	0.4824	0.3663	0.0904	0.1583	0.1151	2363
😊	0.2384	0.1607	0.1920	0.2068	0.1747	0.1894	0.0526	0.0546	0.0536	1282
😎	0.3174	0.1043	0.1570	0.2432	0.1157	0.1568	0.0317	0.0243	0.0275	700
✨	0.4667	0.1579	0.2360	0.3239	0.1729	0.2255	0.0096	0.0075	0.0084	266
🌟	0.6735	0.3103	0.4249	0.6221	0.3354	0.4358	0.0106	0.0063	0.0079	319
🤔	0.3204	0.1220	0.1767	0.2101	0.2680	0.2356	0.0193	0.0185	0.0189	541
👍	0.4278	0.1199	0.1873	0.3043	0.1526	0.2033	0.0249	0.0171	0.0203	642
🔝	0.3220	0.0548	0.0936	0.2368	0.0778	0.1171	0.0187	0.0086	0.0118	347
💕	0.3590	0.0411	0.0737	0.2537	0.0499	0.0833	0.0161	0.0059	0.0086	341
😉	0.2082	0.1181	0.1507	0.1584	0.2451	0.1924	0.0369	0.0419	0.0392	1338
😜	0.2609	0.0248	0.0454	0.1860	0.0331	0.0562	0.0336	0.0083	0.0133	483
avg / total	**0.4071**	**0.4219**	0.3690	0.3870	0.4009	**0.3781**	0.1051	0.1195	0.1097	25000

Table 8: Precision, Recall, F1 Score, and quantity in the test set of the 25 most frequent emojis.

It is important to note that despite the significant presence of the dataset the 😁 has a meager final

F1 score. On the other hand, the 🌟 has a high F1 score even if only present in 319 items.

4 Discussion

In the study of the dataset, three critical issues emerged.

- The first is that the use of similar emojis seems more dictated by a personal choice of the user.
  There are not many pieces of evidence because the use of one emoji is preferred.
  In particular for the following emoji:
  😀🤣😂😀
- The second is that, especially in cases where a tweet begins by indicating a USERNAME, or in a mention or a direct response, the use of emoji takes on a sub-language value. That is, the use of a specific word or emoji has a meaning that only the tweet recipients know. Use of emoji 😅 and 🤣 could be irony or just references to previous pasted experiences in common.
- Thirdly, the strong imbalance of the training dataset is not the only reason for the unbalanced prediction of some emojis, as in the case of 😀 and 💥.

5 Conclusion

The result of the ensemble was pretty good and demonstrate the validity of this kind of approach. The use of emoji is personal and also depends on the context and the people in the discussion. A system with the emojis with the same meaning merged could be more proficient and ready for the production.

In the near future, we will evaluate the speed and effectiveness of a CNN model in which the operation of the BI-LSTM and the features extrapolation used in the LightGBM model can be merged during the same training session.

We will also focus on the creation of fastText vectors of different size containing tweets for specific contexts and published in different periods to identify the periodicity and variation in the use of particular emoji. The intent is to discover other hidden patterns, more than the obvious that has emerged for the holiday periods.

Reference

Francesco Ronzano, Francesco Barbieri, Endang Wahyu Pamungkas, Viviana Patti, and Francesca Chiusaroli (2018) ITAmoji: Overview of the Italian emoji prediction task @ Evalita 2018. In Proceedings of Sixth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2018), CEUR.org, Turin, Italy.

Taraka Rama and Çagri Çöltekin. (2018, June). Tübingen-Oslo at SemEval-2018 Task 2: SVMs perform better than RNNs in Emoji Prediction. Retrieved from https://aclanthology.coli.uni-saarland.de/papers/S18-1004/s18-1004

Francesco Barbieri, José Camacho-Collados, Francesco Ronzano, Luis Espinosa Anke, Miguel Ballesteros, Valerio Basile, Viviana Patti, Horacio. (2018) Saggion: SemEval 2018 Task 2: Multilingual Emoji Prediction. SemEval@NAACL-HLT 2018: 24-33. ACL.

Md Shad Akhtar, Deepanway Ghosal, Asif Ekbal, Pushpak Bhattacharyya, Sadao Kurohashi. (2018, October 15). A Multi-task Ensemble Framework for Emotion, Sentiment and Intensity Prediction. Retrieved from https://arxiv.org/abs/1808.01216

Francesco Barbieri, Luis Marujo, Pradeep Karuturi, William Brendel, Horacio Saggion. (2018, May 02). Exploring Emoji Usage and Prediction Through a Temporal Variation Lens. Retrieved from https://arxiv.org/abs/1805.00731

A Kernel-based Approach for Irony and Sarcasm Detection in Italian

Andrea Santilli and **Danilo Croce** and **Roberto Basili**
Universitá degli Studi di Roma "Tor Vergata"
Via del Politecnico 1, Rome, 00133, Italy
andrea.santilli@live.it
{croce,basili}@info.uniroma2.it

Abstract

English. This paper describes the UNITOR system that participated to the Irony Detection in Italian Tweets task (IronITA) within the context of EvalIta 2018. The system corresponds to a cascade of Support Vector Machine classifiers. Specific features and kernel functions have been proposed to tackle the different subtasks: Irony Classification and Sarcasm Classification. The proposed system ranked first in the Sarcasm Detection subtask (out of 7 submissions), while it ranked sixth (out of 17 submissions) in the Irony Detection task.

Italiano. *Questo lavoro descrive il sistema UNITOR che è stato valutato nel corso dell' Irony Detection in Italian Tweets task IronITA ad EvalIta 2018. Il riconoscimento del sarcasmo e dell'ironia nei tweet corrisponde all'orchestrazione di diversi classificatori di tipo Support Vector Machine (SVM), studiata per risolvere i task legati alla competizione. Rappresentazioni specifiche sono state progettate per modellare i tweet attraverso la applicazione di funzioni kernel diverse utilizzate dai classificatori SVM. Il sistema ha ottenuto risultati promettenti risultando vincitore di 1 dei 2 task proposti.*

1 Introduction

Modern social networks allow users to express themselves, writing their opinions about facts, things and events. In social posting, people often adopt figurative languages, e.g. Irony and Sarcasm. These communication mechanism must be carefully considered in the automatic processing of texts in social media: as an example, they may be used to convey the opposite of literal meaning and thus just intentionally sketching a secondary or extended meaning (Grice, 1975). On Twitter, users can express themselves with very short messages. Given the short length, the information useful to detect figurative uses of natural language is very limited or missing. Irony and sarcasm detection represents challenging tasks within Sentiment Analysis and Opinion Mining often undermining the overall system accuracy. There is not a clear separation between irony and sarcasm, but the former is often considered to include the latter. In particular sarcasm is defined as sharp or cutting ironic expressions towards a particular target with the intention to offend (Joshi et al., 2016).

This paper presents and describes the UNITOR system participating in the *Irony Detection in Italian Tweets* (IronITA) task (Cignarella et al., 2018) within the EvalIta 2018 evaluation campaign. The system faces both the proposed subtasks within IronITA: *Irony Classification* and *Sarcasm Classification*. In a nutshell, the former subtask aims at evaluating the performance of a system in capturing whether a message is ironic or not. The second subtask is intended to verify if, given an ironic tweet, a system is able to detect sarcasm within the message.

The classification of each tweet is carried out by applying a cascade of kernel-based Support Vector Machines (Vapnik, 1998). In particular, two binary SVM classifiers (one per subtask) are designed to adopt specific combinations of different kernel functions, each operating over a task-specific tweet representation. This work extends the modeling proposed in (Castellucci et al., 2014) that was proved to be beneficial within the Irony Detection subtask within SENTIPOLC 2014. The UNITOR system here presented ranked 1^{st} and 2^{nd} in the Sarcasm Detection subtask, while it ranked 6^{th} and 7^{th} within the Irony Detection subtask.

In Section 2 the SVM classifiers, their features and the underlying kernels are described and the adopted workflow is presented. In Section 3 the performance measures of the system are reported, while Section 4 derives the conclusions.

2 System Description

The UNITOR system adopts a supervised learning setting where a multiple kernel-based approach is adopted to acquire two binary Support Vector Machine classifiers (Shawe-Taylor and Cristianini, 2004): a first classifier discriminates between ironic and non ironic tweets, while a second one decides whether an ironic tweet is sarcastic or not. In the rest of this section, we first summarize the pre-processing stage as well as the adopted linguistic resources (e.g. word embeddings or lexicons). Then, the feature modeling designed for the two steps is discussed.

2.1 Tweet processing and resources

Each tweet is linguistically processed through an adapted version of the Chaos parser (Basili and Zanzotto, 2002) in order to extract the information required for feature modeling, e.g. the Part-of-speech tags and lemmas of individual words. A normalization step is applied before the standard Natural Language Processing activity is carried out. A number of actions is performed: fully capitalized words are converted into their lowercase counterparts; hyperlinks are replaced by a special token, i.e. `LINK`; characters repeated more than three times are cleaned, as they increase lexical data sparseness (e.g. "*nooo!!!!!*" is converted into "*noo!!*"); all emoticons are replaced by special tokens[1].

In the feature modeling activities, we relied on several linguistic resources, hereafter reported.

First, we used a **Word Space model** (or Word Embedding) to generalize the lexical information of the (quite small) training material: this semantic space is obtained starting from a corpus of Italian tweets downloaded in July 2016 of about 10 millions of tweets (same used in Castellucci et al. (2016a)) and it is a 250-dimensional embedding generated according to a Skip-gram model (Mikolov et al., 2013)[2].

Moreover, we adopted a large scale sentiment specific lexicon, i.e., the **Distributional Polarity Lexicons** *(DPL)* (Castellucci et al., 2016b)[3]. Distributional Polarity Lexicon *(DPL)* is introduced to inject sentiment information of words in the learning process through a large-scale polarity lexicon that is automatically acquired according to the methodology proposed in (Castellucci et al., 2015). This method leverages on word embeddings to model lexical polarity by transferring it from entire sentences whose polarity is known. The process is based on the capability of word embeddings to represent both sentences and single words in the same space (Landauer and Dumais, 1997). First, sentences (here tweets) are labeled with some polarity classes: in (Castellucci et al., 2015) this labeling is achieved by applying simple heuristics, e.g. Distant Supervision (Go et al., 2009). The labeled dataset is projected in the embedding space by applying a simple but effective linear combination of the word vectors composing each sentence. Then, a polarity classifier is trained over these sentences in order to emphasize dimensions of the space that are more related to the polarity classes. The DPL is generated by classifying each word (represented in the embedding through a vector) with respect to each targeted class, using the confidence level of the classification to derive a word polarity signature. For example, in a DPL the word *ottimo* is 0.89 positive, 0.04 negative and 0.07 neutral. For more details, please refer to (Castellucci et al., 2015).

Finally, we also adopted an **Irony specific Corpus** to capture terms and patterns that are often used to express irony (e.g., "*non lo riconosceresti neanche se ti cascasse*" or "*... allora piove* "): it is a corpus composed by a set of Italian tweets automatically extracted using Distance Supervision (Go et al., 2009). In particular the Irony specific Corpus is composed by a set of 6,000 random tweets in Italian, freely available, assumed to be ironic, as they contain hashtags such as *#irony* or *#ironia*.

2.2 Modeling irony and sarcasm in kernel-based learning

UNITOR is based on kernel functions operating on vector representations of tweets, described hereafter. After the language processing stage, each tweet allows generating one of the follow-

[1] We normalized 113 well-known emoticons in 13 classes.

[2] The following settings were adopted: window 5 and min-count 10 with hierarchical softmax.

[3] The adopted lexicon has been downloaded from `http://sag.art.uniroma2.it/demo-software/distributional-polarity-lexicon/`

ing representations[4] , later exploited by the kernel-based SVM in the training/classification steps.

2.2.1 Irony-specific Features

The aim of this set of features is to capture irony by defining a set of irony-specific features inspired by the work of (Castellucci et al., 2014).

Word Space Vector (WS) is a 250-dimensional vector representation of the average semantic meaning of a tweet according to a Word space model. It is used to generalize the lexical information of tweets. We can summarize it as $\sum_{t \in T} We(t)/|T|$, where T is the set of nouns, verbs, adjectives, adverb and hashtag in a tweet t and $We(t)$ is a function that returns the 250-dimensional word embedding of the word t. Other words, such as articles and preposition are discarded as they do not convey useful information within a word space.

Irony Specific BOW (ISBOW) is a BoW vector representing the lexical information expressed in a message. The main difference with respect to a conventional BOW representation is the adopted weighting scheme. In fact, in this case we leverage on the Word Space previously described. For each dimension representing a lemma/part-of-speech pair, its weight is computed as the cosine similarity between the word embedding vector of the considered word and the vector obtained from the linear combination of all the other words in the message (*WS*)[5]. This vector aims at capturing how much odd is the occurrence of a given word in a sentence aiming at capturing its unconventional uses: it should be an indicator of potential ironic mechanisms, as suggested in (Castellucci et al., 2014).

Irony Specific BOW(Adjective, Noun, Verb) (ISBOW-A), (ISBOW-S), (ISBOW-V) are three BoW vectors that use the same weighting scheme specified in ISBOW. Each vector represents one individual part of speech (i.e. adjective, noun and verb), as words belonging to different POS-tag categories may be characterized by quite different distributions.

Irony Specific Mean and Variance (ISMV) is a 4-dimensional vector representation that summarized the information captured by the previous representations. It contains mean and variance of the cosine similarity, calculated between the words in a tweet in the ISBOW representation, and the maximum and minimum of the cosine similarity per tweet. This vector aims at summarizing the distribution and potential "spikes" of unusual patterns of use for words in a sentence.

Irony Specific Mean and Variance (Adjective, Noun, Verbs) (ISMV-A), (ISMV-S), (ISMV-V) are three distinct 4-dimensional vectors that are the same specified in ISMV, with the only difference that each representation works on one specific part of speech, respectively adjectives, nouns and verbs.

Char n-gram BOWs (n-CHARS) is a representation expressing the char n-grams contained in a message. We used 4 n-CHARS representations: 2-CHARS BoW vector representing 2-char-ngrams contained in a message, 3-CHARS BoW vector representing 3-char-ngrams, 4-CHARS BoW vector representing 4-char-ngrams, 5-CHARS BoW vector representing 5-char-ngrams. The aim of this representation is to capture the usage of specific textual patterns, e.g., *hihihihi* often used to express irony.

Synthetic Features (SF) is a 7-dimensional vector containing the following synthetic features, traditionally used in Sentiment Analysis: percentage of the number of uppercase letters in the tweet, number of exclamation marks, number of question marks, number of colons, number of semicolons, number of dots, number of commas. It has been inspired by works on irony detection of (Carvalho et al., 2009; Reyes et al., 2012).

2.2.2 Features based on Distribution Polarity Lexicons

The aim of this group of features is to exploit the negative evaluation towards a target typical of sarcasm mechanism (Joshi et al., 2016) using a polarity lexicon, here a Distribution Polarity Lexicon (DPL).

Distributional Polarity Lexicon Sum (DSUM) is a 15-dimensional vector representation made by the concatenation of 5 different representations, i.e. $\frac{1}{|N_T|}\sum_{w \in N_T} \underline{w}^p$, $\frac{1}{|V_T|}\sum_{w \in V_T} \underline{w}^p$, $\frac{1}{|Adj_T|}\sum_{w \in Adj_T} \underline{w}^p$, $\frac{1}{|Adv_T|}\sum_{w \in Adv_T} \underline{w}^p$, $\frac{1}{|T|}\sum_{w \in T} \underline{w}^p$, where N_T, V_T, Adj_T, Adv are the nouns, verbs, adjectives and adverbs occurring in the tweet,

[4]The code for the feature vector generation is available at: https://github.com/andry9454/ironySarcasmDetection

[5]If a word was not found in the word embedding, a smoothing weight, representing the mean cosine similarity between word and *WS* in the training set, is applied as cosine similarity measure.

$T = N_T \cup V_T \cup Adj_T \cup Adv_T$ and $\underline{w}^p$ expresses the 3-dimensional polarity lexicon entry[6] for the word w. This feature summarize the a-priori sentiment of words according to the different morphological categories. We speculate that the regularities or contrasts between these distributions may suggest the presence of irony or sarcasm.

Distributional Polarity Lexicon BoW (DBOW) is a BoW vector representing, for each word in a message, its polarity (positive, negative and neutral) as a three dimensional score derived from the DPL.

2.2.3 Irony Corpus Features

Generalizing linguistic information useful for Irony or Sarcasm detection is a very challenging tasks, as the adoption of these figurative languages mainly concern extra-linguistic phenomena. The idea underlying the following features is to define a tweet representation that is not directly connected to their (possibly limited) linguistic material, but that is connected with respect to a larger set of information derived from a Irony specific Corpus, i.e., a large scale collection of a Ironic tweets. This is used to extract an Irony specific Lexicon: a set of words and patterns occurring in such corpus with a high frequency.

Irony Corpus BOW (ICBOW) is a BoW vector representing lemmas of Nouns, Verbs, and Adjective in a message. Again, the main difference with respect to a conventional BoW representation is the adopted weighting scheme: a word is weighted 1.0 if that particular word was in the *Irony specific Corpus*, otherwise is weighted 0.

Irony Corpus weighted BOW (ICwBOW) is a BoW vector representing lemmas of Nouns, Verbs, and Adjective in a message. A word is weighted $log(f + 1)$ where f is the frequency of that particular word in the *Irony Corpus*.

Irony Corpus weighted Mean (ICM) is a 2-dimensional vector representation that summarize the mean words weight observed in a ICBOW representation and the mean over the ICwBOW. These scores indicate how a words or patterns in a tweet occur also in the Irony specific corpus. This information is very interesting as it is not tied to the lexical information from a tweet, so allowing a more robust generalization.

Irony Corpus BOW (bi-grams, three-grams) (IC2BOW), (IC3BOW) are two distinct BoW vector respectively representing bi-grams and three-grams of surface words in a message. The weighting scheme is the same explained in ICBOW.

Irony Corpus weighted BOW (bi-grams, three-grams) (IC2wBOW), (IC3wBOW) are two distinct BoW vectors respectively representing bi-grams and three-grams of terms in a message. The weighting scheme is the same explained in ICwBOW.

Irony Corpus weighted Mean (bi-grams, three-grams) (IC2M), (IC3M) are two distinct 2-dimensional vector representations that contain means that are the same specified in ICM, with the only difference that the first representation works on bi-grams (IC2BOW, IC2wBOW), while the second works on three-grams (IC3BOW, IC3wBOW).

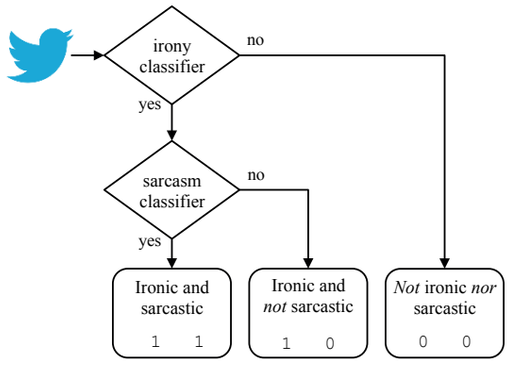


Figure 1: The UNITOR classifier workflow

3 Experimental evaluation and results

The cascade of SVM classifiers implemented in UNITOR is summarized in Figure 1. After the linguistic processing stage and the feature extraction stage, each tweet is classified by a binary classifier, the so-called *irony classifier*. If a message is judged as *not ironic*, we assume that it is also *not sarcastic* (according to the task guidelines) and a label 0 0 is assigned to it. Otherwise, if the tweet is judged as *ironic*, the second binary classifier, devoted to Sarcasm Detection, is invoked. If positive, the tweet is *sarcastic* and the message is labeled with 1 1, otherwise, 1 0.

Separated representations are considered in the *constrained* and *unconstrained* settings, according to the guidelines in (Cignarella et al., 2018). In the constrained setting only feature vectors using tweet information or public available lexicons are considered (*Irony-specific Features* and *Features derived from a DPL*). In the unconstrained

[6] If a word w is not present in the distributional polarity lexicon, $\underline{w}^p$ is set to the default $[0.33, 0.33, 0.33]$.

setting, feature vectors are derived also using the Irony specific Corpus.

In our experiments, we train the SVM classifiers using the same kernel combination for Irony Detection and Sarcasm Detection. Even if this is not a general solution (different tasks may require different representations) we adopted this greedy strategy, leaving the SVM to select the most discriminative information.

A normalized linear combination of specific kernel functions is used in both subtasks. In the linear combination, a specific linear kernel is applied to the following sparse representations: `ISBOW`, `ISBOW-A`, `ISBOW-S`, `ISBOW-V`, `DBOW`, `2BOW`, `3BOW`, `4BOW`, `5BOW`, `ICBOW`, `IC2BOW`, `IC3BOW`, `ICwBOW`, `IC2wBOW`, `IC3wBOW`; in the same combination a RBF kernel (Shawe-Taylor and Cristianini, 2004) is applied to the following dense representations `WS`, `SF`, `ICM`, `IC2M`, `IC3M`, `DSUM`, `ISMV`, `ISMV-A`, `ISMV-S`, `ISMV-V`[7].

Each SVM classifier is built by using the KeLP framework[8] (Filice et al., 2018).

Figure 1 reflects also the learning strategy that has been set up during the training phase: the Irony Classifier was trained on the complete training dataset composed by the entire training set (made of $3,977$ tweets) while the Sarcasm Classifier is trained only on the ironic tweets in the training dataset (made of $2,023$ tweets). A 10-fold cross validation strategy was applied to optimize the SVM parameters, while the linear combination of the kernel assigns the same weights to each kernel function.

In Table 1 the performances of the *Irony Classification* task are reported: in the constrained run the UNITOR system ranks 7^{th}, while in 6^{st} position in the unconstrained one. For this task the adopted representations were able to correctly determine whether a message is ironic with good precision. However, the winning system (about 3 points ahead) results more effective in the detection of non-ironic messages. In fact, according to the F1-score on the Ironic class, the system would have been ranked 2^{nd}. We also evaluated a slightly different modeling with two additional features vector, i.e., a classic BoW composed of lemmas derived from the input tweet, and a BoW of bigrams. These features have been excluded from our official submission to keep the model simple. However, these simple features would have been beneficial and the system would have ranked 2^{nd}. Performances on the *Sarcasm Classification* are in Table 2: UNITOR here ranks in 1^{st} or in 2^{nd} position, in the constrained and unconstrained run, respectively. Differences between the two results are not significant. Nevertheless the further features derived from the Irony specific corpus allow improving results (especially in terms of recall) in the Sarcasm Detection task. For this latter task, results achieved by UNITOR suggest that the proposed modeling, in particular the contribution of Polarity Features, seem to be beneficial. To prove it, we decided to evaluate a run with the same winning features, except Polarity Features. In this case the UNITOR system would have been ranked 4^{th}. These Polarity Features seem to exploit the negative bias typical of sarcasm (Joshi et al., 2016).

	Not Ironic			Ironic			Mean
	P	R	F1	P	R	F1	F1
1st	.785	.643	.707	.696	.823	.754	**.731**
2nd*	.771	.617	.686	.680	.816	.741	.714
6th(*u*)	.778	.577	.662	.662	.834	.739	.700
7th(*c*)	.764	.593	.668	.666	.816	.733	.700
BL	.501	1.00	.668	1.00	.000	.000	.334

Table 1: Constrained (*c*) and Unconstrained (*u*) UNITOR results in Irony Detection, i.e. scores 6th and 7th.

	Not Sarcastic			Sarcastic			Mean
	P	R	F1	P	R	F1	F1
1st(c)	.362	.584	.447	.492	.407	.446	**.520**
2nd(u)	.355	.553	.432	.469	.449	.459	.518
4th*	.344	.566	.428	.344	.566	.428	.508
BL	.296	.132	.183	1.00	.000	.000	.199

Table 2: Constrained (*c*) and Unconstrained (*u*) UNITOR results in Sarcasm Detection, i.e. 1st and 2nd scores

4 Conclusions

In this paper we described the UNITOR system participating to the IronITA task at EvalIta 2018. The system won 1 of the 2 evaluations carried out in the task, and in the worst case it ranked in the 6^{th} position. The good results in constrained and unconstrained settings suggest that the proposed irony and sarcasm specific features were beneficial to detect irony and sarcasm also in short messages. However, further work is needed to improve the *non ironic* F1 scores. The nature of the task seems to be non trivial also for a human reader, as some tweets extracted from

[7] A with $\gamma = 1$ was used in each RBF kernel

[8] `http://www.kelp-ml.org/`

the test set suggest: "@*beppe_grillo Beppe..tu sei un grande..questi si stanno finendo di mangiare l'Italia..*", "s*cusa hai ancora posti liberi nella app di braccialetti rossi*?"; here the interpretation of irony goes beyond the textual information and it is very difficult to state if these messages are ironic or not. Since tweets are very short, useful information for detecting irony is often out of the message, like this ironic tweet extracted from the test set may suggest: "*immagine perfetta ed esplicita che descrive la realtá della "buona scuola" a renzopoli*"; in this case the system may fail without a proper representation for the meaning of the neologism "*renzopoli*". So we think that the contextual approach suggested in (Vanzo et al., 2014) will be explored in future research.

References

Roberto Basili and Fabio Massimo Zanzotto. 2002. Parsing engineering and empirical robustness. *Nat. Lang. Eng.*, 8(3):97–120.

Paula Carvalho, Luís Sarmento, Mário J. Silva, and Eugénio de Oliveira. 2009. Clues for detecting irony in user-generated contents: Oh...!! it's "so easy" ;-). In *1st CIKM WS on Topic-sentiment Analysis for Mass Opinion*, pages 53–56. ACM.

Giuseppe Castellucci, Danilo Croce, Diego De Cao, and Roberto Basili. 2014. A multiple kernel approach for twitter sentiment analysis in italian. In *Fourth International Workshop EVALITA 2014*.

Giuseppe Castellucci, Danilo Croce, and Roberto Basili. 2015. Acquiring a large scale polarity lexicon through unsupervised distributional methods. In *Proc. of 20th NLDB*, volume 9103. Springer.

Giuseppe Castellucci, Danilo Croce, and Roberto Basili. 2016a. Context-aware convolutional neural networks for twitter sentiment analysis in italian. In *Proceedings of 3rd Italian Conference on Computational Linguistics (CLiC-it 2016) & Fifth EVALITA Workshop 2016), Napoli, Italy, December 5-7, 2016.*

Giuseppe Castellucci, Danilo Croce, and Roberto Basili. 2016b. A language independent method for generating large scale polarity lexicons. In *Proceedings of the 10th LREC Conference (LREC'16)*, Portoroz, Slovenia. European Language Resources Association (ELRA).

Alessandra Teresa Cignarella, Simona Frenda, Valerio Basile, Cristina Bosco, Viviana Patti, and Paolo Rosso. 2018. Overview of the evalita 2018 task on irony detection in italian tweets (ironita). In Tommaso Caselli, Nicole Novielli, Viviana Patti, and Paolo Rosso, editors, *Proceedings of the 6th evaluation campaign of Natural Language Processing and Speech tools for Italian (EVALITA'18)*, Turin, Italy. CEUR.org.

Simone Filice, Giuseppe Castellucci, Giovanni Da San Martino, Alessandro Moschitti, Danilo Croce, and Roberto Basili. 2018. Kelp: a kernel-based learning platform. *Journal of Machine Learning Research*, 18(191):1–5.

Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. *Processing*, pages 1–6.

H Paul Grice. 1975. Logic and conversation. *1975*, pages 41–58.

Aditya Joshi, Pushpak Bhattacharyya, and Mark James Carman. 2016. Automatic sarcasm detection: A survey. *CoRR*, abs/1602.03426.

Tom Landauer and Sue Dumais. 1997. A solution to plato's problem: The latent semantic analysis theory of acquisition, induction and representation of knowledge. *Psychological Review*, 104.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Antonio Reyes, Paolo Rosso, and Davide Buscaldi. 2012. From humor recognition to irony detection: The figurative language of social media. *Data and Knowledge Engineering*, 74(0):1 – 12.

John Shawe-Taylor and Nello Cristianini. 2004. *Kernel Methods for Pattern Analysis*. Cambridge University Press.

Andrea Vanzo, Danilo Croce, and Roberto Basili. 2014. A context-based model for sentiment analysis in twitter. In *Proceedings of COLING*, pages 2345–2354. ACL and Dublin City University.

Vladimir N. Vapnik. 1998. *Statistical Learning Theory*. Wiley-Interscience.

UNIBA - Integrating distributional semantics features in a supervised approach for detecting irony in Italian tweets

Pierpaolo Basile and **Giovanni Semeraro**
Department of Computer Science
University of Bari Aldo Moro
Via, E. Orabona, 4 - 70125 Bari (Italy)
{pierpaolo.basile,giovanni.semeraro}@uniba.it

Abstract

English. This paper describes the UNIBA team participation in the IronITA 2018 task at EVALITA 2018. We propose a supervised approach based on LIBLINEAR that relies on keyword, polarity, micro-blogging features and representation of tweets in a distributional semantic model. Our system ranked 3rd and 4th in the irony detection subtask. We participated only in the constraint run exploiting the training data provided by the task organizers.

Italiano. *Questo articolo descrive la partecipazione del team UNIBA al task IronITA 2018 organizzato durante EVALITA 2018. Nell'articolo proponiamo un approccio supervisionato basato su LIBLINEAR che sfrutta le parole chiave, la polarità, attributi tipici dei micro-blog e la rappresentazione dei tweet in uno spazio semantico distribuzionale. Il nostro sistema si è classificato terzo e quarto nel sotto task di identificazione dell'ironia. Abbiamo partecipato solamente nel constraing run utilizzando i dati di training forniti dagli organizzatori del task.*

1 Introduction

The irony is defined as "the use of words that say the opposite of what you really mean, often as a joke and with a tone of voice that shows this"[1]. This suggests us that when we are analyzing written text for detecting irony, we should focus our attention on those words that are used in an unconventional context. For example, given the tweet: *"S&P ha declassato Mario Monti da Premier a Badante #declassaggi"*[2], we can observe that the word "badante" *(caregiver)* is used in an unconventional context, since "caregiver" usually does not co-occur with words "Premier" or "Mario Monti".

Following this idea in our work we introduce a feature able to detect words used out of their usual context. Moreover, we integrate further features based on keywords, bigrams, trigrams, polarity and micro-blogging features as reported in (Basile and Novielli, 2014). Our idea is supported by best systems participating in the Semeval-2018 task 3 - Irony detection in English tweets (Van Hee et al., 2018), where the best systems not based on deep learning exploit features based on polarity contrast information and context incongruity.

We evaluate our approach in the context of the IronITA task at EVALITA 2018 (Cignarella et al., 2018). The goal of the task is to predict irony in Italian tweets. The task is organized in two sub-tasks: 1) irony detection and 2) different types of irony. In the second sub-task participates must identify if irony belongs to sarcasm or not. In this paper, we propose an approach which is able to detect the presence of irony without taking into account different types of irony. We evaluate the approach in a constrained setting using only the data provided by task organizers. The only external resources exploited in our approach are a polarity lexicon and a collection of about 40M tweets randomly extracted from TWITA(Basile and Nissim, 2013) (a collection of about 800M Italian tweets).

The paper is structured as follows: Section 2 describes our system, while evaluation and results are reported in Section 3. Final remarks are provided in Section 4.

2 System Description

Our approach adopts a supervised classifier based on LIBLINEAR (Fan et al., 2008), in particular we use the L2-regularized L2-loss linear SVM. Each tweet is represented using several sets of features:

[1] Oxford Learner Dictionary

[2] In English: "S&P has downgraded Mario Monti from Premier to Caregiver"

keyword-based : keyword-based features exploit tokens occurring in the tweets. Unigrams, bigrams and trigrams are considered. During the tokenization we replace the user mentions and URLs with two metatokens: "_USER_", "_URL_";

microblogging : microblogging features take into account some attributes of the tweets that are peculiar in the context of microblogging. We exploit the following features: the presence of emoticons, item character repetitions[3], informal expressions of laughters[4] and the presence of exclamation and interrogative marks. All microblobbing features are binary.

polarity : this block contains features extracted from the SentiWordNet (Esuli and Sebastiani, 2006) lexicon. We translate SentiWordNet in Italian through MultiWordNet (Pianta et al., 2002). It is important to underline that SentiWordNet is a synset-based lexicon while our Italian translation is a word based lexicon. In order to automatically derive our Italian sentiment lexicon from SentiWordNet, we perform three steps. First, we translate the synset offset in SentiWordNet from version 3.0 to 1.6[5] using automatically generated mapping file. Then, we transfer the prior polarity of SentiWordNet to the Italian lemmata. Finally, we expand the lexicon using Morph-it! (Zanchetta and Baroni, 2005), a lexicon of inflected forms with their lemma and morphological features. We extend the polarity scores of each lemma to its inflected forms. Details about the creation of the sentiment lexicon are reported in (Basile and Novielli, 2014). The obtained Italian translation of SentiWordNet is used to compute three features based on prior polarity of words in the tweets: 1) the maximum positive polarity; 2) the maximum negative polarity; 3) polarity variation: for each token occurring in the tweet a tag is assigned, according to the highest polarity score of the token in the Italian lexicon. Tag values are in the set {OBJ, POS , NEG}. The sentiment variation counts how many switches from POS to NEG, or vice versa, occur in the tweet.

distributional semantics features : we compute two kinds of distributional semantics features:

1. given a set of unlabelled downloaded tweets, we build a geometric space in which each word is represented as a mathematical point. The similarity between words is computed as their closeness in the space. To represent a tweet in the geometric space, we adopt the superposition operator (Smolensky, 1990), that is the vector sum of all the vectors of words occurring in the tweet. We use the tweet vector $\overline{t}$ as a semantic feature in training our classifiers;
2. we extract three features that taking into account the usage of words in an unconventional context. In particular, for each word w_i we compute a score ac_i that measures how the word is out of its conventional context. Finally, we compute three features: the average, the maximum and the minimum of all the ac_i scores. More details about the computation of the ac_i score are reported in Subsection 2.1.

2.1 Distributional Semantics Features

The distributional semantics model is built on a collection of tweets. We randomly extract 40M tweets from TWITA and build a semantic space based on the Random Indexing (RI) (Sahlgren, 2005) technique using a context windows equals to 2. Moreover, we consider only words occurring more than ten times[6]. The context window is dynamic and it does not take into account words that are not in the vocabulary. Our vocabulary contains 105,543 terms.

The mathematical insight behind the RI is the projection of a high-dimensional space on a lower dimensional one using a random matrix; this kind of projection does not compromise distance metrics (Dasgupta and Gupta, 1999).

Formally, given a $n \times m$ matrix A and an $m \times k$ matrix R, which contains random vectors, we define a new $n \times k$ matrix B as:

$$A^{n,m} \cdot R^{m,k} = B^{n,k} \quad k << m \qquad (1)$$

[3]These features usually plays the same role of intensifiers in informal writing contexts.

[4]i.e., sequences of "ah".

[5]Since MultiWordNet is based on WordNet 1.6.

[6]We call this set of words: the vocabulary.

The new matrix B has the property to preserve the distance between points, that is if the distance between two any points in A is d; then the distance d_r between the corresponding points in B will satisfy the property that $d_r \approx c \times d$. A proof of that is reported in the Johnson-Lindenstrauss lemma (Dasgupta and Gupta, 1999).

Specifically, RI creates the *WordSpace* in two steps:

1. A context vector is assigned to each word. This vector is sparse, high-dimensional and ternary, which means that its elements can take values in {-1, 0, 1}. A context vector contains a small number of randomly distributed non-zero elements, and the structure of this vector follows the hypothesis behind the concept of Random Projection;

2. Context vectors are accumulated by analyzing co-occurring words. In particular, the semantic vector for any word is computed as the sum of the context vectors for words that co-occur with the analyzed word.

Formally, given a corpus C of n documents, and a vocabulary V of m words extracted from C, we perform two steps: 1) assign a context vector c_i to each word in V; 2) compute a semantic vector sv_i for each word w_i as the sum of all context vectors assigned to words co-occurring with w_i. The context is the set of m words that precede and follow w_i.

For example, considering the following tweet: *"siete il buono della scuola fatelo capire"*. In the first step we assign a random vector to each term as follows:

$$c_{siete} = (-1, 0, 0, -1, 0, 0, 0, 0, 0, 0)$$
$$c_{buono} = (0, 0, 0, -1, 0, 0, 0, 1, 0, 0)$$
$$c_{scuola} = (0, 0, 0, 0, -1, 0, 0, 0, 1, 0)$$
$$c_{fatelo} = (0, 1, 0, 0, 0, -1, 0, 0, 0, 0)$$
$$c_{capire} = (-1, 0, 0, 0, 0, 0, 0, 0, 0, 1)$$

In the second step, we build a semantic vector for each term by accumulating random vectors of its co-occurring words. For example fixing $m = 2$, the semantic vector for the word *scuola* is the sum of the random vectors *siete, buono, fatelo, capire*. Summing these vectors, the semantic vector for *scuola* results in $(-1, 1, 0, -2, 0, -1, 0, 1, 0, 1)$. This operation is repeated for all the sentences in the corpus and for all the words in V. In this example, we used very small vectors, but in a real scenario, the vector dimension ranges from hundreds to thousands of dimensions. In particular, in our experiment we use a vector dimension equals to 200 with 10 no-zero elements.

In order to compute the ac_i score for a word w_i in a tweet, we build a context vector c_{w_i} as the sum of random vectors assigned to words that co-occur with w_i in the tweet. Then we compare the cosine similarity between c_{w_i} and the semantic vector sv_i assigned to w_i. The idea is to measure how the semantic vector is dissimilar to the context vector. If the word w_i has never appeared in the context under analysis, its semantic vector does not contain the random vectors of the words in the context, this results in low cosine similarity. Finally, the divergence from the context is computed as $1 - cosSim(c_{w_i}, sv_i)$.

3 Evaluation

We perform the evaluation using the data provided by the task organizers. The number of tweets in the training set is 3,977, while the testing set consists of 872 tweets. The only parameter to set in LIBLINEAR is C (the cost), after a 5-fold cross validation on training we set C=1.

We submit two runs: UNIBA1 includes the semantic vector representing the tweet as a feature, while UNIBA2 does not include this vector. Nevertheless, features about the divergence are included in both the runs.

Official results are reported in Table 1. Our runs rank third and fourth in the final rank. Our team is classified as second since the first two runs in the rank belong to the team1. We can notice that runs are very close in the rank. The last run is ranked below the baseline *random*, while any system is ranked below the baseline *baseline-mfc* that assigns the most frequent class (*non-ironic*).

Results show that our system is not able to improve performance exploiting the distributional representation of tweets, since the two runs report the same average F1-score. We performed further experiments in order to understand the contribution of each feature. Some relevant outcomes are reported in Table 2, in particular:

- keyword-based features are able to achieve the best performance, in particular bigrams

team	precision (non-ironic)	recall (non-ironic)	F1-score (non-ironic)	precision (ironic)	recall (ironic)	F1-score (ironic)	average F1-score
team1	0.785	0.643	0.707	0.696	0.823	0.754	0.731
team1	0.751	0.643	0.693	0.687	0.786	0.733	0.713
UNIBA1	0.748	0.638	0.689	0.683	0.784	0.730	0.710
UNIBA2	0.748	0.638	0.689	0.683	0.784	0.730	0.710
team3	0.700	0.716	0.708	0.708	0.692	0.700	0.704
team6	0.600	0.714	0.652	0.645	0.522	0.577	0.614
random	0.506	0.501	0.503	0.503	0.508	0.506	*0.505*
team7	0.505	0.892	0.645	0.525	0.120	0.195	0.420
baseline-mfc	0.501	1.000	0.668	0.000	0.000	0.000	*0.334*

Table 1: Task results.

run	note	no-iro-F	iro-F	avg-F
run1	all	0.6888	0.7301	0.7095
run2	no DSM	0.6888	0.7301	0.7095
1	keyword	0.6738	0.6969	0.6853
2	keyword, bigrams	0.6916	0.7219	0.7067
3	keyword, bigrams, trigrams	0.6992	0.7343	0.7168
4	keyword, bigrams, trigrams, blog	0.7000	0.7337	0.7168
5	keyword, bigrams, trigrams, polarity	0.6906	0.7329	0.7117
6	keyword, bigrams, trigrams, context	0.6937	0.7325	0.7131
7	only DSM	0.6166	0.6830	0.6406
8	only context	0.4993	0.5587	0.5290

Table 2: Task results obtained combining different types of features.

and trigrams contribute to improve the performance (run 1 and 2);

- DSM features introduce some kind of noise when are combined with other features, in fact run 4, 5 and 6 achieve good performance without DSM;

- DSM alone without any other kind of features is able to achieve remarkable results, it is important to notice that in this run only the tweet vector is used as a feature;

- blog, polarity, and context features are not able to give a contribution to the overall system performance, however we can observe that using only context features (only three features for each tweet) we are able to overcome both the baselines.

Analyzing results we can conclude that a more effective way to combine distributional with no-distributional features is needed. We plan to investigate as a future work the combination of two different kernels for distributional and keyword-based features.

4 Conclusions

We propose a supervised system for detecting irony in Italian tweets. The proposed system exploits different kinds of features: keyword-based, microblogging features, polarity, distributional semantics features and a score that measure how a word is used in an unconventional context. The word divergence from its conventional context is computed exploiting the distributional semantics model build by the Random Indexing.

Results prove that our system is able to achieve good performance and rank third in the official ranking. However, a deep study on different combinations of features shows that keyword-based features alone are able to achieve the best result, while distributional features introduce noise during the training. This outcome suggests the need for a different strategy for combining distributional a no-distributional features.

References

Valerio Basile and Malvina Nissim. 2013. Sentiment analysis on italian tweets. In *Proc. of WASSA 2013*, pages 100–107.

Pierpaolo Basile and Nicole Novielli. 2014. Uniba at evalita 2014-sentipolc task: Predicting tweet sentiment polarity combining micro-blogging, lexicon and semantic features. In *Proc. of EVALITA 2014*, pages 58–63, Pisa, Italy.

Alessandra Teresa Cignarella, Simona Frenda, Valerio Basile, Cristina Bosco, Viviana Patti, and Paolo Rosso. 2018. Overview of the evalita 2018 task on irony detection in italian tweets (ironita). In Tommaso Caselli, Nicole Novielli, Viviana Patti, and Paolo Rosso, editors, *Proceedings of the 6th evaluation campaign of Natural Language Processing and Speech tools for Italian (EVALITA'18)*, Turin, Italy. CEUR.org.

Sanjoy Dasgupta and Anupam Gupta. 1999. An elementary proof of the Johnson-Lindenstrauss lemma. Technical report, Technical Report TR-99-006, International Computer Science Institute, Berkeley, California, USA.

Andrea Esuli and Fabrizio Sebastiani. 2006. Sentiwordnet: A publicly available lexical resource for opinion mining. In *Proc. of LREC*, pages 417–422.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Liblinear: A library for large linear classification. *Journal of machine learning research*, 9(Aug):1871–1874.

Emanuele Pianta, Luisa Bentivogli, and Christian Girardi. 2002. Multiwordnet: developing an aligned multilingual database. In *Proc. 1st Intl Conf. on Global WordNet*, pages 293–302.

Magnus Sahlgren. 2005. An Introduction to Random Indexing. In *Methods and Applications of Semantic Indexing Workshop at the 7th International Conference on Terminology and Knowledge Engineering, TKE*, volume 5.

Paul Smolensky. 1990. Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial Intelligence*, 46(1-2):159–216, November.

Cynthia Van Hee, Els Lefever, and Véronique Hoste. 2018. Semeval-2018 task 3: Irony detection in english tweets. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 39–50.

Eros Zanchetta and Marco Baroni. 2005. Morph-it!: a free corpus-based morphological resource for the italian language. *Proc. of the Corpus Linguistics Conf. 2005*.

Irony detection in tweets: X2Check at Ironita 2018

Emanuele Di Rosa
Chief Technology Officer
App2Check s.r.l.
emanuele.dirosa
@app2check.com

Alberto Durante
Research Scientist
App2Check s.r.l.
alberto.durante
@app2check.com

Abstract

English. In this paper we describe and show the results of the two systems that we have specifically developed to participate at Ironita 2018 for the irony detection task. We scored as the third team in the official ranking of the competition, thanks to the X2C-B system, at a distance of just 0.027 of F1 score from the best system.

Italiano. *In questo report descriviamo i due sistemi che abbiamo sviluppato ad hoc per partecipare ad Ironita 2018, nello specifico al task di irony detection. Il nostro team risultato essere il terzo classificato nella classifica ufficiale della competizione, grazie al nostro sistema X2C-B, che ha ottenuto un F1 score solo 0.027 inferiore rispetto al primo classificato.*

1 Introduction

In social media, the use of irony in tweets and Facebook posts is widely spread and makes very difficult for sentiment analysis tools to properly automatically classify people opinion (Hernández and Rosso, 2016). The ability to detect irony with high accuracy would bring an important contribution in opinion mining systems and lead to many industrial applications. For this reason, irony detection has been largely studied in recent research papers like (Farías et al., 2011), (Barbieri et al., 2014), (Farías et al., 2016), (Freitas et al., 2014).

In this paper we describe and show the results of the two systems that we have specifically developed to participate at Ironita 2018 (Cignarella et al., 2018) for the irony detection task. We scored as the third team in the official ranking of the competition, thanks to the X2C-B system, at a distance of just 0.027 of F1 score from the best system.

This paper is structured as follow: after the introduction we present the descriptions of our two systems submitted for the irony detection task; then we show and discuss the results on the official test set of the competition, finally we provide our conclusions.

2 Systems description

The dataset provided by Ironita organizers has been split into training set (80% of the documents) and development set (the remaining 20%). We randomly sampled the examples for each category, thus obtaining different sets for training/test set, by keeping the distribution of ironic and non-ironic samples through the two sets. We submitted two runs, as the results of the two different systems we developed for each category, called X2C-A and X2C-B. The former has been developed on top of the Scikit-learn library in Python language (Pedregosa et al., 2011), and the latter on top of the WEKA library (Frank et al., 2016) in JAVA language. In both cases, input text has been cleaned with a typical NLP pipeline, involving punctuation (with the exclusion of question/exclamation mark), numbers and stopwords removal. In particular, since it is still hard to detect irony in a text, very often also for humans, we tried to take advantage of features trying to help triggering the presence of irony. For instance, question and exclamation marks, text strings representing laughs, emoticons, mixed sentiment in the same sentence are some of the text features that we extracted from the text and represented with a specific explicit marker highlighting their presence.

Both the X2C-A and X2C-B unconstrained run were trained using the SENTIPOLC 2016 Irony training set and test set (Barbieri et al., 2016) as external source, in addition to the Ironita training set.

2.1 X2C-A

The X2C-A system has been created by applying an NLP pipeline including a vectorization of the collection of reviews to a matrix of token counts of bi-grams; then, the count matrix has been transformed to a normalized tf-idf representation (term-frequency times inverse document-frequency). For the training, we created an ensemble model, more specifically a voting ensemble, that takes into account three different algorithms: LinearSVC (an implementation of Support Vector Machines), Multinomial Naive Bayes and the SGD classifier. All of them have an implementation available in the Scikit-learn library. The ensemble model has been the best model in our model selection activity. In order to properly select the best hyper-parameters, we applied a grid search approach for each of the model in the voting ensemble. The resulting ensemble model showed a macro F1 score of 70.98 on our development set and is very close to the final result on the competition test set (shown in table).

	Acc	F1 ironic	Macro F1
LinearSVM	0.706	0.699	0.706
NB	0.706	0.699	0.706
SGD	0.697	**0.728**	0.693
Ensemble	**0.710**	0.709	**0.710**

Table 1: Results on the development set for X2C-A constrained.

2.2 X2C-B

In the model selection process, the two best algorithms have been Naive Bayes Multinomial and SMO, both using unigram features. We took into account the F1 score on the positive labels and the Macro-F1 in order to select the best algorithm. As shown in Table 2, Naive Bayes Multinomial reached a Macro F1 score 2.38% higher on the constrained run and a 14.2% on the unconstrained run, thus both the constrained and the unconstrained submitted runs were produced using this algorithm.

Comparing the results in Table 2 with the ones in Table 1, we can notice that X2C-B unconstrained reached the highest performance on the development set, while X2C-B constrained obtained the lowest score.

	F1 non-iro	F1 iro	Macro F1
NB-const	0.715	0.696	0.707
NB-uncon	**0.729**	**0.750**	**0.740**
SMO-const	0.678	0.689	0.683
SMO-uncon	0.704	0.492	0.598

Table 2: Results on development set for X2C-B.

3 Results and discussion

In Table 3 we show the results of our runs on the official test set of the competition. In accordance with what we noticed before, comparing Table 1 and Table 2, our best run is X2C-B unconstrained, which reached the best F1 overall on non-ironic documents; it also ranks fifth in the overall F1-score, at a distance of 0.027 from the best system. The performance of the X2C-A run is very similar to the unconstrained run, obtaining a F1-score that is only 0.002 higher than the constrained run. The difference between the two X2C-B runs is larger in relative terms, but is only of 0.021. We can also see that our X2C-B-u shows the best F1 score on the non-ironic tweets compared to all of the systems.

We added to this ranking also the model that reached the first position on the Irony task at SENTIPOLC 2016 (Di Rosa and Durante, 2016). The score of that model on this test set, called X2C2016 in the table, reached a F1-score of just 0.432, which is lower than the baseline of this year. This surprising result may indicate either that the irony detection systems had a great improvement in the past two years, or that irony detectors have a performance that is very much dependent on the topics treated in the training set, i.e. they are still not so good to generalize.

4 Conclusions

In this paper we described the two systems that we built and submitted for the Ironita 2018 competition for the irony detection task. The results show that our system X2C-B scored as the third team at a distance of just 0.027 of F1 score from the best system.

References

Alessandra Teresa Cignarella and Simona Frenda and Valerio Basile and Cristina Bosco and Viviana Patti and Paolo Rosso. 2018. *Overview of the Evalita 2018 Task on Irony Detection in Italian Tweets*

	team	F1 non-iro	F1 iro	F1
1	team 1	0.707	**0.754**	**0.731**
2	team 1	0.693	0.733	0.713
3	team 2	0.689	0.730	0.710
4	team 2	0.689	0.730	0.710
5	**X2C-B-u**	**0.708**	0.700	0.704
6	team 4	0.662	0.739	0.700
7	team 4	0.668	0.733	0.700
8	**X2C-A-u**	0.700	0.689	0.695
9	team 5	0.668	0.722	0.695
10	**X2C-A-c**	0.679	0.708	0.693
11	**X2C-B-c**	0.674	0.693	0.683
12	team 6	0.603	0.700	0.651
13	team 6	0.626	0.665	0.646
14	team 6	0.579	0.678	0.629
15	team 6	0.652	0.577	0.614
16	*baseline-1*	*0.503*	*0.506*	*0.505*
17	team 7	0.651	0.289	0.470
18	**X2C2016**	0.665	0.198	0.432
19	team 7	0.645	0.195	0.420
20	*baseline-2*	*0.668*	*0*	*0.334*

Table 3: Ironita 2018 official ranking.

(IronITA) in Proceedings of the 6th evaluation campaign of Natural Language Processing and Speech tools for Italian (EVALITA'18).

Francesco Barbieri and Valerio Basile and Danilo Croce and Malvina Nissim and Nicole Novielli and Viviana Patti. 2016. *Overview of the Evalita 2016 SENTIment POLarity Classification Task* in Proceedings of Third Italian Conference on Computational Linguistics (CLiC-it 2016) & Fifth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2016), Napoli, Italy, December 5-7, 2016.

Emanuele Di Rosa and Alberto Durante. 2016. *Tweet2Check evaluation at Evalita Sentipolc 2016* in Proceedings of Third Italian Conference on Computational Linguistics (CLiC-it 2016) & Fifth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2016), Napoli, Italy, December 5-7, 2016.

Eibe Frank, Mark A. Hall, and Ian H. Witten. 2016. The WEKA Workbench. Online Appendix for "Data Mining: Practical Machine Learning Tools and Techniques", Morgan Kaufmann, Fourth Edition, 2016.

Pedregosa, F. and Varoquaux, G. and Gramfort, A. and Michel, V. and Thirion, B. and Grisel, O. and Blondel, M. and Prettenhofer, P. and Weiss, R. and Dubourg, V. and Vanderplas, J. and Passos, A. and Cournapeau, D. and Brucher, M. and Perrot, M. and Duchesnay, E. 2011. *Scikit-learn: Machine Learning in Python* in Journal of Machine Learning Research, pp. 2825–2830.

Farías, Delia Irazú Hernández et al. *Irony Detection in Twitter: The Role of Affective Content.* 2011. in ACM Trans. Internet Techn. 16 (2016): 19:1-19:24.

Barbieri, Francesco and Horacio Saggion. 2014. *Modelling Irony in Twitter: Feature Analysis and Evaluation.* in LREC (2014).

Delia Irazú Hernández Farías, Viviana Patti, and Paolo Rosso. 2016. *Irony Detection in Twitter: The Role of Affective Content.* in ACM Transaction Internet Technology 16, 3, Article 19 (July 2016), pp. 1-24. DOI: https://doi.org/10.1145/2930663

Freitas, Larissa and Vanin, Aline and Hogetop, Denise and N. Bochernitsan, Marco and Vieira, Renata. 2014. *Pathways for irony detection in tweets.* in Proceedings of the ACM Symposium on Applied Computing. 10.1145/2554850.2555048.

Hernández I., Rosso P. 2016. *Irony, Sarcasm, and Sentiment Analysis*. Chapter 7 In: Sentiment Analysis in Social Networks, F.A. Pozzi, E. Fersini, E. Messina, and B. Liu (Eds.), Elsevier Science and Technology, pp. 113-128

Sulis E., Hernández I., Rosso P., Patti V., Ruffo G. 2016. *Figurative Messages and Affect in Twitter: Differences Between #irony, #sarcasm and #not.* In: Knowledge-Based Systems, vol. 108, pp. 132143

Aspie96 at IronITA (EVALITA 2018): Irony Detection in Italian Tweets with Character-Level Convolutional RNN

Valentino Giudice
Computer Science Department of the University of Turin
valentino.giudice96@gmail.com

Abstract

English. Irony is characterized by a strong contrast between what is said and what is meant: this makes its detection an important task in sentiment analysis. In recent years, neural networks have given promising results in different areas, including irony detection. In this report, I describe the system used by the Aspie96 team in the IronITA competition (part of EVALITA 2018) for irony and sarcasm detection in Italian tweets.

Italiano. *L'ironia è caratterizzata da un forte contrasto tra ciò che viene detto e ciò che si intende: questo ne rende la rilevazione un importante task nell'analisi del sentimento. In anni recenti, le reti neurali hanno prodotto risultati promettenti in aree diverse, tra cui la rilevazione dell'ironia. In questo report, descrivo il sistema utilizzato dal team Aspie96 nella competizione di IronITA (parte di EVALITA 2018) per la rilevazione dell'ironia e del sarcasmo in tweet italiani.*

1 Introduction

Irony is a rhetorical trope characterized by a strong contrast between what is literally said and what is really meant. Detecting irony trough automatic devices is, therefore, important for other tasks of text analysis too, such as sentiment analysis, as it strongly changes the meaning (and the sentiment) of what is said.

Sarcasm is defined in different ways in different contexts. One such definition is that it is a particular kind of irony: irony with a specific target to attack, more offensive and delivered with a hash tone.

IronITA (Irony Detection in Italian Tweets) (Cignarella et al., 2018), a shared task organized within EVALITA 2018, the 6th evaluation campaign of Natural Language Processing and Speech tools for Italian, has, as a purpose, irony and sarcasm detection in Italian tweets and considers sarcasm as a kind of irony. The competition contained two subtasks: subtask A was about labeling each tweet as either non ironic or ironic, whereas subtask B was about labeling each tweet as non ironic, ironic but not sarcastic or sarcastic. The training dataset was provided by the organizers: within it, the annotations specify, for each tweet, if it is ironic and if it is sarcastic, marking no non-ironic tweet as sarcastic. The non-annotated test dataset was given to the teams to annotate. Each team was allowed a maximum of 4 runs (attempts to annotating the dataset) for either task, each of which was ranked in the leaderboard. Taking part in subtask B implied taking part in subtask A with the same run annotations. Of the maximum of 4 runs for each task, each team was allowed 2 constrained runs (using no other dataset containing irony or sarcasm annotations of tweets or sentences but the provided one) and 2 unconstrained runs (where teams were allowed to use other training data), and taking part in a subtask with an unconstrained run implied taking part in the same subtask with a constrained run as well.

A separate leaderboard was provided for subtask A and subtask B. For each leaderboard, the F1-score of every run for each of the two (for subtask A) or three (for subtask B) classes is shown. The actual score of each run in either leaderboard is the arithmetical average of the F1-scores for all classes in the corresponding subtask.

In recent years, neural network have proven themselves to be a promising approach to various problems of text analysis, including irony detection; the following sections propose, for the task, the model used by the Aspie96 team: a multilayer

neural network for binary classification.

The proposed model was used by the team for an individual, constrained, run for the subtask B (therefore also taking part in subtask A with the same annotations). The results of the competition, as well as the details of the model, are described in the following sections.

2 Description of the System

The proposed system is a neural network composed as follows.

It begins with a series of unidimensional convolutional layers, followed by a bidirectional recurrent layer based on GRU units (Cho et al., 2014) (a variation of LSTM (Hochreiter and Schmidhuber, 1997)). The output of the bidirectional layer is then used as the input for a simple feed forward neural network, whose output is just one number between 0 and 1 (low numbers represent the negative class, whereas large numbers represent the positive class): the sigmoid activation function is used to ensure an output within the range. The purpose of the convolutional layers is to convert the input to a form more meaningful for the neural network and to recognize short sequences within the text, whereas the recurrent part of the network has the purpose of converting a sequence of vectors into an individual vector of high-level features.

To better understand the role of the convolutional layers, the first layer should be considered. Its main difference with the following ones is that it also has the purpose of reducing the depth of the input (which can be done without loss of information as the input vectors are rather sparse). Its inputs are in a sequence and every short subsequence is converted into an individual vector. This preserves the information in the order of the sequence since only a short subsequence is used to produce every output vector and, thus, each output vector depends on a specific (short) part of the input sequence. The sequence every output vector depends on is shifted by one for each of them. The reason to use convolutional layers is to provide a context for each character being encoded as the meaning and importance of a character in an alphabetical language depends on its surrounding ones as well. The output of the last convolutional layer, thus, is a sequence of vectors which is just shorter than the original one and still encodes the useful information of each individual character in the sequence, but provides, for each character, a context dependent on the surrounding ones. Each convolutional layer expands the amount of character each vector depends on, while still keeping important information encoded in the sequence (indeed, the context of each character also provides information about which information is most useful and has to be represented). The input produced for the recurrent layer is slightly smaller, preserves temporal information and, for each character, provides information depending on the context, which constitutes a higher level feature than the character itself. The benefit of using convolutional layers is that the kernel doesn't vary throughout the sequence. This is particularly useful because short sequences within the text might have the same or similar meanings regardless of their position.

The convolutional layers do not use any padding and, because of that, they slightly reduce the length of the sequence (the length of the input sequence is slightly larger than the length of the output sequence), but no subsampling layer is used. The width of each kernel is rather small, as is the depth of the output of each layer and they are both hyperparameters that can be decided arbitrarily (except, of course, for the depth of the output layer, which has to be 1). After the recurrent layer, no use was found in adding extra fully connected layers before the output to deal non-linearity, so it is followed only by a fully connected layer whose input is the output of the recurrent layer and whose output is the output of the neural network.

In order to have regularization, dropout layers between the layers of the network of above and a gaussian noise layer applied to the input are used. Their purpose is similar and is to make the neural network better able to generalize, even without a very big training dataset. Particularly, thanks to the gaussian noise applied to the input, during training, the same tweet, read multiple times, does not constitute the same input data and the alteration in the data is propagated through the network.

A visualization of the model is given in fig. 1. The depth of the data at every layer isn't shown for simplicity, thus each box is meant to represent a vector (rather than an individual value). The length of the input sequence would actually be larger, but only a part is shown for simplicity. The regularization layers aren't shown.

The input is represented as an array with length 140, where each element is a vector of flags whose

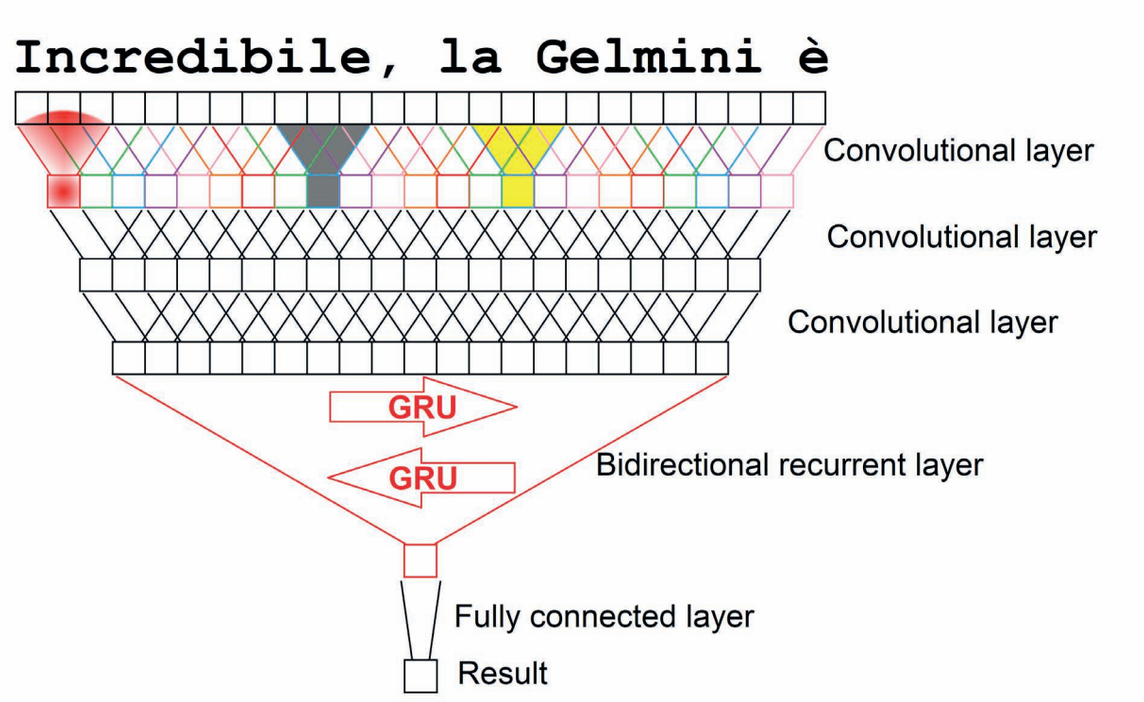


Figure 1: Visualization of the proposed model.

values are either 0 or 1. Each vector represents a character of the tweet: the tweet representation is either padded (by adding vectors with every flag at 0 at the beginning) or truncated (by ignoring its right part) in order to meet the length requirement. Most of the flags (namely 62 of them) are mutually exclusive and each of them represents a known character. Additional flags (namely 6) are used to represent properties of the current character (such as being an uppercase one). The total lenght of each vector is 68. Emojis are represented similarly to their Unicode (English) name, with no delimiters, and flags are used to mark which letters and underscores belong to the name of an emoji and which letter is the start of the name of an emoji. Spaces at the beginning or the end of the tweet and multiple spaces, as unknown characters, are ignored. The full list of known characters (excluding emojis) is as follows:

```
Space ! " # $ % & ' ( ) * + , - . / 0 1 2 3 4 5 6 7
8 9 : ; = ? @ [ ] _ a b c d e f g h i j k l m n o p q
r s t u v w x y z | ~
```

The subtask A is of binary classification: the model can be directly applied. The subtask B, however, has three classes. Since taking part in subtask B implied taking part in subtask A as well with the same annotations, the following approach was used: two identical copies of the model were created and trained, the purpose of one was to tell apart non ironic tweets and ironic ones and that of the other was to distinguish, among ironic tweets, which ones were not sarcastic and which ones were. In essence, the subtask B was seen as two separate problems of classification: one identical to subtask A and the second between non sarcastic and sarcastic ironic tweets.

The two identical models were trained individually by using only the data provided by the task organizers for the training phase: all data was used to train the model for the subtask A and only the ironic tweets were used to train the model for the second part of subtask B (detection of sarcasm in ironic examples).

The testing dataset was, therefore, annotated as follows: every tweet which was recognized by the first copy of the model as belonging to the negative class was labeled as non-ironic and non-sarcastic. Every tweet recognized in the positive class was labeled as ironic and also evaluated using the second copy of the model, marking it as non sarcastic when it was recognized in the negative class and as sarcastic otherwise. Therefore, no tweet was

Perceived class	Actual class		
	Non ironic	Ironic (non sarcastic)	Sarcastic
Non ironic	1008	453	295
Ironic (non sarcastic)	184	637	188
Sarcastic	138	449	227

Table 1: Confusion matrix from the 10-fold cross validation on the training dataset.

marked as sarcastic without being also marked as ironic.

3 Results

After training on the dataset provided by the organizers, an individual run was generated from the testing dataset and used for both subtask B and subtask A.

The proposed model ranked 9th among 17 runs (ignoring the baseline ones) in subtask A, as the 5th among 7 teams, considering the best run for each team. The F1-score for the negative class (non-ironic) was 0.668 (precision 0.742 and recall 0.606) and the F1-score for the positive class (ironic) was 0.722 (precision 0.666 and recall 0.789). The score was, thus, 0.695. This is consistent with the results obtained during the testing of the model (before the actual submission of the results), with a relatively low drop in the F1-scores. As a comparison, the first ranked team, with two runs, ranking first and second, got a score of 0.731 and 0.713.

In subtask B, the model ranked 5th among 7 runs (3rd among 4 teams), with the same F1-score for the neutral class (0.668), 0.438 for the ironic (non-sarcastic) class and 0.289 for the sarcastic class, with a score of 0.465.

A 10-fold cross validation using the training data produced the confusion matrix shown in Table 1.

The F1-score of the ironic (sarcastic and non sarcastic combined) class is 0.737. The F1-score for the ironic non-sarcastic class is 0.500 and the F1-score of the sarcastic class is 0.298. The F1-score for the negative class is, instead, 0.653 (for both subtasks). Therefore, using this data to compute the scores for the two subtasks, the score is 0.695 for the subtask A (the same as in the competition) and 0.484 for subtask B.

Table 2 shows, for the copy of the model trained to distinguish between irony and non irony, a few examples of tweets correctly and wrongly classified. These classifications were obtained using cross validation.

Similarly, Table 3 shows a few examples of tweets correctly and wrongly classified by the copy of the model trained to distinguish between ironic non sarcastic and sarcastic tweets, among those correctly detected as ironic.

4 Related work

A roughly similar model based on convolutions for text classification has been presented in 2014 (Kim, 2014) in the context of EMNLP. The model used word-level features (trough a pretrained and then fine-tuned embedding layer) as the input for an individual convolutional layer. The convolutional layer used multiple kernels of different sizes, to detect different high-level feature. To convert the output of the convolutional layer (whose depth was the number of its filters) into an individual vector of high-level features, a timewise max-pooling layer was used, producing a vector whose length was the same as the number of kernels in the convolutional layer (for each element in the resulting vector, its value was the maximum produced by the corresponding kernel along its input). The resulting vector was the input of a fully connected layer producing the output of the neural network. The model produced results better of those of the state of the art at the time on 4 out of 7 tasks.

In 2015 (Zhang et al., 2015), a model more similar to the one proposed was presented. The model used a character-level convolutional neural network for text classification, achieving competitive results. However, it did not use a recurrent layer (and was, in fact, compared with recurrent neural networks) and represented each input character as a one-hot encoded vector (without any additional flags). The model was trained using very big datasets (hundreds of thousands of instances, whereas only 3977 tweets were provided for IronITA) as this works better for character-level neural networks (because other kind of model often depend on pretrained layers and can, thus, use knowledge, for instance that related to the meaning of words, which couldn't be derived from the training dataset alone). Because of its structure and attributes, the model wasn't much flexible and easily adaptable to different kinds of usage.

5 Discussion

The system is different from others proposed in the past because it strictly works at character-level,

Tweet	Predicted class	Actual class
@matteorenzi ma quale buona scuola con un ministro incompetente? #gianninidimettiti miur ha combinato pasticcio su #concorsonazionale	Non ironic	Non ironic
#sfplm85bis #labuonascuola dopo 5 anni di sfp nel 2017, noi del NO che faremo? Fuori dal concorsone del 2015 e senza supplenze!	Non ironic	Non ironic
#Terremoto #Cile. 3 differenze con l'Italia: magnitudo superiore, rischio #Tsunami e nessuno che nell'emergenza incolpi i #migranti. #Natale	Non ironic	Ironic
Ma in italia non viene Obama a terrorizzarci nel caso di una vittoria del NO? #IoVotoNO #IoDicoNo	Non ironic	Ironic
Mario Monti senatore a vita?	Ironic	Non ironic
l'imbarbarimento è avvenuto perchè ai rom ormai è concesso tutto:rubano,schippano,ti entrano in casa e se ti difendi arrestano te #tagadala7	Ironic	Non ironic
SpecialiTG dopo attacchi terroristici: Sigla A)ISLAM è religione di Pace B)Attentatori eran depressi,omofobi,vittime bullismo o folli Sigla	Ironic	Ironic
Com'è che vince il no, Renzi si dimette ma i migranti arrivano ancora???? Perzone falzeeeee	Ironic	Ironic

Table 2: Examples of ironic and non ironic tweets classified by the proposed model.

Tweet	Predicted class	Actual class
#governo #Monti: ma non c'è nessun indagato fra i #ministri? Nemmeno fra i sottosegretari? E' possibile? In Italia?	Non sarcastic	Non sarcastic
@matteorenzi le risorse della scuola pubblica alle private... Questa è la buona scuola!	Non sarcastic	Non sarcastic
Incredibile, la Gelmini è entusiasta delle linee guida della riforma Renzi - Giannini. Chi lo avrebbe mai detto!? #labuonascuola	Non sarcastic	Sarcastic
#terroristaucciso oltre che nome e cognome dei due agenti,date anche gli indirizzi e i numeri di telefono, così li trovano prima .	Non sarcastic	Sarcastic
Qui nell'hinterland m.se siamo alla follia, posti di blocco dovunque, scene d'apocalisse zombie, rom inseguiti nei parchi #papamilano2017	Sarcastic	Non sarcastic
Passare da Berlusconi a Mario Monti è un salto troppo grosso. Ci vorrebbe almeno un governo di transizione presieduto da Checco Zalone	Sarcastic	Non sarcastic
#tfaordinario = morto che cammina. Anche quest'anno mi fate lavorare 18h...ma sono SENZA futuro grazie a #labuonascuola di @matteorenzi	Sarcastic	Sarcastic
Salvini,oltre a propagandare aggressività,riuscirà a superare il complesso del narciso felpato?Dopo immigrati,rom,si dedicherà ai politici?	Sarcastic	Sarcastic

Table 3: Examples of non sarcastic and sarcastic ironic tweets classified by the proposed model.

without any word features and because it doesn't use any data but what is provided for the specific task during learning, nor is it based on other, simpler, models to extract features. Many other models are instead based on feature-engineering and they, thus, often use layers (such as word embedding) pretrained on data different from that generated for the task. The results of the model in subtask A differ from those of the top ranked run by slightly less than 0.04 and by a maximum of 0.018 from all other runs in between. If other models are based on word-level features or on other simple and pretrained models extracting high level features, this suggests that good and similar results can be obtained by using strictly the data provided and without any word-level feature. The proposed model is not claimed to be the best possible with this properties; rather, it is an extremely simple attempt to the task. Other models may be built in the future which do not use any information outside of that provided in the learning dataset, but obtaining significantly better results than the proposed one. Still, the ranking position of the proposed model suggests that there is value in using knowledge outside of that available for a specific task, giving information about the language (and the meaning and sentiment of words and phrases): further research is needed to understand where the limits of strict character-level text analysis lay and the contexts in which it is a better or a worse solution.

References

Kyunghyun Cho, Bart van Merriënboer, Çağlar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, oct. Association for Computational Linguistics.

Alessandra Teresa Cignarella, Simona Frenda, Valerio Basile, Cristina Bosco, Viviana Patti, and Paolo Rosso. 2018. Overview of the evalita 2018 task on irony detection in italian tweets (ironita). In Tommaso Caselli, Nicole Novielli, Viviana Patti, and Paolo Rosso, editors, *Proceedings of the 6th evaluation campaign of Natural Language Processing and Speech tools for Italian (EVALITA'18)*, Turin, Italy. CEUR.org.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. 9:1735–80, 12.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751. Association for Computational Linguistics.

Xiang Zhang, Junbo Zhao Jake, and Yann LeCun. 2015. Character-level convolutional networks for text classification. *CoRR*, abs/1509.01626.

UO_IRO: Linguistic informed deep-learning model for irony detection

Reynier Ortega-Bueno
Center for Pattern Recognition and
Data Mining, Santiago de Cuba, Cuba
reynier.ortega@cerpamid.co.cu
Computer Science Department,
University of Oriente
reynier@uo.edu.cu

José E. Medina Pagola
University of Informatics Sciences
Havana, Cuba
jmedinap@uci.cu

Abstract

English. This paper describes our UO_IRO system developed for participating in the shared task IronITA, organized within EVALITA: 2018 Workshop. Our approach is based on a deep learning model informed with linguistic knowledge. Specifically, a Convolutional (CNN) and Long Short Term Memory (LSTM) neural network are ensembled, also, the model is informed with linguistics information incorporated through its second to last hidden layer. Results achieved by our system are encouraged, however a more fine-tuned hyper-parameters setting is required for improving the model's effectiveness.

Italiano. *Questo articolo descrive il nostro sistema UO_IRO, sviluppato per la partecipazione allo shared task IronITA, presso EVALITA 2018. Il nostro approccio si basa su un modello di deep learning con conoscenza linguistica. In particolare: una Convolutional Neural Network (CNN) e una Long Short Term Memory Neural Network (LSTM). Inoltre, il modello è arricchito da conoscenza linguistica, incorporata nel penultimo hidden layer del modello. Sebbene sia necessario un miglioramento a grana fine dei parametri per migliorare le prestazioni del modello, i risultati ottenuti sono incoraggianti.*

1 Introduction

Computers interacting with humans through language, in natural way, continues to be one of the most salient challenge for Artificial Intelligent researchers and practitioners. Nowadays, several basic tasks related to natural language comprehension have been effectively resolved. Notwithstanding, slight advances have been archived by the machines when figurative devices and creativity are used in language with communicative purposes. Irony is a peculiar case of figurative devices frequently used in real life communication. As human beings, we appeal to irony for expressing in implicit way a meaning opposite to the literal sense of the utterance (Attardo, 2000; Wilson and Sperber, 1992). Thus, understanding irony requires a more complex set of cognitive and linguistics abilities than literal meaning. Due to its nature, irony has important implications in sentiment analysis and other related tasks, which aim at recognizing feelings and emotions from texts. Considering that, detecting irony automatically from textual messages is an important issue to enhance sentiment analysis and it is still an open research problem (Gupta and Yang, 2017; Maynard and Greenwood, 2014; Reyes et al., 2013).

In this work we address the fascinating problem of automatic irony detection in tweets written in Italian language. Particularly, we describe our irony detection system (UO_IRO) developed for participating in IronITA 2018: Irony Detection in Italian Tweets (Cignarella et al., 2018a). Our proposed model is based on a deep learning model informed with linguistic information. Specifically, a CNN and an attention based LSTM neural network are ensembled, moreover, the model is informed with linguistic information incorporated through its second to last hidden layer. We only participated in Task A (irony detection). For that, two constrained runs and two unconstrained runs were submitted. The official results shown that our system obtains interesting results. Our best run was ranked in 12^{th} position out of 17 submissions. The paper is organized as follows. In Section 2, we introduce our UO_IRO system for irony detection. Experimental results are subsequently

discussed in Section 3. Finally, in Section 4, we present our conclusions and attractive directions for future work.

2 UO_IRO system for irony detection

The motivation for this work comes from two directions. In a first place, the recent and promising results found by some authors (Deriu and Cieliebak, 2016; Cimino and Dell'Orletta, 2016; González et al., 2018; Rangwani et al., 2018; Wu et al., 2018; Peng et al., 2018) in the use of convolutional networks and recursive networks, also the hybridization of them for dealing with figurative language. The second direction is motivated by the wide use of linguistic features manually encoded which have showed to be good indicators for discriminating among ironic and non ironic content (Reyes et al., 2012; Reyes and Rosso, 2014; Barbieri et al., 2014; Farías et al., 2016; Farías et al., 2018).

Our proposal learns a representation of the tweets in three ways. In this sense, we propose to learn a representation based on a recursive network with the purpose of capturing long dependencies among terms in the tweets. Moreover, a representation based on convolutional network is considered, it tries to encode local and partial relation between words which are near among themselves. The last representation is based on linguistic features which are calculated for the tweets. After that, all linguistic features previously computed are concatenated in a one-dimensional vector and it is passed through a dense hidden layer which encodes the linguistic knowledge and includes this information to the model.

Finally, the three neural network based outputs are combined in a merge layer. The integrated representations is passed to a dense hidden layer and the final classification is performed by the output layer, which use a softmax as activation function for predicting ironic or not ironic labels. For training the complete model we use categorical cross-entropy as loss function and the Adam method (Kingma and Ba, 2014) as the optimizer, also, we use a batch size of 64 and training the model for 20 epochs. Our proposal was implemented using the Keras Framework[1]. The architecture of the UO_IRO is shown in Figure 1 and described below.

[1]https://keras.io/

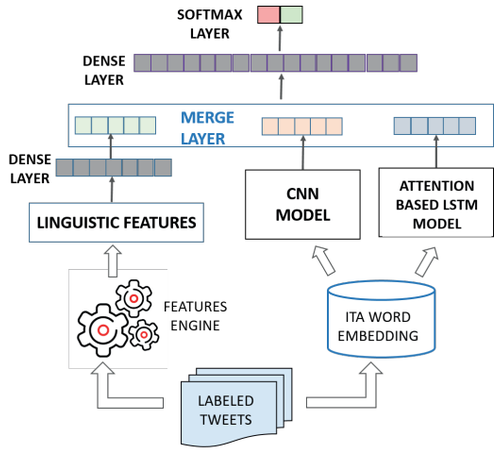


Figure 1: Overall Architecture of UO_IRO: Irony Detection System.

2.1 Preprocessing

In the preprocessing step, the tweets are cleaned. Firstly, the emoticons, urls, hashtags, mentions and twitter-specific tokens (RT for retweet and FAV for favorite) are recognized and replaced by a corresponding wild-card which encodes the meaning of these special words. Afterwards, tweets are morphologically analyzed by FreeLing (Padró and Stanilovsky, 2012). In this way, for each resulting token, its lemma is assigned. Then, the words in the tweets are represented as vectors using a word embedding model. In this work we use the Italian pre-trained vectors[2] public available (Bojanowski et al., 2017).

2.2 Attention Based LSTM

We use a model that consists in a Bidirectional LSTM neural network (Bi-LSTM) at the word level. Each time step t, the Bi-LSTM gets as input a word vector w_t with syntactic and semantic information known as word embedding. The idea behind this Bi-LSTM is to capture long-range and backwards dependencies in the tweets. Afterward, an attention layer is applied over each hidden state h_t. The attention weights are learned using the concatenation of the current hidden state h_t of the Bi-LSTM and the past hidden state s_{t-1}. The goal of this layer is then to derive a context vector c_t that captures relevant information for feeding it as input to the next level. Finally, a LSTM layer is stacked at the top. This network at each time step receives the context vector c_t which is propagated

[2]https://s3-us-west-1.amazonaws.com/fasttext-vectors/wiki.it.zip

until the final hidden state s_{T_x}. This vector (s_{T_x}) can be considered as a high level representation of the tweet. For more details, please see (Ortega-Bueno et al., 2018).

2.3 Convolutional Neural Network

We use a CNN model that consists in 3 pairs of convolutional layers and pooling layers in this architecture. Filters of size three, four and five were defined for the convolutional layers. In case of pooling layer, the maxpooling strategy was used. We also use the Rectified Linear Unit (ReLU), Normalization and Dropout methods to improve the accuracy and generalizability of the model.

2.4 Linguistic Features

In our work, we explored some linguistic features useful for irony detection in texts which can be grouped in three main categories: Stylistic, Structural and Content, and Polarity Contrast. We define a set of features distributed as follows:

Stylistic Features

- *Length*: Three different features were considered: number of words, number of characters, and the means of the length of the words in the tweet.
- *Hashtags*: The amount of hashtags.
- *Urls*: The number of url.
- *Emoticons*: The number of emoticons.
- *Exclamations*: Occurrences of exclamation marks.
- *Emphasized Words*: Four different features were considered: word emphasized through repetition, capitalization, character flooding and exclamation marks.
- *Punctuation Marks*: The frequency of dots, commas, semicolons, and question marks.
- *Quotations*: The number of expressions between quotation marks.

Structural and Content Features

- *Antonyms*: This feature considers the number of pairs of antonyms existing in the tweet. WordNet (Miller, 1995) antonym relation was used for that.
- *Lexical Ambiguity*: Three different features were considered using WordNet: the first one is the mean of the number of synsets of each word. The second one is the greatest number of synsets that has a single word. The last is the difference between the number of synsets of the word with major number of synsets and the average number of synsets.
- *Domain Ambiguity*: Three different features were considered using WordNet: the first one is the mean of the number of domains of each word. The second one is the greatest number of domains that a single word has in the tweet. The last one is the difference between the number of domains of the word with major number of domains and the average number of domains. It is important to clarify that the resources WordNet Domains[3] and SUMO[4] were separately used.
- *Persons*: This feature tries to capture verbs conjugated in the first, second, third person and nouns and adjectives which agree with such conjugations.
- *Tenses*: This feature tries to capture the different verbal tenses used in the tweet.
- *Questions-answers*: Occurrences of questions and answers pattern in the tweet.
- *Part of Speech*: The number of nouns, verbs, adverbs and adjectives in the tweet are quantified.
- *Negation*: The amount of negation words.

Polarity Contrast Features

With the purpose of capturing some types of explicit polarity contrast we consider the set of features proposed in (Peña et al., 2018). The Italian polarity lexicon (Basile and Nissim, 2013) was used to determine the contrast between different parts of the tweet.

- *WordPolarityContrast*: It is the polarity difference between the most positive and the most negative word in the tweet. This feature, also consider the distance, in terms of tokens, between the words.
- *EmotiTextPolarityContrast*: It is the polarity contrast between the emoticons and the words in the tweet.
- *AntecedentConsequentPolarityContrast*: This considers the polarity contrast between two parts of the tweet, when it is split by a delimiter. In this case, adverbs and punctuation marks were used as delimiters.
- *MeanPolarityPhrase*: It is the mean of the polarities of the words that belong to quotes.
- *PolarityStandardDeviation*: It is the standard deviation of the polarities of the words that

[3] http://wndomains.fbk.eu/hierarchy.html
[4] http://www.adampease.org/OP/

belong to quotes.

- *PresentPastPolarityContrast*: It computes the polarity contrast between the parts of the tweet written in present and past tense.
- *SkipGramPolarityRate*:It computes the rate among skip-grams with polarity contrast and all valid skip-grams. The valid skip-grams are those composed by two words (nouns, adjectives, verbs, adverbs) with skip=1. The skip-grams with polarity opposition are those that match with the patterns *positive-negative*, *positive-neutral*, *negative-neutral*, and vise versa.
- *CapitalLetterTextPolarityContrast*: It computes the polarity contrast between capitalized words and the rest of the words in the tweets.

3 Experiments and Results

In this section we show the results of the proposed model in the shared task of "Irony Detection" and discuss them. In a first experiment we analyze the performance of four variants of our model using 10 fold cross-validation strategy on the training set. Also, each variant was running in unconstrained and constrained setting, respectively. In Table 1, we summarize the obtained results in terms of F1 measure macro averaged (F1-AVG). Specifically, we rely on the macro for preventing systems biased towards the most populated classes.

Table 1: Results obtained by UO_IRO on the training set by using 10-fold cross-validation.

Run	**Model**	**AVG-F_1**
	Constrained	
run1-c	CNN-LSTM	0.7019
run2-c	CNN-LSTM-SVM	0.6927
run3-c	*CNN-LSTM-LING*	**0.7124**
run4-c	CNN-LSTM-LING-SVM	0.7040
	Unconstrained	
run1-u	CNN-LSTM	0.7860
run2-u	CNN-LSTM-SVM	0.7900
run3-u	*CNN-LSTM-LING*	**0.8226**
run4-u	CNN-LSTM-LING-SVM	0.8207

For the run1-c and run1-u (CNN-LSTM) we only combine the representation obtained by the attention based LSTM model with the CNN model, in these runs, no linguistic knowledge was considered. Run2-c and run2-u (CNN-LSTM-SVM) are a modification of the CNN-LSTM model, in this case we change the softmax layer at the output of the model and use a Linear Support Vector Machine (SVM) with default parameters as final classifier. Run3-c and run3-u (CNN-LSTM-LING) represent the original introduced model without any variations. Finally, for run4-c and run4-u (CNN-LSTM-LING-SVM) we change the softmax layer by a linear SVM as final classifier. For unconstrained runs, we include the ironic tweets provided by the corpus Twittirò (Cignarella et al., 2018b), to the official training set releases by the IronITA organizers.

Analyzing Table 1, several observations can be made. Firstly, unconstrained runs achieved better results than constrained ones. These results reveal that introducing more ironic examples improves the performance of the UO_IRO. Secondly, the results achieved with the variants that consider the linguistic knowledge (run3-c, run4-c, run3-u and run4-u) obtain an increase in the effectiveness. With respect to the strategy used for the final classification of the tweets, generally, those variants that use SVM obtain a slight drop in the AVG-F_1.

Regarding the official results, we submitted four runs, two for constrained setting (RUN1-c and RUN2-c) and two for unconstrained setting (RUN3-u and RUN4-u). For the unconstrained variants of the UO_IRO, the tweets provided by the corpus Twittirò were also used with the training set. Taking into account the results of the Table 1 we select to CNN-LSTM-LING (RUN1-c and RUN3-u) and CNN-LSTM-LING-SVM (RUN2-c and RUN4-u) as the most promising variants of the model for evaluating in the official test set.

As can be observed in Table 2, our four runs were ranked 12^{th}, 13^{th}, 14^{th} and 15^{th} from a total of 17 submissions. The unconstrained variants of the UO_IRO achieved better results than constrained ones. Contrary to the results shown in the Table 1, the runs that use SVM as final classification strategy (RUN2-c and RUN4-u) were better ranked than the other ones. We think that this behavior may be caused by softmax classifiers (last layer of the UO_IRO), those are more sensitive to the over-fitting problem than Support Vector Machines. Notice that, in all cases our model surpass the two baseline methods established by the organizers.

Table 2: Official results for the Irony Detection subtask.

Rank	Runs	F_1-I	F_1-noI	Avg-F_1
12/17	*RUN4-u*	**0.700**	**0.603**	**0.651**
13/17	RUN3-u	0.665	0.626	0.646
14/17	RUN2-c	0.678	0.579	0.629
15/17	RUN1-c	0.577	0.652	0.614

4 Conclusions

In this paper we presented the UO_IRO system for the task of Irony Detection in Italian Tweets (IronITA) at EVALITA 2018. We participated in the "Irony classification' subtask and our best submission ranked 12^{nd} out of 17. Our proposal combines attention-based Long Short-Term Memory Network, Convolutional Neural Network, and linguistics information which is incorporated through the second to last hidden layer of the model. The results shown that the consideration of linguistic features in combination with the deep representation learned by the neural network models obtain better effectiveness based on F1-measure. Results achieved by our system are interesting, however a more fine-tuned hyper-parameters setting is required for improving the model's effectiveness. We think that including the linguistic features of irony into the firsts layers of the model could be a way to increase the effectiveness. We would like to explore this approach in the future work. Also, we plan to analyze how affective information flows through the tweets, and how it impacts on the irony realization.

References

Salvatore Attardo. 2000. Irony as relevant inappropriateness. *Journal of Pragmatics*, 32(6):793–826.

Francesco Barbieri, Horacio Saggion, and Francesco Ronzano. 2014. Modelling Sarcasm in Twitter, a Novel Approach. In *Proceedings ofthe 5th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 136–141.

Valerio Basile and Malvina Nissim. 2013. Sentiment analysis on Italian tweets. In *4th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 100–107.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching Word Vectors with Subword Information. *Transactions of the ACL.*, 5:135–146.

Alessandra Cignarella, Frenda Simona, Basile Valerio, Bosco Cristina, Patti Viviana, and Rosso Paolo. 2018a. Overview of the EVALITA 2018 Task on Irony Detection in Italian Tweets (IronITA). In Tommaso Caselli, Nicole Novielli, Viviana Patti, and Paolo Rosso, editors, *Proceedings of Sixth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2018)*, Turin, Italy. CEUR.org.

Alessandra Teresa Cignarella, Cristina Bosco, Viviana Patti, and Mirko Lai. 2018b. Application and Analysis of a Multi-layered Scheme for Irony on the Italian Twitter Corpus TWITTIRÓ. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, pages 4204–4211.

Andrea Cimino and Felice Dell'Orletta. 2016. Tandem LSTM-SVM Approach for Sentiment Analysis. In *CLiC-it/EVALITA. 2016*, pages 1–6. CEUR-WS.org.

Jan Deriu and Mark Cieliebak. 2016. Sentiment Analysis using Convolutional Neural Networks with Multi-Task Training and Distant Supervision on Italian Tweets. In *CLiC-it/EVALITA. 2016*, pages 1–5. CEUR-WS.org.

Delia Irazú Hernańdez Farías, Viviana Patti, and Paolo Rosso. 2016. Irony Detection in Twitter. *ACM Transactions on Internet Technology*, 16(3):1–24.

Delia-Irazú Hernádez Farías, Viviana Patti, and Paolo Rosso. 2018. ValenTO at SemEval-2018 Task 3 : Exploring the Role of Affective Content for Detecting Irony in English Tweets. In *Proceedings ofthe 12th International Workshop on Semantic Evaluation (SemEval-2018)*, pages 643–648. Association for Computational Linguistics.

José Angel Gonález, Lluís-F. Hurtado, and Ferran Pla. 2018. ELiRF-UPV at SemEval-2018 Tasks 1 and 3 : Affect and Irony Detection in Tweets. In *Proceedings ofthe 12th International Workshop on Semantic Evaluation (SemEval-2018)*, pages 565–569.

Raj Kumar Gupta and Yinping Yang. 2017. CrystalNest at SemEval-2017 Task 4 : Using Sarcasm Detection for Enhancing Sentiment Classification and Quantification. In *Proceedings ofthe 11th International Workshop on Semantic Evaluations (SemEval-2017)*, pages 626–633, Vancouver, Canada. Association for Computational Linguistics.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Diana Maynard and Mark A Greenwood. 2014. Who cares about sarcastic tweets ? Investigating the impact of sarcasm on sentiment analysis. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*. European Language Resources Association.

George A Miller. 1995. WordNet: a lexical database for English. *Communications of the ACM*, 38(11):39–41.

Reynier Ortega-Bueno, Carlos E Mu, and Paolo Rosso. 2018. UO_UPV : Deep Linguistic Humor Detection in Spanish Social Media. In *Proceedings of the Third Workshop on Evaluation of Human Language Technologies for Iberian Languages (IberEval 2018)*, pages 1–11.

Lluís Padró and Evgeny Stanilovsky. 2012. FreeLing 3.0: Towards Wider Multilinguality. In *Proceedings of the (LREC 2012)*.

Anakarla Sotolongo Peña, Leticia Arco García, and Adrián Rodríguez Dosina. 2018. Detección de ironía en textos cortos enfocada a la minería de opinión. In *IV Conferencia Internacional en Ciencias Computacionales e Informáticas (CICCI' 2018)*, number 1-10, Havana, Cuba.

Bo Peng, Jin Wang, and Xuejie Zhang. 2018. YNU-HPCC at SemEval-2018 Task 3 : Ensemble Neural Network Models for Irony Detection on Twitter. In *622 Proceedings ofthe 12th International Workshop on Semantic Evaluation (SemEval-2018)*, pages 622–627. Association for Computational Linguistics.

Harsh Rangwani, Devang Kulshreshtha, and Anil Kumar Singh. 2018. NLPRL-IITBHU at SemEval-2018 Task 3 : Combining Linguistic Features and Emoji Pre-trained CNN for Irony Detection in Tweets. In *Proceedings ofthe 12th International Workshop on Semantic Evaluation (SemEval-2018)*, pages 638–642. Association for Computational Linguistics.

Antonio Reyes and Paolo Rosso. 2014. On the difficulty of automatically detecting irony: beyond a simple case of negation. *Knowledge and Information Systems*, 40(3):595–614.

Antonio Reyes, Paolo Rosso, and Davide Buscaldi. 2012. From humor recognition to irony detection: The figurative language of social media. *Data and Knowledge Engineering*, 74:1–12.

Antonio Reyes, Paolo Rosso, and Tony Veale. 2013. A multidimensional approach for detecting irony in Twitter. *Language Resources and Evaluation*, 47(1):239–268.

Deirdre Wilson and Dan Sperber. 1992. On verbal irony. *Lingua*, 87(1):53–76.

Chuhan Wu, Fangzhao Wu, Sixing Wu, Junxin Liu, Zhigang Yuan, and Yongfeng Huang. 2018. THU NGN at SemEval-2018 Task 3 : Tweet Irony Detection with Densely Connected LSTM and Multi-task Learning. In *Proceedings ofthe 12th International Workshop on Semantic Evaluation (SemEval-2018)*, pages 51–56. Association for Computational Linguistics.

CapetownMilanoTirana for GxG at Evalita2018. Simple n-gram based models perform well for gender prediction. Sometimes.

Angelo Basile
Symanto Research
angelo.basile@symanto.net

Gareth Dwyer
CoGrammar
garethdwyer@gmail.com

Chiara Rubagotti
Independent Researcher
chiara.rubagotti@gmail.com

Abstract

In this paper we describe our participation in the Evalita 2018 GxG cross-genre/domain gender prediction shared task for Italian. Building on previous results obtained on in-genre gender prediction, we try to assess the robustness of a linear model using n-grams in a cross-genre setting. We show that performance drops significantly when the training and testing genres differ. Furthermore, we experiment with abstract features in trying to capture genre-independent features. We achieve an average F1-score of 0.55 on the official in-genre test set — being thus ranked first out of five submissions — and 0.51 on the cross-genre test set.

In questo articolo presentiamo il nostro contributo per lo shared task GxG di Evalita 2018 per l'analisi predittiva del genere di un autore su corpora di dominî diversi. Usiamo un modello basato su una macchina a vettori supporto con kernel lineare che usa gli n-grammi come feature. Il modello, che ha ottenuto in passato risultati eccellenti nella predizione di genere quando allenato e valutato all'interno di un unico dominio (Twitter), crolla significativamente nella performance in questo lavoro, anche quando usato in combinazione con una serie di feature astratte. Sul test set ufficiale il nostro modello raggiunge una F1-score pari a 0.55 (permettendoci di piazzarci in cima alla classifica) nel contesto in-genre e 0.51 nel contesto cross-genre.

1 Introduction

Gender prediction is the task of profiling authors to infer their gender based on their writing. This task has been carried out so far with success within one-genre/single-domain data sets, reaching state-of-the-art accuracy of 85% on English tweets (Basile et al., 2017). Cross-domain gender classification, on the other hand, has proven to be more difficult, with state-of-the-art accuracy halting at 60% (Medvedeva et al., 2017).

The theoretical assumption behind gender prediction from text is language variation: the same meaning can be expressed in different forms and this variation can be explained in terms of social variables, such as social status, personality, age and, indeed, gender (Labov, 2006; Verhoeven et al., 2016; Johannsen et al., 2015).

Proof of significant variation in language use between men and women has been found at the morpho-syntactic level (Johannsen et al., 2015) and indeed syntactic features have been used effectively for gender attribution (Sarawgi et al., 2011). Using syntax for attribution tasks has the benefit of modelling the problem in a space which is more resilient to topic and genre effects. However, we do not experiment here with deep syntactic features, but instead we try to leverage surface and frequency-based features.

In this paper we use a model built, trained, and tested for gender prediction on a single domain (i.e. Twitter). Instead of experimenting with new techniques, our aim is to sound the existing model's resilience in the different context of a cross-genre train and test setting (**RQ1**). Since we expect our model to fail on this task, we set out to design an experiment using a set of abstract features that has recently been applied for cross-lingual gender prediction (van der Goot et al., 2018): this way we want to investigate if surface features can be used to mitigate topic and genre effects (**RQ2**).

We organise this work as follows: in Section 2 we present an overview of the data set released by the organisers; in Section 3 we describe the exper-

imental set up, the model and the features used; in Section 4 we give an overview of the results and finally we conclude our work in Section 5.

The contributions of this work for the GxG task for EVALITA 2018 are the following:

- we test if a gender prediction model achieving state-of-the-art performances when trained and tested on the same domain can also achieve good results when tested in a cross-domain setting
- we experiment with abstract features in order to factor out domain-dependent effects
- we release all the code for further reproducibility at `https://github.com/anbasile/gxg-partecipation`

Task description The GxG task is a document classification task. Given a document belonging to a given genre, we have to predict the gender of the author. Thus the task is a binary classification task. The task is composed by two sub-tasks: in-genre prediction and cross-genre prediction. In the first case, the training set and the test set belong to the same genre. In the cross-genre sub-task, on the other hand, models will be trained on four genres and then tested on the single genre which they have not been exposed to during training.

2 Data

We use only the data released by the task organisers: that is, texts from five different genres. The genres are as follows.

- YouTube comments
- Tweets
- Children's writing
- News
- Personal diaries

From the data description given by the organisers we know that one author can possibly have authored multiple documents. We provide an overview of the data set in Table 1.

Data set	F	M	Tokens
Children	100	100	65986
Diaries	100	100	82989
Journalism	100	100	113437
Twitter	3000	3000	101534
Youtube	2200	2200	90639

Table 1: Overview of number of instances and corpus size per genre and label distribution.

3 Experiments

In this section we describe the feature extraction process and the model that we built. We develop one single model and train it in ten different configurations, as required by the task assignment. We train and test on five different domains, in-domain and across domains.

3.1 Pre-processing

We decide not to pre-process the data in any way, since we have no linguistic (nor non-linguistic) reasons for doing so. As a tokenization strategy for the lexicalized models we simply split on all white space tokens. For building the abstract feature representation we use `spaCy's` Italian tokenizer (Honnibal and Johnson, 2015).

3.2 Model and Features

We build a sparse linear model for approaching this task.

As features we use n-grams extracted at the word level as well as at the character level. We use 3-10 n-grams and binary TF-IDF. We feed these features to a Support Vector Machine (SVM) model with a linear kernel; we use the implementation included in `scikit-learn` (Pedregosa et al., 2011). This model in this same configuration has achieved excellent results during the PAN 2017 evaluation campaign (Potthast et al., 2017; Basile et al., 2017).

Furthermore, we experiment with feature abstraction: we follow the bleaching approach recently proposed by (van der Goot et al., 2018). First, we transform each word into a list of symbols that 1) represents the shape of the individual characters and 2) abstracts from meaning by still approximating the vowels and characters that compose the word; then, we compute the length of the word and its frequency (while taking care of padding the first one with a zero in order to

	IN					CROSS				
	DI	YT	TW	CH	JO	DI	YT	TW	CH	JO
Words (W)	0.73	0.60	0.73	0.55	0.65	0.64	0.53	0.55	0.58	0.55
Chars (C)	0.68	0.62	0.74	0.52	0.54	0.67	0.56	0.54	0.59	0.52
W+C	0.70	0.62	0.74	0.54	0.62	0.62	0.57	0.52	0.60	0.56
Bleaching	0.67	0.59	0.67	0.53	0.54	0.53	0.53	0.50	0.53	0.53

Table 2: Accuracy results on the training set per genre (DI (diaries), YT (YouTube), TW (Twitter), CH (children writing), JO (journalism)), in-genre and cross-genre. Scores are obtained via cross-validation. The cross-genre results are obtained by training on everything but the tested genre.

avoid feature collision); finally, we use a boolean label for explicitly distinguishing words from non-alphanumeric tokens (e.g. emojis). Table 3 shows an example of this feature abstraction process.

	SHAPE	FREQ	LEN	ALPHA
Questo	Cvvccv	46	06	True
è	v	650	01	True
solo	cvcv	116	04	True
un	vc	1	02	True
esempio	vcvccvv	1	07	True
.	.	60	01	False
😎	😎	1	01	False

Table 3: An illustration of the bleaching process.

(van der Goot et al., 2018) proposed this bleaching approach for successfully modelling gender across languages, by leveraging the language-independent nature of these features: here, we test if this approach is sound for mitigating the genre effect on the model.

4 Evaluation and Results

Since the data set labels are evenly distributed across the two classes, we use accuracy to evaluate our model. First, we report results obtained via a 10-fold cross-validation on the training set; then, we report results from the official test set, whose labels have been released.

4.1 Development Results

We report the development results obtained by using different text representations. Table 2 presents an overview of these results. Overall, we see that all the different feature representation formats lead to comparable results; the combination of words and characters seems to be the best combination. This is the same combination that we use for the bleached representation.

4.2 Test Results

We present official test results in Table 4. We submitted only one run. For one genre (*Diaries*) we obtained exactly the same score in both the in- and cross-genre settings: this outcome is extremely unlikely, however we ran the models several times and inspected the code for bugs and yet the results remained identical. In the cross-genre setting, we did not tune the hyper-parameters of our model considering the target genre.

The overview of the results is puzzling. First, compared to related in-domain work (Potthast et al., 2017), the overall performance is considerably lower, even taking into account domain variance. Second, the drop in performance from the in-genre to the cross-genre setting is not as high as expected. Third, even in the in-genre setting the difference in performance between genres is not trivial and it seems to be independent from training corpus's size: the two social network domains are considerably bigger in size and yet the testing scores are lower when compared to other genres.

GENRE	IN		CROSS	
	acc.	f1	acc.	f1
Diaries	0.635	0.624	0.635	0.624
YouTube	0.547	0.527	0.503	0.461
Twitter	0.545	0.542	0.555	0.540
Children	0.615	0.614	0.535	0.533
Journalism	0.480	0.460	0.515	0.381
Average	0.564	0.553	0.548	0.507

Table 4: Official test results

5 Conclusions

We presented our participation to the GxG cross-genre gender prediction task and we obtained good

results using a simple system. On top of that, we experimented with abstract features and got sub-optimal results.

Based on our experiment with a gender-prediction model which obtained state-of-the-art in-domain performance in the past, we conclude that genre plays a crucial role in gender prediction: not only genre, but the notion of *variety space*, as instructed by (Plank, 2016), should be taken into consideration for a fuller account of social variability and for building more robust systems (**RQ1**). We then attempted to improve our stock model using abstract, delexicalized features, but we failed to demonstrate any substantial improvement (**RQ2**). Furthermore, from our results it emerges that not all the examined genres pose the same challenges for gender prediction purposes: in some genres, namely journalism, the personality of the author, whether male or female, is hedged by the domain style, which favours objectivity and neutrality over self-expression and abandonment in writing. Therefore we suspect that genre-inherent style elements might make it harder for the model to carry out effective profiling of the author (be it for gender or for other social variables).

Recently, Variational Auto Encoders (VAE) (Kingma and Welling, 2013) are emerging as a good tool for properly modelling language in presence of latent variables: we plan to investigate the effectiveness of VAEs in predicting gender while modelling genre as a latent variable.

References

Angelo Basile, Gareth Dwyer, Maria Medvedeva, Josine Rawee, Hessel Haagsma, and Malvina Nissim. 2017. N-gram: New groningen author-profiling model. *arXiv preprint arXiv:1707.03764*.

Matthew Honnibal and Mark Johnson. 2015. An improved non-monotonic transition system for dependency parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1373–1378, Lisbon, Portugal, 9. Association for Computational Linguistics.

Anders Johannsen, Dirk Hovy, and Anders Søgaard. 2015. Cross-lingual syntactic variation over age and gender. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*, pages 103–112.

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

William Labov. 2006. *The social stratification of English in New York city*. Cambridge University Press.

Maria Medvedeva, Hessel Haagsma, and Malvina Nissim. 2017. An analysis of cross-genre and in-genre performance for author profiling in social media. In *International Conference of the Cross-Language Evaluation Forum for European Languages*, pages 211–223. Springer.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Barbara Plank. 2016. What to do about non-standard (or non-canonical) language in nlp. *arXiv preprint arXiv:1608.07836*.

Martin Potthast, Francisco M. Rangel Pardo, Michael Tschuggnall, Efstathios Stamatatos, Paolo Rosso, and Benno Stein. 2017. Overview of pan'17 - author identification, author profiling, and author obfuscation. In *CLEF*.

Ruchita Sarawgi, Kailash Gajulapalli, and Yejin Choi. 2011. Gender attribution: tracing stylometric evidence beyond topic and genre. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*, pages 78–86. Association for Computational Linguistics.

Rob van der Goot, Nikola Ljubešić, Ian Matroos, Malvina Nissim, and Barbara Plank. 2018. Bleaching text: Abstract features for cross-lingual gender prediction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 383–389.

Ben Verhoeven, Walter Daelemans, and Barbara Plank. 2016. Twisty: a multilingual twitter stylometry corpus for gender and personality profiling. In *Proceedings of the 10th Annual Conference on Language Resources and Evaluation (LREC 2016)/Calzolari, Nicoletta [edit.]; et al.*, pages 1–6.

A Markovian Kernel-based Approach for itaLIan Speech acT labEliNg

Danilo Croce and **Roberto Basili**
University of Roma, Tor Vergata
Via del Politecnico 1, Rome, 00133, Italy
{croce,basili}@info.uniroma2.it

Abstract

English. This paper describes the UNITOR system that participated to the *itaLIan Speech acT labEliNg* task within the context of EvalIta 2018. A Structured Kernel-based Support Vector Machine has been here applied to make the classification of the dialogue turns sensitive to the syntactic and semantic information of each utterance, without relying on any task-specific manual feature engineering. Moreover, a specific Markovian formulation of the SVM is adopted, so that the labeling of each utterance depends on speech acts assigned to the previous turns. The UNITOR system ranked first in the competition, suggesting that the combination of the adopted structured kernel and the Markovian modeling is beneficial.

Italian. *Questo lavoro descrive il sistema UNITOR che ha partecipato all'itaLIan Speech acT labEliNg task organizzato nell'ambito di EvalIta 2018. Il sistema è basato su una Structured Kernel-based Support Vector Machine (SVM) che rende la classificazione dei turni di dialogo dipendente dalle informazioni sintattiche e semantiche della frase, evitando la progettazione di alcuna feature specifica per il task. Una specifica formulazione Markoviana dell'algoritmo di apprendimento SVM permette inoltre di etichettare ciascun turno in funzione delle classificazioni dei turni precedenti. Il sistema UNITOR si é classificato al primo posto nella competizione, e questo conferma come la combinazione della funzione kernel e del modello Markoviano adottati sia molto utile allo sviluppo di sistemi di dialoghi robusti.*

1 Introduction

A dialogue agent is designed to interact and communicate with other agents, in a coherent manner, not just through one-shot messages, but according to sequences of meaningful and related messages on an underlying topic or in support to an overall goal (Traum, 1999). These communications are seen not just as transmitting information but as actions that change the state of the world, e.g., the mental states of the agents involved in the conversation, as well as the state, or context, of the dialogue. In other words, speech act theory allows to design an agent in order to place its communication within the same general framework as the agent's actions. In such a context, the robust recognition of the speech acts characterizing an interaction is crucial for the design and deployment of artificial dialogue agents.

This specific task has been considered in the first *itaLIan Speech acT labEliNg task at EvalIta* (iLISTEN, (Novielli and Basile, 2018)): given a dialogue between an agent and a user, the task consists in automatically annotating the user's dialogue turns with speech act labels, i.e. with the communicative intention of the speaker, such as *statement*, *request for information* or *agreement*. Table 1 reports the full set of speech act labels considered in the challenge, with definition and examples.

In this paper, the `UNITOR` system participating in the iLISTEN task within the EvalIta 2018 evaluation campaign is described. The system realize the classification task through a Structured Kernel-based Support Vector Machine (Vapnik, 1998) classifier. A structured kernel, namely a Smoothed Partial Tree Kernel (SPTK, (Croce et al., 2011)) is applied in order to make the classification of each utterance dependent from the syntactic and semantic information of each individual utterance.

Since turns are not observed in isolation, but immersed in a dialogue, we adopted a Markovian for-

Speech Act	Description	Example
OPENING	Dialogue opening or self-introduction	*Ciao, io sono Antonella*
CLOSING	Dialogue closing, e.g. farewell, wishes, intention to close the conversation	*Va bene, ci vediamo prossimamente*
INFO-REQUEST	Utterances that are pragmatically, semantically, and syntactically questions	*E cosa mi dici delle vitamine?*
SOLICIT-REQ-CLARIF	Request for clarification (please explain) or solicitation of system reaction	*Mmm, si ma in che senso?*
STATEMENT	Descriptive, narrative, personal statements	*Devo controllare maggiormente il mio peso.*
GENERIC-ANSWER	Generic answer	*Si, No, Non so*
AGREE	Expression of agreement, e.g. acceptance of a proposal, plan or opinion	*Si, so che importante.*
REJECT	Expression of disagreement, e.g. rejection of a proposal, plan, or opinion	*Ho sentito tesi contrastanti al proposito.*
KIND-ATT-SMALLTALK	Expression of kind attitude through politeness, e.g. thanking, apologizing or smalltalk	*Grazie,Sei per caso offesa per qualcosa che ho detto?*

Table 1: Full set of speech act labels considered at iLISTEN

mulation of SVM, namely SVM^{hmm} (Altun et al., 2003) so that the classification of the i^{th} utterance also depends from the dialogue act assigned at the $i-1^{th}$ utterance.

The UNITOR system ranked first in the competition, suggesting that the combination of the adopted structured kernel and the Markovian learning algorithm is beneficial.

In the rest of the paper, Section 2 describes the adopted machine learning method and the underlying semantic kernel functions. In Section 3, the performance measures of the system are reported while Section 4 derives the conclusions.

2 A Markovian Kernel-based Approach

The UNITOR system implements a Markovian formulation of the Support Vector Machine (SVM) learning algorithm. The SVM adopts a structured kernel function in order to estimate the syntactic and semantic similarity between utterances in a dialogue, without the need of any task-specific manual feature engineering (only the dependency parse of each sentence is required). In the rest of this section, first the learning algorithm is presented, then the adopted kernel method is discussed.

2.1 A Markovian Support Vector Machine

The aim of a Markovian formulation of SVM is to make the classification of a input example $x_i \in \mathbb{R}^n$ (belonging to a sequence of examples) dependent on the label assigned to the previous elements in a history of length m, i.e., $x_{i-m}, \ldots, x_{i-1}$.

In our classification task, a dialogue is a sequence of utterances $\mathbf{x} = (x_1, \ldots, x_s)$ each of them representing an example x_i, i.e., the specific i-th utterance. Given the corresponding sequence of expected labels $\mathbf{y} = (y_1, \ldots, y_s)$, a sequence of m step-specific labels (from a dictionary of d different dialogue acts) can be retrieved, in the form $y_{i-m}, \ldots, y_{i-1}$.

In order to make the classification of x_i dependent also from the previous decisions, we augment the feature vector of x_i introducing a projection function $\psi_m(x_i) \in \mathbb{R}^{md}$ that associates to each example a $md-$dimensional feature vector where each dimension set to 1 corresponds to the presence of one of the d possible labels observed in a history of length m, i.e. m steps before the target element x_i.

In order to apply a SVM, a projection function $\phi_m(\cdot)$ can be defined to consider both the observations x_i and the transitions $\psi_m(x_i)$ by concatenating the two representations as follows:

$$\phi_m(x_i) = x_i \,||\, \psi_m(x_i)$$

with $\phi_m(x_i) \in \mathbb{R}^{n+md}$. Notice that the symbol $||$ here denotes the vector concatenation, so that $\psi_m(x_i)$ does not interfere with the original feature space, where x_i lies.

Kernel-based methods can be applied in order to model meaningful representation spaces, encoding both the feature representing individual examples together with the information about the transitions. According to kernel-based learning (Shawe-Taylor and Cristianini, 2004), we can define a kernel function $K_m(x_i, z_j)$ between a generic item of a sequence x_i and another generic item z_j from the same or a different sequence, parametric in the history length m: it surrogates the product between

$\phi_m(\cdot)$ such that:

$$K_m(x_i, z_j) = \phi_m(x_i)\phi_m(z_j) = \\ = K^{obs}(x_i, z_j) + K^{tr}\big(\psi_m(x_i), \psi_m(z_j)\big)$$

In other words, we define a kernel that is the linear combination of two further kernels: K^{obs} operating over the individual examples x_i and a K^{tr} operating over the feature vectors encoding the involved transitions. It is worth noticing that K^{obs} does not depend on the position nor the context of individual examples in line with Markov assumption characterizing a large class of these generative models, e.g. HMM. For simplicity, we define K^{tr} as a linear kernel between input instances, i.e. a dot-product in the space generated by $\psi_m(\cdot)$:

$$K_m(x_i, z_j) = K^{obs}(x_i, x_j) + \psi_m(x_i)\psi_m(z_j)$$

At training time, we use the kernel-based SVM in a One-Vs-All schema over the feature space derived by $K_m(\cdot,\cdot)$. The learning process provides a family of classification functions $f(x_i; m) \subset \mathbb{R}^{n+md} \times \mathbb{R}^d$, which associate each x_i to a distribution of scores with respect to the different d labels, depending on the context size m. At classification time, all possible sequences $\mathbf{y} \in \mathcal{Y}^+$ should be considered in order to determine the best labeling $\hat{\mathbf{y}}$, where m is the size of the history used to enrich x_i, that is:

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{y}\in\mathcal{Y}^+}\{\sum_{i=1...m} f(x_i; m)\}$$

In order to reduce the computational cost, a *Viterbi-like decoding algorithm* is adopted[1]. The next section defines the kernel function K^{obs} applied to specific utterances.

2.2 Structured Kernel Methods for Speech Act Labeling

Several NLP tasks require the explorations of complex semantic and syntactic phenomena. For instance, in Paraphrase Detection, verifying whether two sentences are valid paraphrases involves the analysis of some rewriting rules in which the syntax plays a fundamental role. In Question Answering, the syntactic information is crucial, as largely demonstrated in (Croce et al., 2011).

[1]When applying $f(x_i; m)$ the classification scores are normalized through a softmax function and probability scores are derived.

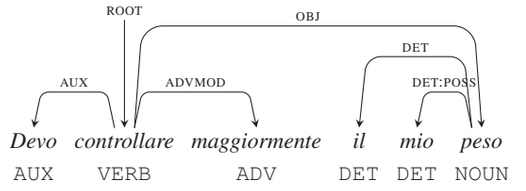


Figure 1: Dependency Parse Tree of *"Devo controllare maggiormente il mio peso"* (In English: *"I need to control my weight more"*)

A natural approach to exploit such linguistic information consists in applying kernel methods (Robert Müller et al., 2001; Shawe-Taylor and Cristianini, 2004) on structured representations of data objects, e.g., documents. A sentence s can be represented as a parse tree[2] that expresses the grammatical relations implied by s. Tree kernels (TKs) (Collins and Duffy, 2001) can be employed to directly operate on such parse trees, evaluating the tree fragments shared by the input trees. This operation corresponds to a dot product in the implicit feature space of all possible tree fragments.

Whenever the dot product is available in the implicit feature space, kernel-based learning algorithms, such as SVMs, can operate in order to automatically generate robust prediction models. TKs thus allow to estimate the similarity among texts, directly from sentence syntactic structures, that can be represented by parse trees. The underlying idea is that the similarity between two trees T_1 and T_2 can be derived from the number of shared tree fragments. Let the set $\mathcal{T} = \{t_1, t_2, \ldots, t_{|\mathcal{T}|}\}$ be the space of all the possible substructures and $\chi_i(n_2)$ be an indicator function that is equal to 1 if the target t_i is rooted at the node n_2 and 0 otherwise. A tree-kernel function over T_1 and T_2 is defined as follows: $TK(T_1, T_2) = \sum_{n_1\in N_{T_1}} \sum_{n_2\in N_{T_2}} \Delta(n_1, n_2)$ where N_{T_1} and N_{T_2} are the sets of nodes of T_1 and T_2 respectively, and $\Delta(n_1, n_2) = \sum_{k=1}^{|\mathcal{T}|} \chi_k(n_1)\chi_k(n_2)$ which computes the number of common fragments between trees rooted at nodes n_1 and n_2. The feature space generated by the structural kernels obviously depends on the input structures. Notice that different tree representations embody different linguistic theories and may produce more or less effective syntactic/semantic feature spaces for a

[2]Parse trees can be extracted using automatic parsers. In our experiments, we used SpaCy `https://spacy.io/`.

given task.

Dependency grammars produce a significantly different representation which is exemplified in Figure 1. Since tree kernels are not tailored to model the labeled edges that are typical of dependency graphs, these latter are rewritten into explicit hierarchical representations. Different rewriting strategies are possible, as discussed in (Croce et al., 2011): a representation that is shown to be effective in several tasks is the Grammatical Relation Centered Tree (GRCT) illustrated in Figure 2: the PoS-Tags are children of grammatical function nodes and direct ancestors of their associated lexical items.

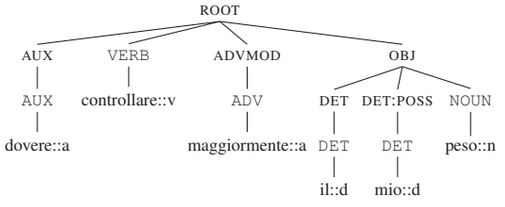


Figure 2: Grammatical Relation Centered Tree (GRCT) of *"Devo controllare maggiorment il mio peso"*.

Different tree kernels can be defined according to the types of tree fragments considered in the evaluation of the matching structures. In the *Subtree Kernel* (Vishwanathan and Smola, 2002), valid fragments are only the grammatically well formed and complete subtrees: every node in a subtree corresponds to a context free rule whose left hand side is the node label and the right hand side is completely described by the node descendants. Subset trees are exploited by the *Subset Tree Kernel* (Collins and Duffy, 2001), which is usually referred to as Syntactic Tree Kernel (STK); they are more general structures since their leaves can be non-terminal symbols. The subset trees satisfy the constraint that grammatical rules cannot be broken and every tree exhaustively represents a CFG rule. *Partial Tree Kernel* (PTK) (Moschitti, 2006) relaxes this constraint considering partial trees, i.e., fragments generated by the application of partial production rules (e.g. sequences of non terminal with gaps). The strict constraint imposed by the STK may be problematic especially when the training dataset is small and only few syntactic tree configurations can be observed. The Partial Tree Kernel (PTK) overcomes this limitation, and usually leads to higher accuracy, as shown in (Moschitti, 2006).

Capitalizing lexical information in Convolution Kernels. The tree kernels introduced above perform a hard match between nodes when comparing two substructures. In NLP tasks, when nodes are words, this strict requirement reflects in a too strict lexical constraint, that poorly reflects semantic phenomena, such as the synonymy of different words or the polysemy of a lexical entry. To overcome this limitation, we adopt Distributional models of Lexical Semantics (Sahlgren, 2006; Mikolov et al., 2013) to generalize the meaning of individual words by replacing them with geometrical representations (also called Word Embeddings) that are automatically derived from the analysis of large-scale corpora. These representations are based on the idea that words occurring in the same contexts tend to have similar meaning: the adopted distributional models generate vectors that are similar when the associated words exhibit a similar usage in large-scale document collections. Under this perspective, the distance between vectors reflects semantic relations between the represented words, such as paradigmatic relations, e.g., quasi-synonymy. These word spaces allow to define meaningful soft matching between lexical nodes, in terms of the distance between their representative vectors. As a result, it is possible to obtain more informative kernel functions which are able to capture syntactic and semantic phenomena through grammatical and lexical constraints.

The *Smoothed Partial Tree Kernel* (SPTK) described in (Croce et al., 2011) exploits this idea extending the PTK formulation with a similarity function σ between nodes:

$$\Delta_{SPTK}(n_1, n_2) = \mu\lambda\sigma(n_1, n_2)\ , \text{ if } n_1 \text{ and } n_2 \text{ are leaves}$$
$$\Delta_{SPTK}(n_1, n_2) = \mu\sigma(n_1, n_2)\Big(\lambda^2 + \sum_{\vec{I}_1, \vec{I}_2 : l(\vec{I}_1) = l(\vec{I}_2)} \lambda^{d(\vec{I}_1)+d(\vec{I}_2)} \prod_{k=1}^{l(\vec{I}_1)} \Delta_{SPTK}\left(c_{n_1}(i_k^1), c_{n_2}(i_k^2)\right)\Big) \quad (1)$$

In the SPTK formulation, the similarity function $\sigma(n_1, n_2)$ between two nodes n_1 and n_2 can be defined as follows:

- if n_1 and n_2 are both lexical nodes, then $\sigma(n_1, n_2) = \sigma_{LEX}(n_1, n_2) = \tau \frac{\vec{v}_{n_1} \cdot \vec{v}_{n_2}}{\|\vec{v}_{n_1}\| \|\vec{v}_{n_2}\|}$. It is the cosine similarity between the word vectors $\vec{v}_{n_1}$ and $\vec{v}_{n_2}$ associated with the labels of n_1 and n_2, respectively. τ is called *terminal factor* and weighs the contribution

of the lexical similarity to the overall kernel computation.

- else if n_1 and n_2 are nodes sharing the same label, then $\sigma(n_1, n_2) = 1$.
- else $\sigma(n_1, n_2) = 0$.

In the challenge we adopt the SPTK in order to implement the K^{obs} function used in the Markovian SVM. This kernel in fact has been showed very robust in the classification of (possibly short) sentences, such as in Question Classification (Croce et al., 2011) or Semantic Role Labeling (Croce et al., 2011; Croce et al., 2012).

Dataset	#dialogues	#user turns	#system turns	#total turns
train	40	1,097	1,119	2,216
test	20	479	492	971
complete	60	1,576	1,611	3,187

Table 2: Statistics about the iLISTEN dataset

3 Experimental Results

In iLISTEN, the reference dataset includes the transcriptions of 60 dialogues amounting to about $22,000$ words. The detailed statistics regarding dialogues and turns in the train and test dataset are reported in Table 2.

	Micro			Macro		
Run	**P**	**R**	**F1**	**P**	**R**	**F1**
UNITOR	.733	.733	**.733**	.681	.628	**.653**
System2	.685	.685	.685	.608	.584	.596
Baseline	.340	.340	.340	.037	.111	.056

Table 3: Results of the UNITOR system

In the proposed classification workflow each utterance from the training/test material is processed through the SpaCy[3] dependency parser whose outputs are automatically converted into GRCT structures[4] discussed in Section 2. These structures are used within the Markovian SVM implemented in KeLP[5] (Filice et al., 2015; Filice et al., 2018).

The learning algorithm is based on a SPTK combined with a One-Vs-All multi-classification schema adopted to assign individual utterances to the targeted classes. All the parameters of the specific kernel (i.e. the contribution of the lexical nodes in the overall computation) and of the SVM algorithm have been tuned via 10-cross fold validation over the training set. In the Markovian SVM, a history of $m = 1$ previous steps allowed to achieve the best results during the parameterization step. Given the limited size of the dataset, higher values of m led to sparse representation of the transitions ψ_m that are not helpful. As a multi-classification task, results are reported in terms of precision (P), recall (R) and F1-score with respect to the gold standard, as reported in Table 3. These are averaged across each utterance (*micro*-statistics) and per class (*macro*-statistics).

Among the two submitted systems, UNITOR (reported on top of the table) achieved best results, both considering micro and macro statistics, where a F1 of 0.733 and 0.653 are achieved, respectively. These results are higher with respect to the other participant (namely System2) and far higher than the baseline (that confirms the difficulty of the task).

Given the unbalanced number of examples for each class, UNITOR achieves higher results w.r.t. the micro statistics, while lower results are achieved w.r.t. classes with a reduced number of examples. The confusion matrix reported in Table 4 shows that some recurrent misclassifications (e.g. the `STATEMENT` class with respect to the `REJECT` class) need to be carefully addressed. Clearly this is a very challenging task, also for the annotators, where the differences between the speech act is not strictly defined: as for example, given the stimulus of the system "*Bisognerebbe mangiare solo se si ha fame, ed aspettare che la digestione sia completata prima di assumere altri cibi*[6]" the answer "*a volte il lavoro non mi permette di mangiare con ritmi regolari!*[7]" should be labeled as `REJECT` while the system provides `STATEMENT`. Overall this results is straightforward, also considering that the system did not required any task specific feature modeling, but the adopted structured kernel based method allows capitalizing the syntactic and semantic information useful for the task. The only requirement of the system is the availability of a dependency parser.

[3] It is freely available for several languages (including Italian) at `https://spacy.io/`

[4] Utterances may include more than one sentence and potentially generate different trees. These cases are handled as follows: all trees after the first one are linked through the creation of an artificial link between their roots and the root of the tree generated by the first sentence.

[5] `http://www.kelp-ml.org/?page_id=215`

[6] Translated in English: "*You should only eat if you are hungry, and wait until digestion is complete before eating again.*"

[7] Translated in English: "*sometimes work doesn't allow me to eat at a regular pace!*"

	STATEMENT	KIND-ATT.	GEN.-ANSW.	REJECT	CLOSING	SOL.-CLAR.	OPENING	AGREE	INFO-REQ.
STATEMENT	**153**	6	3	24	0	3	0	2	13
KIND-ATT.	4	**17**	0	5	1	2	0	3	2
GEN.-ANSW.	1	0	**48**	0	0	1	0	6	0
REJECT	0	3	0	**3**	0	0	0	0	1
CLOSING	0	0	0	0	**7**	1	0	1	0
SOL.-CLAR.	0	6	0	2	1	**8**	0	1	2
OPENING	0	0	0	0	0	0	**11**	0	0
AGREE	0	3	1	1	0	0	0	**11**	1
INFO-REQ.	4	9	0	4	1	9	0	0	**93**

Table 4: Confusion Matrix of the UNITOR system w.r.t. gold standard. In column the number of classes from the gold standard, while rows report the system decisions. In bold correct classifications.

4 Conclusions

In this paper the description of the UNITOR system participating to the iLISTEN task at EvalIta 2018 has been provided. The system ranked first in all the evaluations. Thus, the proposed classification strategy shows the beneficial impact of the combination of a structured kernel-based method with a Markovian classifier, capitalizing the contribution of the dialogue modeling in deciding the speech act of individual sentences. One important finding of this evaluation is that a quite robust speech classifier can be obtained with almost no requirement in term of task-specific feature and system engineering: results are appealing mostly considering the reduced size of the dataset. Further work is needed to improve the overall F1 scores, possibly extending the adopted kernel function by addressing other dimensions of the linguistic information or also making the kernel more sensitive to task-specific knowledge. Also the combination of the adopted strategy with recurrent neural approaches is an interesting research direction.

References

Y. Altun, I. Tsochantaridis, and T. Hofmann. 2003. Hidden Markov support vector machines. In *Proceedings of the International Conference on Machine Learning*.

Michael Collins and Nigel Duffy. 2001. Convolution kernels for natural language. In *Proceedings of Neural Information Processing Systems (NIPS'2001)*, pages 625–632.

Danilo Croce, Alessandro Moschitti, and Roberto Basili. 2011. Structured lexical similarity via convolution kernels on dependency trees. In *Proceedings of EMNLP*.

Danilo Croce, Alessandro Moschitti, Roberto Basili, and Martha Palmer. 2012. Verb classification using distributional similarity in syntactic and semantic structures. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 263–272.

Simone Filice, Giuseppe Castellucci, Danilo Croce, and Roberto Basili. 2015. Kelp: a kernel-based learning platform for natural language processing. In *Proceedings of ACL-IJCNLP 2015 System Demonstrations*, pages 19–24.

Simone Filice, Giuseppe Castellucci, Giovanni Da San Martino, Alessandro Moschitti, Danilo Croce, and Roberto Basili. 2018. Kelp: a kernel-based learning platform. *Journal of Machine Learning Research*, 18(191):1–5.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Alessandro Moschitti. 2006. Efficient convolution kernels for dependency and constituent syntactic trees. In *ECML*, Berlin, Germany.

Nicole Novielli and Pierpaolo Basile. 2018. Overview of the evalita 2018 italian speech act labeling (ilisten) task. In Tommaso Caselli, Nicole Novielli, Viviana Patti, and Paolo Rosso, editors, *Proceedings of the 6th evaluation campaign of Natural Language Processing and Speech tools for Italian (EVALITA'18)*, Turin, Italy. CEUR.org.

Klaus Robert Müller, Sebastian Mika, Gunnar Rätsch, Koji Tsuda, and Bernhard Schölkopf. 2001. An introduction to kernel-based learning algorithms. *IEEE Transactions on Neural Networks*, 12(2):181–201.

Magnus Sahlgren. 2006. *The Word-Space Model*. Ph.D. thesis, Stockholm University.

John Shawe-Taylor and Nello Cristianini. 2004. *Kernel Methods for Pattern Analysis*. Cambridge University Press.

David R. Traum, 1999. *Speech Acts for Dialogue Agents*, pages 169–201. Springer Netherlands, Dordrecht.

Vladimir N. Vapnik. 1998. *Statistical Learning Theory*. Wiley-Interscience.

S.V.N. Vishwanathan and Alexander J. Smola. 2002. Fast kernels on strings and trees. In *Proceedings of Neural Information Processing Systems*, pages 569–576.

Misogyny Detection and Classification in English Tweets: The Experience of the ITT Team

Elena Shushkevich
Social Media Research Group
Institute of Technology Tallaght
Dublin, Ireland
e.shushkevich@yandex.ru

John Cardiff
Social Media Research Group
Institute of Technology Tallaght
Dublin, Ireland
john.cardiff@it-tallaght.ie

Abstract

English. The problem of online misogyny and women-based offending has become increasingly widespread, and the automatic detection of such messages is an urgent priority. In this paper, we present an approach based on an ensemble of Logistic Regression, Support Vector Machines, and Naïve Bayes models for the detection of misogyny in texts extracted from the Twitter platform. Our method has been presented in the framework of the participation in the Automatic Misogyny Identification (AMI) Shared Task in the EVALITA 2018 evaluation campaign.

Italiano. *Il problema della misoginia online e dell'odio diretto verso le donne si sta diffondendo sempre più, e così il riconoscimento automatico di tali messaggi è una priorità importante.*
In questo articolo, presentiamo un approccio basato sui classificatori Logistic Regression, SVM e Naive Bayes per il riconoscimento automatico della misoginia in testi estratti da Twitter.
Il nostro metodo è stato presentato attraverso la nostra partecipazione allo shared task AMI presso la campagna di valutazione EVALITA 2018.

1 Introduction

It is hard to miss the fact that an intensive growth of social networking has led not only to the rise of personal communication opportunities, but also to an increase in aggression on social media. Hate speech can be aimed at sexual orientation, race, religion as gender as a whole. In particular, when the target of hate speech is women, we could say that this is misogyny. Nowadays, more and more attention is paid to this problem, and one of the directions for the hate speech recognition is the women-oriented aggression detection in social networks.

It is important to work with hate speech and misogyny detection now, because over the course of time the data from social networks will grow and this problem will become more and more serious. It is necessary to create a range of systems which allow us to detect and control the number of hate speech messages, and we need to understand how to classify this type of information and how we

could reduce the number of it. So, it is a big challenge to find the way of misogyny data detection and processing.

This paper describes our participation in the Automatic Misogyny Identification (AMI) Shared Task, in EVALITA 2018 (Fersini, Nozza and Rosso, 2018). The aim of the task is to identify misogynistic text in tweets. The task contained two different subtasks:
Subtask A - Misogyny Identification: the main goal of the task was to separate misogynous tweets from non-misogynous.
Subtask B - Misogynistic Behavior and Target Classification: the idea of the target classification was to define misogynous tweet which offends a specific person (Active) and tweets which insult a group of people (Passive).
Misogynistic behavior task was intended to divide misogynous tweets into different groups:
- Stereotype & Objectification: a widely held but fixed and oversimplified image or idea of a woman, description of women's physical and/or comparisons to narrow standards.
- Dominance: to assert the superiority of men over women or to highlight gender inequality.
- Derailing: to justify abuse of women, rejecting male responsibility and an attempt to disrupt the conversation in order to redirect women's conversations on something more comfortable for men.
- Sexual Harassment & Threats of Violence: to describe actions as sexual advances, requests for sexual favours, harassment of a sexual nature, intent to physically assert power over women through threats of violence.
- Discredit: slurring of women with no other larger intention.

There were two datasets for the task, one of which contained tweets in the English language and another containing Italian tweets. Our team worked with English dataset only. The English dataset was composed of 4,000 tweets for training and 1,000 tweets for testing. The results were evaluated using the accuracy performance for Task A and macro F-measure performance for Task B.

This paper presents our approach to solve the above problems. The main thrust of our approach is to build a model that allows us to assess the classification of any tweet to its assigned group.

The paper is organized as follows. Some relevant related works in the area are described in Section 2. Section 3 presents the way we conducted data preprocessing and the approach we chose for building the desired model. In Section 4 the results are described and analyzed. In Section 5 we summarize our work.

2 Related work

There are a number of approaches in the area of text processing by machine learning methods which allow us to deal with misogyny and harassment in texts. Some of these were presented in the AMI@IBEREVAL-2018 shared task (Fersini, Anzovino and Rosso, 2018). The aim of this challenge was to detect misogynistic tweets and to create the model which was able to classify misogynistic tweets for different groups depending on the type of misogyny. In particular, it was demonstrated that, using models based on Support Vector Machines (Pamungkas et al., 2018) and ensembles of models (Frenda et al., 2018), it is possible and quite successful in cases where the aim is to make a classification of tweets for different types and functions of misogyny. In our work we apply several of the same techniques - Support Vectors Machines and ensembles of models - to the task of misogyny tweets detection.

Some works which could help us to understand the way to hate speech messages classification were published in recent years. In (Schmidt and Wiegand, 2017) the authors demonstrated methodologies of hate speech data processing. In another work (Waseem and Hovy, 2016) there were presented useful approaches to detect racial and sexist offenses. It should be noted that there was a classification for 3 different groups (hate speech, derogatory, profanity) with the understanding that hate speech is a kind of abusive language.

In the research reported in (Nobata et al., 2016), it was shown (Bartlett et al., 2014) how to use NLP to analyse English-language misogynistic tweets to find the frequencies of abusive words and the users who used this type of words more often. In other works (Alexandrov et al., 2013; Kaurova et al., 2010) the authors focused on creating models which could allow the evaluation of the tone of the text on a scale from very negative to very positive. They constructed a model for the groups of 3, 5 and 8 different categories and were able to achieve the results with a high accuracy using additional tools like GMDH Shell and Semantic Orientation Calculation (So-CAL), which demonstrates the very high potential of using inductive modelling for text-mining tasks. We are planning to use techniques which were mentioned above to improve the results of our model in future.

3 System

In our approach we perform a number of sequential actions including preprocessing, model design, and finally embedding the constructed models in one ensemble.

3.1 Preprocessing

In the first step, we prepared the data for the classification. To clean the data we removed the string punctuation and converted words to lower case. For the vectorization we used the tf-idf (term frequency–inverse document frequency) method which allows us to reduce the weight of frequently occurring in many documents words and to increase the weight of frequently occurring words in the documents. These were carried out for the first run. For the subsequent two runs, we added some extra preprocessing steps:

- the replacement of all links with the string "URL"
- the replacement of all references to Twitter users (i.e, terms starting with the "@" symbol) with the term "USER".
- we marked some combinations of symbols which were used often in messages such as "!!! ", "??? " and other emotional expressions, and replaced them with the term "emoji".

3.2 Models

The main idea of the modeling was to create an ensemble of different models which could complement each other to achieve the best results. The final blended model assigns the tweet to a specific class by majority voting. We used a number of simple models which include:

- Logistic regression model. Logistic regression involves the construction of a discriminant model, which calculates the probability from a function of a weighted set of observation features and assigns a class to each observation. The classifier based on logistic regression applies an exponential function to a linear combination of objects obtained from the input data (Wang et al., 2012; Wright, 1995).

- Support Vector Machines classifier. As it was shown in (Joachims et al., 2002), this method is very useful in work with texts. The idea of this method is to translate the source vectors into a higher dimension space and search for such a separating hyperplane so that the gap in this space is maximal. There are two parallel hyperplanes on both sides of the hyperplane that are constructed to separate the classes, and one hyperplane that will maximize the distance to two parallel ones is sought.

- Naive Bayes classifier. One of the advantages of this method is the high speed of calculations (Zhang and Di Li, 2007), and another one is the number of the data which is needed to train the model - in this case it is not necessary to have a big training dataset to achieve a high level of classification parameter estimation.

In the next step we combined the Naive Bayes approach and Logistic regression approach in one model, as presented in the work

(Genkin et al., 2007),which produced quite good results.

In the final step we combined the models we have mentioned, Logistic regression (LR), Support Vector Machines (SVM), Naive Bayes and Logistic Regression (NB+LR), into one ensemble. In this blended model the probabilities of belonging to different classes from the simple models were summed and averaged. We marked as a final choice the class which had the highest average probability.

4 Results

We chose three different runs for the evaluation: one of them was implemented by using the simplest type of preprocessing (we just deleted punctuation symbols and changed all letters to the low case) and this variant supposed that we marked a tweet as misogynistic one in case that two of three types of classification marked this tweet as misogynous (Misogyny+Target or Misogyny+Misogynistic Behavior or Target+Misogynistic Behavior).

In the next step, we carried out a more intricate preprocessing as described in Section 3.1 and applied the type of tweets labeling such a way as we detected a tweet as misogynistic each time when at least one classifier worked.

The last run was implemented by using the most complicated preprocessing and the type of tweets labeling such as at the first run.

Table 1 shows the results of all three classification types. As can be seen, the fourth type of selection was the most successful. It could be concluded that the blended model which contained more simple models (Logistic Regression, Naive Bayes + Logistic Regression and Support Vector Machines) allows us to achieve the best results for all classification types: Misogyny Identification, Target Classification and Misogynistic Behavior classification.

It should be noted that we used the F-Measure for the results' evaluation because this assessment allows bringing together both recall and precision and because of the imbalance within both the Misogynistic Category Classification and the Target Classification.

Task	Classifier	F1-score
Misogyny Identification	LR	0.78
	NB+LR	0.72
	SVM	0.71
	Blend	**0.78**
Target Classification	LR	0.60
	NB+LR	0.66
	SVM	0.76
	Blend	**0.76**
Misogynistic Behavior	LR	0.50
	NB+LR	0.52
	SVM	0.57
	Blend	**0.64**

Table 1.Performance on the validation set.

Also note that the results of our model increase when the number of different classes decreases, thus an efficiency of the blended model is reduced from the Misogyny Identification classification results to the Misogynistic Behavior classification ones.

The results of all 3 runs for the blended model with the testing dataset are presented in Table 2.

Subtask A - English		
Rank	Team	Accuracy
8	ITT.c.run2.tsv	0.638
9	ITT.c.run3.tsv	0.636
10	ITT.c.run1.tsv	0.636

Table 2. Results of the classification.

It can be concluded by the results on the test data, the best run is the one with the most complicated preprocessing and the type of labelling, when we mark tweet as misogynistic every time when at least one of classifiers worked.

5 Conclusion

A negative aspect of the increased usage of platforms like Twitter is that incidents of aggression and related activities like harassment and misogyny have increased significantly. Nowadays it is an urgent problem to deal with such type of text information and messages, and there are a lot of challenges that have a connection with this task. In this article we have described our approach to misogyny detection and classification of tweets. The method was presented for evaluation in the framework of the Automatic Misogyny Identification (AMI) Shared Task at EVALITA 2018. We built an ensemble of models that includes Logistic regression, Naive Bayes and Support Vector Machines approaches, which classified the data taking into account the probabilities of belonging to classes calculated by simpler models. It was shown that it is possible to achieve quite good results using the final blended model and our model showed the best results for the binary classification of misogynistic tweets and non-misogynistic ones.

We observed preprocessing to be a very important part of the data handling and it has a high impact on the results of all models. From our results it could be concluded that the highest accuracy has been produced with maximum additional work at the preprocessing stage. It was important to pay attention to the replacement of links and references with special symbols, because the run with this type of alteration demonstrated the best results. Also, the best type of labelling misogynistic tweets was to mark the message as misogyny if any one of the type of classification worked. At first we had an idea that it could be more reliably if we mark tweet when 2 of 3 classifications mark it, but the real results disproved that hypothesis. We are currently investigating the addition of more features and models for the blended model to improve our results in the future.

References

Alexandrov M., Danilova V., Koshulko A., Tejada, J. 2013. *Models for opinion classication of blogs taken from Peruvian Facebook.* Proceedings of 4th International Conference on Inductive Modeling (ICIM-2013), pp. 241–246 .

Bartlett J., Norrie R., Patel S., Rumpel R., Wibberley S. 2014. *Misogyny on twitter*, http://www.demos.co.uk/, 05.

Fersini, E., Anzovino, M., Rosso. P. 2018. *Overview of the Task on Automatic Misogyny Identification at IberEval*. Proceedings of the Third Workshop on Evaluation of Human Language Technologies for Iberian Languages (IberEval 2018), co-located with 34th Conference of the Spanish Society for Natural Language Processing (SEPLN 2018). CEUR Workshop Proceedings. CEUR-WS.org

Fersini E., Nozza D., Rosso P. 2018. *Overview of the Evalita 2018 Task on Automatic Misogyny Identification (AMI).* Proceedings of the 6th evaluation campaign of Natural Language Processing and Speech tools for Italian (EVALITA'18). Caselli, Tommaso and Novielli, Nicole and Patti, Viviana and Rosso, Paolo CEUR.org, Turin, Italy

Frenda S., Ghanem B. 2018. *Montes-y-Gómez M. Exploration of Misogyny in Spanish and English tweets.* CEUR Workshop Proceedings. CEUR-WS.org.

Genkin A., Lewis D., Madigan D. 2007. *Large-scale bayesian logistic regression for text categorization*. Technometrics, 49(3):291–304.

Joachims, T. 2002. *Learning to classify text using support vector machines: Methods, theory and algorithms.* Kluwer Academic Publishers.

Kaurova O., Alexandrov M., Ponomareva N. 2010. *The Study of Sentiment Word Granularity for Opinion Analysis (a Comparison with Maite Taboada Works*). International Journal on Social Media. MMM: Monitoring, Measurement, and Mining 1(1), 45–57.

Nobata C., Tetreault J., Thomas A., Mehdad Y., Chang Y. 2016. *Abusive language detection in online user content.* Proceedings of the 25th International Conference on World Wide Web, pp. 145–153. International World Wide Web Conferences Steering Committee.

Pamungkas E.W., Cignarella A.T., Basile V., Patti V. 2018. *14-ExLab@UniTo for AMI at IberEval2018: Exploiting Lexical Knowledge for Detecting Misogyny in English and Spanish Tweets.* CEUR Workshop Proceedings. CEUR-WS.org.

Schmidt, A., Wiegand, M. 2017. *A survey on hate speech detection using natural language processing.* Proceedings of the Fifth International

Workshop on Natural Language Processing for Social Media. Association for Computational Linguistics, Valencia, Spain, pp. 1–10.

Shushkevich E., Cardiff J. 2018. *Classifying Misogynistic Tweets Using a Blended Model: The AMI Shared Task in IBEREVAL 2018*. CEUR Workshop Proceedings. CEUR-WS.org.

Wang S., Manning C.D. 2012. *Baselines and bigrams: simple, good sentiment and topic classification.* Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers, ACL 2012, vol. 2, pp. 90–94.

Waseem, Z., Hovy, D. 2016. *Hateful symbols or hateful people? predictive features for hate speech detection on Twitter.* SRW@ HLT-NAACL, pp. 88–93.

Wright R. 1995. *Logistic regression*. L.C. Grimm & P.R. Yarnold (Eds.) Reading and understanding multivariate statistics. Washington, DC: American Psychological Association, 217-244

Zhang H. and Di Li. 2007. *Naıve bayes text classifier.* Granular Computing. GRC 2007. IEEE International Conference on, pages 708–708. IEEE.

Automatic Expansion of Lexicons for Multilingual Misogyny Detection

Simona Frenda
Università degli Studi di Torino, Italy
Universitat Politècnica de València, Spain
simona.frenda@unito.it

Bilal Ghanem
Universitat Politècnica de València, Spain
bigha@doctor.upv.es

Estefanía Guzmán-Falcón, Manuel Montes-y-Gómez and Luis Villaseñor-Pineda
Instituto Nacional de Astrofísica
Óptica y Electrónica (INAOE), Mexico.
{fany.guzman, mmontesg, villasen}@inaoep.mx

Abstract

English. The automatic misogyny identification (AMI) task proposed at IberEval and EVALITA 2018 is an example of the active involvement of scientific Research to face up the online spread of hate contents against women. Considering the encouraging results obtained for Spanish and English in the precedent edition of AMI, in the EVALITA framework we tested the robustness of a similar approach based on topic and stylistic information on a new collection of Italian and English tweets. Moreover, to deal with the dynamism of the language on social platforms, we also propose an approach based on automatically-enriched lexica. Despite resources like the lexica prove to be useful for a specific domain like misogyny, the analysis of the results reveals the limitations of the proposed approaches.

Italiano. *Il task AMI circa l'identificatione automatica della misoginia proposto a IberEval e a EVALITA 2018 è un chiaro esempio dell'attivo coinvolgimento della Ricerca per fronteggiare la diffusione online di contenuti di odio contro le donne. Considerando i promettenti risultati ottenuti per spagnolo e inglese nella precedente edizione di AMI, nel contesto di EVALITA abbiamo testato la robustezza di un approccio simile, basato su informationi stilistiche e di dominio, su una nuova collezione di tweet in inglese e in italiano. Tenendo conto dei repentini cambiamenti del linguaggio nei social network, proponiamo anche un approccio basato su lessici automaticamente estesi. Nonostante risorse come i lessici risultano utili per domini specifici come quello della misoginia, analizzando i risultati emergono i limiti degli approcci proposti.*

1 Introduction

The anonymity and the interactivity, typical of computer-mediated communication, facilitate the spread of hate messages and the perpetuated presence of hate contents online. As investigated by Fox et al. (2015), these factors increase and influence social misbehaviors also offline. In order to foster scientific research to find optimal solutions that could help to monitor the spread of hate speech contents, different tasks have been proposed in various campaigns of evaluation. An example is the AMI shared task proposed at IberEval 2018[1] and later at EVALITA 2018[2]. This task focuses on the automatic identification of misogyny in different languages. In particular, the first edition focuses on Spanish and English languages, and the second one on a new English corpus and Italian language. The multilingual context allows to observe the analogies and differences between different languages. The AMI's organizers (Fersini et al., 2018a; Fersini et al., 2018b) asked participants to detect firstly misogynistic tweets and then classify the misogynistic categories and the kind of target (individuals or groups). In the first edition, we proposed an approach based on stylistic and topic information captured respectively by means of character n-grams and a set of modeled lexica (Frenda et al., 2018). Considering the encouraging results obtained with the lexicon-based approach in Spanish and English languages, we re-proposed a similar approach for Italian language and a new collection of English tweets in

[1] http://amiibereval2018.wordpress.com/
[2] http://amievalita2018.wordpress.com/

order to test the performance and robustness of this approach. Actually, in this paper we propose two approaches. The first one, similar to previous work (Frenda et al., 2018), involves topic, linguistic and stylistic information. The second one focuses mainly on the automatic extension of the original lexica. Indeed, to deal with the continuous variation of the language on social platforms, the modeled lexica are enriched considering the contextual similarity of lexica by the use of pre-trained word embeddings. This technique helps the system to consider also new terms relative to the topic information of the original lexica. It could be considered as a good methodology to upgrade automatically the existing list of words used to block offensive contents in real applications of Internet companies. Indeed, a comparison between the two approaches reveals that the automatic enrichment of the lexica improves the results especially for English language. However, comparing the results obtained in both competitions and observing the error analyses, we notice that lexica represent a good resource for a specific domain like misogyny, but they are not sufficient to detect misogyny online.

Following, Section 2 describes the studies that inspired our work. Section 3 explains the approaches employed in both languages. Section 4 discusses the obtained results and delineates some conclusions.

2 Related Work

A first work about misogyny detection is proposed in Anzovino et al. (2018). In this study, the authors compared the performance of different supervised approaches using word embeddings, stylistic and syntactic features. In particular, their results reveal that the best machine learning approach for identification of misogyny is the linear Support Vector Machine (SVM) classifier. In general machine learning techniques are the most used in hate speech detection (Escalante et al., 2017; Nobata et al., 2016), because they allow researchers for exploring closely the issue exploiting different features, such as textual (Chen et al., 2012) and syntactical aspects (Burnap and Williams, 2014) or semantic and sentiment information (Samghabadi et al., 2017; Nobata et al., 2016; Gitari et al., 2015). Finally, some recent works have investigated also the potential of deep learning techniques (Mehdad and Tetreault, 2016; Del Vigna et al., 2017). Considering the specific domain concerning the hate against women, this work exploits stylistic, linguistic and topic information about the misogynistic speech. In particular, differently from previous studies, we use specific lexica relative to offensiveness and discredit of women for English and Italian languages, and we extend them with new words relative to the issues of the considered lexica. Considering the fact that commercial methods rely currently on the use of blacklists to monitor or block offensive contents, the proposed approach could help to upgrade their blacklists automatizing the process of the lexicon building.

3 Proposed Approaches

The AMI shared task proposed at EVALITA 2018 aims to detect misogyny in English and Italian collections of tweets. The organizers asked participants to detect misogynistic texts (Task A), and then, if the tweet is predicted as misogynistic, to distinguish the nature of target (individuals or groups labeled respectively “active” and “passive”), and identify the type of misogyny (Task B), according to the following classes proposed by Poland (2016): (a) stereotype and objectification, (b) dominance, (c) derailing, (d) sexual harassment and threats of violence, and (e) discredit. Actually, these classes represent the different manifestations and the various aspects of this social misbehavior. Table 1 shows the composition of the datasets.

Considering the promising results obtained at the IberEval campaign, in this work we use two approaches mainly based on lexica. The first one (Section 3.1) is similar to the approach used in Frenda et al. (2018), based on topic, linguistic and stylistic information captured by means of modeled lexica and n-grams of characters and words. The second one (Section 3.2) principally involves the automatically extended versions of the original lexica (Guzmán Falcón, 2018). In particular, we aim: 1) to test the robustness of lexicon based approaches in the new collections of tweets and in a new language, and 2) to understand the impact of automatically enriched lexica to face up the variation of the language in the multilingual computer-mediated communication.

	Misogynistic							Non-misogynistic
	(a)	(b)	(c)	(d)	(e)	active	passive	
Italian								
Training set	668	71	24	431	634	1721	97	2172
Test set	175	61	2	170	104	446	66	488
English								
Training set	179	148	92	352	1014	1058	727	2215
Test set	140	124	11	44	141	401	59	540

Table 1: Composition of AMI's datasets at EVALITA 2018.

3.1 Approach 1: using manually-modeled lexica (MML)

The first proposed approach aims to capture topic, linguistic and stylistic information by means of manually-modeled lexica and n-grams of words and characters. Below the features description for each language.

English Features. For the detection of misogyny in English tweets, we employed the manually-modeled lexica proposed in Frenda et al. (2018). These lexica concerns sexuality, profanity, femininity and human body as described in Table 2.

These lexica contain also slang expressions. Moreover, we take into account hashtags and abbreviations collected in Frenda et al. (2018): 40 misogynistic hashtags, such as: $\#iha \quad e$ or $\#bit \quad e \quad ink$; and a list of 50 negative abbreviations, such as wt or $s \quad$. Considering the most relevant n-grams of words, we employ the bigrams for the first task and the combination of unigrams, bigrams and trigrams (hence defined as UBT) for the second task. Moreover, the bag of characters (BoC) in a range from 1 to 7 grams is employed to manage misspellings and to capture stylistic aspects of digital writing. In order to perform the experiments, each tweet is represented as a vector. The presence of words in each lexicon is pondered with Information Gain, and character and word n-grams are weighted with Term Frequency-Inverse Document Frequency (TF-IDF) measure. In addition, considering the fact that in Frenda et al. (2018) several misclassified misogynistic tweets were ironic or sarcastic, we try to analyze the impact of irony in misogyny detection in English. Indeed, Ford and Boxer (2011) reveal that sexist jokes that in general are considered innocent, truthfully they are experienced by women as sexual harassment. In particular, inspired by Barbieri and Saggion (2014), we calculate the imbalance of the sentiment polarities (positive and negative) in each tweet using SentiWordNet provided by Baccianella et al. (2010). For each degree of imbalance, we associate a weight used in the vectorial representation of the tweets. Despite our hypothesis is well funded, we obtained lower results for the runs that contain sentiment imbalance among the features (see Table 4).

Italian Features. For the Italian language, we selected some specific issue groups, described in Bassignana et al. (2018), from the Italian lexicon "Le parole per ferire" provided by Tullio De Mauro[3]. In particular, we consider the lists of words described in Table 3. Differently from English, the experiments reveal that: the UBT is useful for both tasks and the best range for BoC is from 3 to 5 grams[4]. Indeed, in a morphological complex language like Italian the desinences of the words (such as the extracted n-grams "tona" or "ana ") contain relevant linguistic information. Diversely, in English, longer sequences of characters could help to capture multi-word expressions containing also pronouns, adjectives or prepositions, such as "ing at" or "ss bitc".

To extract the features correctly, in order to train our models, we pre-process the data deleting emoticons, emojis and URLs. Indeed, from our experiments, the emoticons and emojis do not prove to be relevant for these tasks. In order to perform a correct match between the dictionaries of the corpora and the single lexicon, we use the lemmatizer provided by the Natural Language Toolkit (NLTK[5]) for English, and the Snowball Stemmer for Italian. Differently from English, the use of lemmatizer for Italian tweets hinders the match.

[3] http://www.internazionale.it/opinione/tullio-de-mauro/2016/09/27/razzismo-parole-ferire

[4] The experiments are carried out using the Grid Search.

[5] http://www.nltk.org/

Lexicons	*Words*	*Definition*
Sexuality	290	contains words relative to sexual subject ($orgasm$, $orgy$, $pussy$) and especially male domination on women ($rape$, $pimp$, $slave$)
Profanity	170	is a collection of vulgar words such as $motherfucker$, $slut$ and $scum$
Femininity	90	is a list of terms used to identify the women as target. It contains personal pronouns or possessive adjectives (such as she, her, $herself$), common words used to refer to women ($girl$, $mother$) and also offensive words towards women (such as $barbie$, $hooker$ or $non-male$)
Human body	50	is a lexicon strongly connected with sexuality collecting words referred especially to feminine body also with negative connotations (such as $holes$, $throat$ or $boobs$)

Table 2: Composition of English lexica.

Lexicons	*Words*	*Definition*
AN	111	collects words relative to animals, such as $sanguisuga$ or $pecora$
ASF	31	contains terms referred to female genitalia, such as $fessa$
ASM	76	contains terms referred to male genitalia, such as $verga$
CDS	298	is a list of derogatory words, such as $bastardo$ or $spazzatura$
OR	17	contains words derived from plants but that are used as offensive words, such as $finocchio$ or $rapa$
PA	83	is a list of professions or jobs that have also a negative connotations, such as $portinaia$ or $impiegato$
PR	54	contains terms about prostitution, such as $bagascia$ or $zoccolona$
PS	42	is a list of words relative to stereotypes, such as $negro$ or $ostrogoto$
QAS	82	collects words that have in general negative connotations, such as $parassita$ or $dilettante$
RE	37	contains terms relative to criminal acts or immoral actions, such as $stupro$ or $violento$

Table 3: Composition of Italian lexica.

3.2 Approach 2: using automatically-enriched lexica (AEL)

The second approach aims to deal with the dynamism of the informal language online trying to capture new words relative to contexts defined in each lexicon. Therefore, we use enriched versions of the original lexica (described above), and stylistic and linguistic information captured by means of n-grams of words and characters as in the first approach. The method for the expansion of a given lexicon shares the idea of identifying new words by considering their contextual similarity with known words, as defined by some pre-trained word embeddings. For its description, let assume that $\mathcal{L} = \{l_1, \ldots, l_m\}$ is the initial lexicon of m words, and $\mathcal{W} = \{(w_1, e(w_1)), \ldots, (w_n, e(w_n))\}$ is the set of pre-trained word embeddings, where each pair represents a word and its corresponding embedding vector. This method aims to enrich the lexicon with words strongly related to the context from the original lexicon without being necessarily associated to any particular word. Its idea is to search for words having similar contexts to the entire lexicon. This method has two main steps, described below.

Dictionary modeling. Firstly, we extract the embedding $e(l_i)$ for each word $l_i \in \mathcal{L}$; then, we compute the average of these vectors to obtain a vector describing the entire lexicon, $e(\mathcal{L})$. We name this vector the context embedding.

Dictionary expansion. Using the cosine similarity, we compare $e(\mathcal{L})$ against the embedding $e(w_i)$ of each $w_i \in \mathcal{W}$; then, we extract the k most similar words to $e(\mathcal{L})$, defining the set $E_L = (w_1, \ldots, w_k)$. Finally, we insert the extracted words into the original lexicon to build the new lexicon, i.e., $\mathcal{L}_\mathcal{E} = \mathcal{L} \cup E_L$.

Therefore, we carry out the experiments using different pre-trained word embeddings for each language: *GloVe* embeddings trained on 2 billion tweets (Pennington et al., 2014) for English, and word embeddings built on TWITA corpus[6] for Italian (Basile and Novielli, 2014). Finally, the proposed expansion method is parametric and requires a value for k, the number of words that are going to extend the lexica. In particular, we use $k = 1000$, 500 and 100.

3.3 Experiments and Results

To carry out the experiments, a SVM classifier is employed with the radial basis function kernel (RBF) using the following parameters: $C = 5$ and $\gamma = 0.1$ for English and $\gamma = 0.01$ for Italian. Considering the complexity of the target classification for the Italian language due to imbalanced training set (see Table 1), we used a Random Forest (RF) classifier that aggregates the votes from different

[6] http://valeriobasile.github.io/twita/about.html

decision trees to decide the final class of the tweet.

The evaluation is performed using the test set provided by the organizers of the AMI shared task. For the competition, they use as evaluation measures the Accuracy for Task A and the average of F-score of both classes for Task B.

English			
Run	Approach	Accuracy	Rank
run 2[7]	AEL	**0.613**	**17**
baseline_AMI		*0.605*	*19*
run 1	AEL	0.592	21
run 3	MML	0.584	25
Italian			
Run	Approach	Accuracy	Rank
baseline_AMI		*0.830*	*7*
run 1	AEL	0.824	9
run 3[8]	AEL	0.823	11
run 2	MML	0.822	12

Table 4: Results obtained in Task A.

Table 4 and Table 5 show the results obtained in the competition compared with the baselines provided by the organizers for each task. Comparing the two approaches, in general AEL seems to work better than MML. However, the improvement of the results is very slight, especially for Italian language. This soft variation is unexpected considered the results obtained during the experiments employing 10-fold cross validations. In fact, AEL with enriched lexica using k equal 100 performed an Accuracy of 0.880. Moreover, looking at Table 4, reporting the official results of the AMI Task, only run 2 overcomes the baseline for the detection of misogyny in English, and for this run we used AEL approach excluding the sentiment imbalance as feature. About the identification of misogyny in Italian, the obtained results are lower than provided baselines as well as the values of F-score obtained in Task B for both languages (see Table 5). Despite the usefulness of lexica for a specific domain like misogyny, a lexicon-based approach proves to be insufficient for this task. Indeed, as the error analysis will confirm, misogyny, as well as general hate speech, involves linguistic devices such as humour, exclamations typical of orality and contextual information that completes the meaning transmitted by the tweet. Moreover, the low values obtained also in Task B suggest the necessity to implement dedicated approach for each misogynistic category.

[7]This run does not involve the sentiment imbalance

[8]This run involves the expansions of lexica with $k = 100$

4 Discussion and Conclusions

This paper reports our participation in the AMI shared task. The organizers provide also the gold test set that helps us to understand better what are the misclassified cases and the aspects that should be considered in the next experiments. Carrying out the error analysis, we notice that in both datasets the content of URL affects the transmitted information in the tweet (such as *Right! As they rape and butcher women and children !!!!!! https://t.co/maEhwuYQ8B*). The swear words are often used also as exclamation without the aim to offend (such as *Volevo dire alla Yamamay che tettona non sinonimo di curvy dato che di vita ha una 40, quindi confidence sta minchia.*). Moreover, despite the actual English corpus does not contain several jokes, Italian misclassified tweets involve humourous utterances (such as *@GrianneOhmsfor1 @BarbaraRaval A parte il fatto poi che culona inchiavabile" è il miglior giudizio politico sentito sulla Merkel negli ultimi anni??"*). In fact, in general, humour, irony and sarcasm hinder the correct classification of the texts, as we noticed in English and Spanish corpora provided in the IberEval framework. Participating in this shared task gave us the opportunity to analyze and compare multilingual datasets, and thus, to discover and infer general aspects typical of hate speech against women.

Acknowledgments

The work of Simona Frenda was partially funded by the Spanish research project SomEMBED TIN2015-71147-C2-1-P (MINECO/FEDER). We also thank the support of CONACYT-Mexico (projects FC-2410, CB-2015-01-257383).

References

Maria Anzovino, Elisabetta Fersini, and Paolo Rosso. 2018. Automatic identification and classification of misogynistic language on twitter. In *International Conference on Applications of Natural Language to Information Systems*, pages 57–64.

Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. 2010. Sentiwordnet 3.0: an enhanced lexical resource for sentiment analysis and opinion mining. In *Lrec*, volume 10, pages 2200–2204.

Francesco Barbieri and Horacio Saggion. 2014. Modelling irony in twitter. In *Proceedings of the Stu-*

English						
Run	Categories	F-score	Target	F-score	total	ranks
baseline_AMI		*0.342*		*0.399*	*0.370*	*3*
run 2	UBT	0.282	UBT+BoC	**0.407**	0.344	6
run 1	UBT	0.282	UBT+BoC	0.389	0.335	8
run 3	UBT	0.269	UBT+BoC	0.387	0.328	10
Italian						
Run	Categories	F-score	Target	F-score	Total	ranks
baseline_AMI		*0.534*		*0.440*	*0.487*	*2*
run 3	UBT+BoC	0.485	UBT+BoC	0.414	0.449	7
run 1	UBT+BoC	0.483	UBT+BoC	0.414	0.448	8
run 2	UBT+BoC	0.480	UBT+BoC	0.411	0.446	10

Table 5: Results obtained in Task B.

dent Research Workshop at the 14th Conference of the European Chapter of the ACL.

Pierpaolo Basile and Nicole Novielli. 2014. Uniba at evalita 2014-sentipolc task: Predicting tweet sentiment polarity combining micro-blogging, lexicon and semantic features. In *Proceedings of EVALITA 2014*.

Elisa Bassignana, Valerio Basile, and Patti Viviana. 2018. Hurtlex: A multilingual lexicon of words to hurt. In *Proceedings of CLiC-it, Turin, 10-12 December 2018, CEUR*.

Peter Burnap and Matthew Leighton Williams. 2014. Hate speech, machine classification and statistical modelling of information flows on twitter: Interpretation and communication for policy decision making. *Internet, Policy & Politics*.

Ying Chen, Yilu Zhou, Sencun Zhu, and Heng Xu. 2012. Detecting offensive language in social media to protect adolescent online safety. In *Privacy, Security, Risk and Trust (PASSAT)*, pages 71–80. IEEE.

Fabio Del Vigna, Andrea Cimino, Felice Dell'Orletta, Marinella Petrocchi, and Maurizio Tesconi. 2017. Hate me, hate me not: Hate speech detection on facebook. In *Proceedings of ITASEC17*.

Hugo Jair Escalante, Esaú Villatoro-Tello, Sara E Garza, A Pastor López-Monroy, Manuel Montes-y Gómez, and Luis Villaseñor-Pineda. 2017. Early detection of deception and aggressiveness using profile-based representations. *Expert Systems with Applications*, 89:99–111.

Elisabetta Fersini, Maria Anzovino, and Paolo Rosso. 2018a. Overview of the task on automatic misogyny identification at ibereval. In *Proceedings of Workshop IBEREVAL at 3rd SEPLN*.

Elisabetta Fersini, Debora Nozza, and Paolo Rosso. 2018b. Overview of the evalita 2018 task on automatic misogyny identification (ami). In Tommaso Caselli, Nicole Novielli, Viviana Patti, and Paolo Rosso, editors, *Proceedings of the 6th evaluation campaign of Natural Language Processing and Speech tools for Italian (EVALITA'18)*, Turin, Italy. CEUR.org.

Thomas E Ford and Christie Fitzgerald Boxer. 2011. Sexist humor in the workplace: A case of subtle harassment. In *Insidious Workplace Behavior*, pages 203–234. Routledge.

Jesse Fox, Carlos Cruz, and Ji Young Lee. 2015. Perpetuating online sexism offline: Anonymity, interactivity, and the effects of sexist hashtags on social media. *Computers in Human Behavior*, 52:436–442.

Simona Frenda, Bilal Ghanem, and Manuel Montes-y Gómez. 2018. Exploration of misogyny in spanish and english tweets. In *Proceedings of Workshop IBEREVAL at 3rd SEPLN*.

Njagi Dennis Gitari, Zhang Zuping, Hanyurwimfura Damien, and Jun Long. 2015. A lexicon-based approach for hate speech detection. *International Journal of Multimedia and Ubiquitous Engineering*, 10(4):215–230.

Estefanía Guzmán Falcón. 2018. *Detección de lenguaje ofensivo en Twitter basada en expansión automática de lexicones (tesis de maestría)*. Instituto Nacional de Astrofísica, Óptica y Electrónica. Puebla, México.

Yashar Mehdad and Joel Tetreault. 2016. Do characters abuse more than words? In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 299–303.

Chikashi Nobata, Joel Tetreault, Achint Thomas, Yashar Mehdad, and Yi Chang. 2016. Abusive language detection in online user content. In *Proceedings of the 25th international conference on WWW*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of EMNLP*.

Bailey Poland. 2016. *Haters: Harassment, abuse, and violence online*. U of Nebraska Press.

Niloofar Safi Samghabadi, Suraj Maharjan, Alan Sprague, Raquel Diaz-Sprague, and Thamar Solorio. 2017. Detecting nastiness in social media. In *Proceedings of ALW1*.

Detecting Hate Speech Against Women in English Tweets

Resham Ahluwalia, Himani Soni, Edward Callow, Anderson Nascimento, Martine De Cock*
School of Engineering and Technology
University of Washington Tacoma
{resh,himanis7,ecallow,andclay,mdecock}@uw.edu

Abstract

English. Hate speech is prevalent in social media platforms. Systems that can automatically detect offensive content are of great value to assist human curators with removal of hateful language. In this paper, we present machine learning models developed at UW Tacoma for detection of misogyny, i.e. hate speech against women, in English tweets, and the results obtained with these models in the shared task for Automatic Misogyny Identification (AMI) at EVALITA2018.

Italiano. *Commenti offensivi nei confronti di persone con diversa orientazione sessuale o provenienza sociale sono oggigiorno prevalenti nelle piattaforme di social media. A tale fine, sistemi automatici in grado di rilevare contenuti offensivi nei confronti di alcuni gruppi sociali sono importanti per facilitare il lavoro dei moderatori di queste piattaforme a rimuovere ogni commento offensivo usato nei social media. In questo articolo, vi presentiamo sia dei modelli di apprendimento automatico sviluppati all'Università di Washington in Tacoma per il rilevamento della misoginia, ovvero discorsi offensivi usati nei tweet in lingua inglese contro le donne, sia i risultati ottenuti con questi modelli nel processo per l'identificazione automatica della misoginia in EVALITA2018.*

1 Introduction

Inappropriate user generated content is of great concern to social media platforms. Although social media sites such as Twitter generally prohibit hate speech[1], it thrives online due to lack of accountability and insufficient supervision. Although social media companies hire employees to moderate content (Gershgorn and Murphy, 2017), the number of social media posts exceeds the capacity of humans to monitor without the assistance of automated detection systems.

In this paper, we focus on the automatic detection of misogyny, i.e. hate speech against women, in tweets that are written in English. We present machine learning (ML) models trained for the tasks posed in the competition for Automatic Misogyny Identification (AMI) at EVALITA2018 (Fersini et al., 2018b). Within this competition, Task A was the binary classification problem of labeling a tweet as misogynous or not. As becomes clear from Table 1, Task B consisted of two parts: the multiclass classification problem of assigning a misogynous tweet to the correct category of misogyny (e.g. sexual harassment, stereotype, ...), and the binary classification problem of determining whether a tweet is actively targeted against a specific person or not.

Interest in the use of ML for automatic detection of online harassment and hate speech is fairly recent (Razavi et al., 2010; Nobata et al., 2016; Anzovino et al., 2018; Zhang and Luo, 2018). Most relevant to our work are approaches published in the context of a recent competition on automatic misogyny identification organized at IberEval2018 (Fersini et al., 2018a), which posed the same binary classification and multiclass classification tasks addressed in this paper. The AMI-baseline system for each task in the AMI@IberEval competition was an SVM trained on a unigram representation of the tweets, where each tweet was represented as a bag of words (BOW) composed of 1000 terms. We participated in the AMI@IberEval competition with an Ensem-

*Guest Professor at Dept. of Applied Mathematics, Computer Science and Statistics, Ghent University

[1] https://help.twitter.com/en/rules-and-policies/twitter-rules

Task A: Misogyny	Train	Test	**Task B: Category**	Train	Test	**Task B: Target**	Train	Test
Non-misogynous	2215	540	0	2215	540	0	2215	540
Misogynous	1785	460	Discredit	1014	141	Active	1058	401
			Sexual harassment	352	44	Passive	727	59
			Stereotype	179	140			
			Dominance	148	124			
			Derailing	92	11			

Table 1: Distribution of tweets in the dataset

ble of Classifiers (EoC) containing a Logistic Regression model, an SVM, a Random Forest, a Gradient Boosting model, and a Stochastic Gradient Descent model, all trained on a BOW representation of the tweets (composed of both word unigrams and word bigrams) (Ahluwalia et al., 2018). In AMI@IberEval, our team *resham* was the 7th best team (out of 11) for Task A, and the 3rd best team (out of 9) for Task B. The winning system for Task A in AMI@IberEval was an SVM trained on vectors with lexical features extracted from the tweets, such as the number of swear words in the tweet, whether the tweet contains any words from a lexicon with sexist words, etc. (Pamungkas et al., 2018). Very similarly, the winning system for the English tweets in Task B in AMI@IberEval was also an SVM trained on lexical features derived from the tweets, using lexicons that the authors built specifically for the competition (Frenda et al., 2018).

For the AMI@EVALITA competition, which is the focus of the current paper, we experimented with the extraction of lexical features based on dedicated lexicons as in (Pamungkas et al., 2018; Frenda et al., 2018). For Task A, we were the 2nd best team (*resham.c.run3*), with an EoC approach based on BOW features, lexical features, and sentiment features. For Task B, we were the winning team (*himani.c.run3*) with a two-step approach: for the first step, we trained an LSTM (Long Short-Term Memory) neural network to classify a tweet as misogynous or not; tweets that are labeled as misogynous in step 1 are subsequently assigned a category and target label in step 2 with an EoC approach trained on bags of words, bigrams, and trigrams. In Section 2 we provide more details about our methods for Task A and Task B. In Section 3 we present and analyze the results.

2 Description of the System

The training data consists of 4,000 labeled tweets that were made available to participants in the AMI@EVALITA competition. As Table 1 shows, the distribution of the tweets over the various labels is imbalanced; the large majority of misogynistic tweets in the training data for instance belong to the category "Discredit". In addition, the distribution of tweets in the test data differs from that in the training data. As the ground truth labels for the test data were only revealed after the competition, we constructed and evaluated the ML models described below using 5-fold cross-validation on the training data.

2.1 Task A: Misogyny

Text Preprocessing. We used NLTK[2] to tokenize the tweets and to remove English stopwords.

Feature Extraction. We extracted three kinds of features from the tweets:

- *Bag of Word Features.* We turned the preprocessed tweets into BOW vectors by counting the occurrences of token unigrams in tweets, normalizing the counts and using them as weights.
- *Lexical Features.* Inspired by the work of (Pamungkas et al., 2018; Frenda et al., 2018), we extracted the following features from the tweets:
  - Link Presence: 1 if there is a link or URL present in the tweet; 0 otherwise.
  - Hashtag Presence: 1 if there is a Hashtag present; 0 otherwise.
  - Swear Word Count: the number of swear words from the *noswearing dictionary*[3] that appear in the tweet.
  - Swear Word Presence: 1 if there is a swear word from the *noswearing dictionary* present in the tweet; 0 otherwise.
  - Sexist Slur Presence: 1 if there is a sexist word from the list in (Fasoli et al., 2015) present in the tweet; 0 otherwise.
  - Women Word Presence: The feature value is 1 if there is a *woman synonym word* [4] present in the tweet; 0 otherwise.

[2] https://www.nltk.org/, TweetTokenizer
[3] https://www.noswearing.com/dictionary
[4] https://www.thesaurus.com/browse/woman

- *Sentiment scores.* We used SentiWordNet (Baccianella et al., 2010) to retrieve a positive and a negative sentiment score for each word occurring in the tweet, and computed the average of those numbers to obtain an aggregated positive score and an aggregated negative score for the tweet.

Model Training. We trained 3 EoC models for designating a tweet as misogynous or not (Task A). The EoC models differ in the kind of features they consume as well as in the kinds of classifiers that they contain internally.

- *EoC with BOW (resham.c.run2)*[5]*:* an ensemble consisting of a Random Forest classifier (RF), a Logistic Regression classifier (LR), a Stochastic Gradient Descent (SGD) classifier, and a Gradient Boosting (GB) classifier, each of them trained on the BOW features.
- *EoC with BOW and sentiment scores (resham.c.run1):* an ensemble consisting of the same 4 kinds of classifiers as above, each of them trained on the BOW and sentiment score features.
- *EoC with BOW, sentiment scores, and lexical features (resham.c.run3):* an ensemble consisting of
  - RF on the BOW and sentiment score features
  - SVM on the lexical features
  - GB on the lexical features
  - LR on the lexical features.
  - GB on the BOW and sentiment features

All the ensembles use hard voting. For training the classifiers we used scikit-learn (Pedregosa et al., 2011) with the default choices for all parameters.

2.2 Task B: Category And Target

For Task B, our winning system *himani.c.run3* consists of a pipeline of two classifiers: the first classifier (step 1) in the pipeline labels a tweet as misogynous or not, while the second classifier (step 2) assigns the tweets that were labeled misogynous to their proper category and target.

For Step 1 we trained a deep neural network that consists of a word embedding layer, followed by a bi-directional LSTM layer with 50 cells, a hidden dense layer with 50 cells with relu activation, and an output layer with sigmoid activation. For the embedding layer we used the pretrained Twitter Embedding from the GloVe package (Pennington et al., 2014), which maps each word to a 100-dimensional numerical vector. The LSTM network is trained to classify tweets as misogynous or not. We participated with this trained network in Task A of the competition as well (*himani.c.run3*). The results were not as good as those obtained with the models described in Section 2.1, so we do not go into further detail.

Next we describe how we trained the models used in Step 2 in *himani.c.run3*.

Text Preprocessing. We used the same text preprocessing as in Section 2.1. In addition we removed words occurring in more than 60 percent of the tweets along with those that had a word frequency less than 4.

Feature Extraction. We turned the preprocessed tweets into *Bag of N-Gram* vectors by counting the occurrences of token unigrams, bigrams and trigrams in tweets, normalizing the counts and using them as weights. For simplicity, we keep referring to this as a BOW representation.

Model Training. For category and target identification, *himani.c.run3* uses an EoC approach where all classifiers are trained on the BOW features mentioned above. The EoC models for category identification on one hand, and target detection on the other hand, differ in the classifiers they contain internally, and in the values of the hyperparameters. Below we list parameter values that differ from the default values in scikit-learn (Pedregosa et al., 2011).

- *EoC for Category Identification:*
  - LR: inverse of regularization strength C is 0.7; norm used in the penalization is L1; optimization algorithm is 'saga'.
  - RF: number of trees is 250; splitting attributes are chosen based on information gain.
  - SGD: loss function is 'modified huber'; constant that multiplies the regularization term is 0.01; maximum number of passes over the training data is 5.
  - Multinomial Naive Bayes: all set to defaults.
  - XGBoost: maximum depth of tree is 25; number of trees is 200.
- *EoC for Target Identification:*

[5]Here 'resham.c.run2' refers to the second run of the data submitted by the author in connection with the competition. Similar citations that follow have a corresponding meaning.

Approach	5-fold CV on Train	Test
majority baseline	0.553	0.540
resham.c.run1	0.790	0.648
resham.c.run2	0.787	0.647
resham.c.run3	**0.795**	**0.651**
himani.c.run3	0.785	0.614

Table 2: Accuracy results for Task A: Misogyny detection on English tweets.

- LR: inverse of regularization strength C is 0.5; norm used in the penalization is L1; optimization algorithm is 'saga'.
- RF: number of trees is 200; splitting attributes are chosen based on information gain.

For completeness we mention that *himani.c.run2* consisted of a two-step approach very similar to the one outlined above. In Step 1 of *himani.c.run2* tweets are labeled as misogynous or not with an EoC model (RF, XGBoost) trained on the Bag of N-Gram features. In Step 2, a category and target label are assigned with respectively an LR, XGBoost-EoC model and an LR, RF-EoC model in which all classifiers are trained on the Bag of N-Gram features as well. Since this approach is highly similar to the *himani.c.run3* approach described above and did not give better results, we do not go into further detail.

3 Results and Discussion

3.1 Results for Task A

Table 2 presents accuracy results for Task A, i.e. classifying tweets as misogynous or not, evaluated with 5-fold cross-validation (CV) on the 4,000 tweets in the training data from Table 1. In addition, the last column of Table 2 contains the accuracy when the models are trained on all 4,000 tweets and subsequently applied to the test data. We include a simple majority baseline algorithm that labels all tweets as non-misogynous, which is the most common class in the training data.

The accuracy on the test data is noticeably lower than the accuracy obtained with 5-fold CV on the training data. At first sight, this is surprising because the label distributions are very similar: 45% of the training tweets are misogynous, and 46% of the testing tweets are misogynous. Looking more carefully at the distribution across the different categories of misogyny in Table 1, one can observe that the training and test datasets do vary quite a lot in the kind (category) of misogyny. It is plausible that tweets in different misogyny categories are characterized by their own, particular language, and that during training our binary classifiers have simply become good at flagging misogynous tweets from categories that occur most often in the training data, leaving them under-prepared to detect tweets from other categories.

Regardless, one can see that the ensembles benefit from having more features available. Recall that *resham.c.run2* was trained on BOW features, *resham.c.run1* on BOW features and sentiment scores, and *resham.c.run3* on BOW features, sentiment scores, and lexical features. As is clear from Table 2, the addition of each feature set increases the accuracy. As already mentioned in Section 2.2, the accuracy of *himani.c.run3*, which is a bidirectional LSTM that takes tweets as strings of words as its input, is lower than that of the *resham* models, which involve explicit feature extraction.

3.2 Results for Task B

Table 3 contains the results of our models for Task B in terms of F1-scores. Following the approach used on the AMI@EVALITA scoreboard, both subtasks are evaluated as multiclass classification problems. For Category detection, there are 6 possible class labels, namely the label 'non-misogynous' and each of the 5 category labels. Similarly, for Target detection, there are 3 possible class labels, namely 'non-misogynous', 'Active', and 'Passive'.

When singling out a specific class c as the "positive" class, the corresponding F1-score for that class is defined as usual as the harmonic mean of the precision and recall for that class. These values are computed treating all tweets with ground truth label c as positive examples, and all other tweets as negative examples. For example, when computing the F1-score for the label "Sexual harassment" in the task of Category detection, all tweets with ground truth label "Sexual harassment" are treated as positive examples, while the tweets from the other 4 categories of misogyny *and* the non-misogynous tweets are considered to be negative examples. The average of the F1-scores computed in this way for the 5 categories of misogyny is reported in the columns F1 (Category) in Table 3, while the average of the F1-scores for 'Active' and 'Passive' is reported in the columns F1 (Target) in Table 3. The first columns contain results obtained

	5-fold CV on Train			Test		
Approach	**F1 (Category)**	**F1 (Target)**	**Average F1**	**F1 (Category)**	**F1 (Target)**	**Average F1**
majority baseline	0.079	0.209	0.135	0.049	0.286	0.167
himani.c.run2	0.283	0.622	0.452	0.323	0.431	0.377
himani.c.run3	**0.313**	**0.626**	**0.469**	**0.361**	**0.451**	**0.406**
Step 1 from resham.c.run3 & Step 2 from himani.c.run3	0.278	0.515	0.396	0.246	0.361	0.303

Table 3: F1-score results for Task B on English tweets

	5-fold CV on Train						Test					
Approach	**Pr (A)**	**Re (A)**	**F1 (A)**	**Pr (P)**	**Re (P)**	**F1 (P)**	**Pr (A)**	**Re (A)**	**F1 (A)**	**Pr (P)**	**Re (P)**	**F1 (P)**
himani.c.run3	0.61	0.79	0.69	0.53	0.56	0.54	0.61	0.75	0.67	0.14	0.61	0.23
Step 1 from resham.c.run3 & Step 2 from himani.c.run3	0.70	0.70	0.70	0.51	0.31	0.39	0.67	0.45	0.54	0.17	0.19	0.18

Table 4: Detailed precision (Pr), recall Re), and F1-score (F1) results for Task B: Target Identification on English tweets; 'A' and 'P' refer to 'Active' and 'Passive' respectively.

		Predicted value		
		N	**A**	**P**
Actual value	**N**	202	176	162
	A	40	301	60
	P	8	15	36

Table 5: Confusion matrix for Task B: Target Identification with *himani.c.run3* on the test data; 'N', 'A', and 'P' refer to 'Non-misogynous', 'Active' and 'Passive' respectively.

		Predicted value		
		N	**A**	**P**
Actual value	**N**	428	78	34
	A	201	182	18
	P	38	10	11

Table 6: Confusion matrix for Task B: Target Identification with Step 1 from *resham.c.run3* and Step 2 from *himani.c.run3* on the test data; 'N', 'A', and 'P' refer to 'Non-misogynous', 'Active' and 'Passive' respectively.

with 5-fold CV over the training data with 4,000 tweets from Table 1, while the last columns contain results for models trained on the entire training data of 4,000 tweets and subsequently applied to the test data. The latter correspond to the results on the competition scoreboard.

As a simple baseline model, we include an algorithm that labels every tweet as misogynous and subsequently assigns it to the most frequently occurring Category and Target from the training data, i.e. 'Discredit' and 'Active'. This model has a very low precision, which explains why its F1-scores are so low. The best results on the test data are obtained with *himani.c.run3*, which is an EoC approach using a BOW representation of extracted word unigrams, bigrams, and trigrams as features. This was the best performing model for Task B in the AMI@EVALITA competition.

Recall that *himani.c.run3* uses a two step approach where tweets are initially labeled as misogynous or not (Step 1) and then assigned to a Category and Target (Step 2). Given that for the task in Step 1, the binary classifier of *himani.c.run3* was outperformed in terms of accuracy by the binary classifier of *resham.c.run3* (see Table 2), an obvious question is whether higher F1-scores for Task B could be obtained by combining the binary classifier for misogyny detection from *resham.c.run3* with the EoC models for Category and Target identification from *himani.c.run3*. As the last row in Table 3 shows, this is not the case. To give more insight into where the differences in predictive performance in the last two rows of Table 3 stem from, Table 4 contains more detailed results about the precision, recall, and F1-scores for Task B: Target Identification on the train as well as the test data, while Table 5 and 6 contain corresponding confusion matrices on the test data. These tables reveal that the drop in F1-scores in the last row in Table 3 is due to a substantial drop

in recall. As can be seen in Table 4, replacing the binary classifier in Step 1 by the method from *resham.c.run3*, causes the recall for 'Active' tweets in the test data to drop from 0.75 to 0.45, and for 'Passive' tweets from 0.61 to 0.19. The slight increase in precision is not sufficient to compensate for the loss in recall. As can be inferred from Table 5 and 6, the recall of misogynous tweets overall with *himani.c.run3* is $(301 + 60 + 15 + 36)/460 \approx 0.896$ while with *resham.c.run3* it is only $(182 + 18 + 10 + 11)/460 \approx 0.480$.

4 Conclusion

In this paper we presented machine learning models developed at UW Tacoma for detection of hate speech against women in English language tweets, and the results obtained with these models in the shared task for Automatic Misogyny Identification (AMI) at EVALITA2018. For the binary classification task of distinguishing between misogynous and non-misogynous tweets, we obtained our best results (2nd best team) with an Ensemble of Classifiers (EoC) approach trained on 3 kinds of features: bag of words, sentiment scores, and lexical features. For the multiclass classification tasks of Category and Target Identification, we obtained our best results (winning team) with an EoC approach trained on a bag of words representation containing unigrams, bigrams, and trigrams. All EoC models contain traditional machine learning classifiers, such as logistic regression and tree ensemble models.

Thus far, the success of our deep learning models has been modest. This could be due to the limited size of the dataset and/or the limited length of tweets. Regarding the latter, an interesting direction to explore next is training neural networks that can consume the tweets at character level instead of at word level, as we did in this paper.

References

Resham Ahluwalia, Evgeniia Shcherbinina, Edward Callow, Anderson Nascimento, and Martine De Cock. 2018. Detecting misogynous tweets. In *Proc. of IberEval 2018*, volume 2150 of *CEUR-WS*, pages 242–248.

Maria Anzovino, Elisabetta Fersini, and Paolo Rosso. 2018. Automatic identification and classification of misogynistic language on Twitter. In *International Conference on Applications of Natural Language to Information Systems*, pages 57–64. Springer.

Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. 2010. Sentiwordnet 3.0: an enhanced lexical resource for sentiment analysis and opinion mining. In *Lrec*, volume 10, pages 2200–2204.

Fabio Fasoli, Andrea Carnaghi, and Maria Paola Paladino. 2015. Social acceptability of sexist derogatory and sexist objectifying slurs across contexts. *Language Sciences*, 52:98–107.

Elisabetta Fersini, M Anzovino, and P Rosso. 2018a. Overview of the task on automatic misogyny identification at IberEval. In *Proc. of IberEval 2018*, volume 2150 of *CEUR-WS*, pages 214–228.

Elisabetta Fersini, Debora Nozza, and Paolo Rosso. 2018b. Overview of the Evalita 2018 Task on Automatic Misogyny Identification (AMI). In Tommaso Caselli, Nicole Novielli, Viviana Patti, and Paolo Rosso, editors, *Proceedings of the 6th evaluation campaign of Natural Language Processing and Speech tools for Italian (EVALITA'18)*, Turin, Italy. CEUR.org.

Simona Frenda, Bilal Ghanem, and Manuel Montes-y Gómez. 2018. Exploration of misogyny in Spanish and English tweets. In *Proc. of IberEval 2018*, volume 2150 of *CEUR-WS*, pages 260–267.

Dave Gershgorn and Mike Murphy. 2017. Facebook is hiring more people to moderate content than Twitter has at its entire company. `https://qz.com/1101455/facebook-fb-is-hiring-more-people-to-moderate-content-than-twitter-twtr-has-at-its-entire-company/`.

Chikashi Nobata, Joel Tetreault, Achint Thomas, Yashar Mehdad, and Yi Chang. 2016. Abusive language detection in online user content. In *Proc. of the 25th International Conference on World Wide Web*, pages 145–153.

Endang Wahyu Pamungkas, Alessandra Teresa Cignarella, Valerio Basile, and Viviana Patti. 2018. 14-ExLab@UniTo for AMI at IberEval2018: Exploiting lexical knowledge for detecting misogyny in English and Spanish tweets. In *Proc. of IberEval 2018*, volume 2150 of *CEUR-WS*, pages 234–241.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proc. of EMNLP*, pages 1532–1543.

Amir H Razavi, Diana Inkpen, Sasha Uritsky, and Stan Matwin. 2010. Offensive language detection using multi-level classification. In *Canadian Conference on Artificial Intelligence*, pages 16–27. Springer.

Ziqi Zhang and Lei Luo. 2018. Hate speech detection: A solved problem? The challenging case of long tail on Twitter. *arXiv preprint arXiv:1803.03662*.

Automatic Identification of Misogyny in English and Italian Tweets at EVALITA 2018 with a Multilingual Hate Lexicon

Endang Wahyu Pamungkas[1], **Alessandra Teresa Cignarella**[1,2], **Valerio Basile**[1] **and Viviana Patti**[1]
[1]Dipartimento di Informatica, Università degli Studi di Torino
[2]PRHLT Research Center, Universitat Politècnica de València
{pamungka | cigna | basile | patti}@di.unito.it

Abstract

English. In this paper we describe our submission to the shared task of Automatic Misogyny Identification in English and Italian Tweets (AMI) organized at EVALITA 2018. Our approach is based on SVM classifiers and enhanced by stylistic and lexical features. Additionally, we analyze the use of the novel *HurtLex* multilingual linguistic resource, developed by enriching in a computational and multilingual perspective of the hate words Italian lexicon by the linguist Tullio De Mauro, in order to investigate its impact in this task.

Italiano. *Nel presente lavoro descriviamo il sistema inviato allo shared task di Automatic Misogyny Identification (AMI) ad EVALITA 2018. Il nostro approccio si basa su classificatori SVM, ottimizzati da feature stilistiche e lessicali. Inoltre, analizziamo il ruolo della nuova risorsa linguistica* HurtLex*, un'estensione in prospettiva computazionale e multilingue del lessico di* parole per ferire *in italiano proposto dal linguista Tullio De Mauro, per meglio comprendere il suo impatto in questo tipo di task.*

1 Introduction

Hate Speech (HS) can be based on race, skin color, ethnicity, gender, sexual orientation, nationality, or religion, it incites to violence and discrimination, abusive, insulting, intimidating, and harassing. Hateful language is becoming a huge problem in social media platforms such as Twitter and Facebook (Poland, 2016). In particular, a type of cyberhate that is increasingly worrying nowadays is the use of hateful language that specifically targets women, which is normally referred to as: MISOGYNY (Bartlett et al., 2014).

Misogyny can be linguistically manifested in numerous ways, including social exclusion, discrimination, hostility, threats of violence and sexual objectification (Anzovino et al., 2018). Many Internet companies and micro-blogs already tried to tackle the problem of blocking this kind of online contents, but, unfortunately, the issue is far from being solved because of the complexity of the natural language[1] (Schmidt and Wiegand, 2017). For the above-mentioned reasons, it has become necessary to implement targeted NLP techniques that can be automated to treat hate speech online and misogyny.

The first shared task specifically aimed at Automatic Misogyny Identification (AMI) took place at IberEval 2018[2] within SEPLN 2018 considering English and Spanish tweets (Fersini et al., 2018a). Hence, the aim of the proposed shared task is to encourage participating teams in proposing the best automatic system firstly to distinguish misogynous and non-misogynous tweets, and secondly to classify the type of misogynistic behaviour and judge whether the target of the misogynistic behaviour is a specific woman or a group of women. In this paper, we describe our submission to the 2nd shared task of Automatic Misogyny Identification (AMI)[3] organized at EVALITA 2018, organized in the same manner but focusing on Italian tweets, rather than Spanish and English as in the IberEval task.

2 Task Description

The aim of the AMI task is to detect misogynous tweets written in English and Italian (Task A) (Fersini et al., 2018b). Furthermore, in Task

[1]https://www.nytimes.com/2013/05/29/business/media/facebook-says-it-failed-to-stop-misogynous-pages.html
[2]https://sites.google.com/view/ibereval-2018
[3]https://amievalita2018.wordpress.com/

B, each system should also classify each misogynous tweet into one of five different misogyny behaviors (STEREOTYPE, DOMINANCE, DERAILING, SEXUAL HARASSMENT, AND DISCREDIT) and two targets of misogyny classes (active and passive). Participants are allowed to submit up to three runs for each language. Table 1 shows the dataset label distribution for each class. Accuracy will be used as an evaluation metric for Task A, while macro F-score is used for Task B.

The organizers provided the same amount of data for both languages: 4,000 tweets in the training set and 1,000 in the test set. The label distribution for Task A is balanced, while in Task B the distribution is highly unbalanced for both misogyny behaviors and targets.

3 Description of the System

We used two Support Vector Machine (SVM) classifiers which exploit different kernels: linear and radial basis function (RBF) kernels.

SVM with Linear Kernel. Linear kernel was used to find the optimal hyperplane when SVM was firstly introduced in 1963 by Vapnik et al., long before Cortes and Vapnik (1995) proposed to use the kernel trick. Joachims (1998) recommends to use linear kernel for text classification, based on the observation that text representation features are frequently linearly separable.

SVM with RBF Kernel. Choosing the kernel is usually a challenging task, because its performance will be dataset dependent. Therefore, we also experimenteed with a Radial Basis Function (RBF) kernel, which has been already proven as an effective classifier in text classification problems. The drawback of RBF kernels is that they are computationally expensive and obtain a worse performance in big and sparse feature matrices.

3.1 Features

We employed several lexical features, performing a simple preprocessing step including tokenization and stemming, using the NLTK (Natural Language Toolkit) library[4]. A detailed description of the features employed by our model follows.

Bag of Words (BoW). We used bags of words in order to build the tweets representation. Before producing the word vector, we changed all the characters from upper to lower case. Our vector space consists of the count of unigrams and bigrams as a representation of the tweet. In addition, we also employed **Bag of Hashtags (BoH)** and **Bag of Emojis (BoE)** features, which are built by using the same technique as BoW, focusing on the presence of hashtags and emojis.

Swear Words. This feature takes into account the presence of a swear word and the number of its occurrences in the tweet. For English, we took a list of swear words from `www.noswearing.com`, while for Italian we gathered the swear word list from several sources[5] including a translated version of `www.noswearing.com`'s list and a list of swear words from Capuano (2007).

Sexist Slurs. Beside swear words, we also considered sexist words, that are specifically targeting women. We used a small set of sexist slurs from previous work by Fasoli et al. (2015). We translated and expanded that list manually for our Italian systems. This feature has a binary value, 1 when at least one sexist slur presence on tweet and 0 when there is no sexist slur on tweet.

Women Words. We manually built a small set of words containing synonyms and several words related to word “woman" in English and “donna" in Italian. Based on our previous work (Pamungkas et al., 2018), these words were effective to detect the target of misogyny on the tweet. Similar to sexist slur feature, this feature also has binary value show the presence of women words on tweet.

Surface Features. We also considered several surface level features including: **upper case** character count, number of **hashtags**, number of **URLs**, and the **length** of the tweet counting the characters.

Hate Words Lexicon. HurtLex (Bassignana et al., 2018) is a multilingual lexicon of hate words, built starting from a list of words compiled manually (De Mauro, 2016). The lexicon is semi-automatically translated into 53 languages, and the lexical items are divided into 17 categories (see Table 2). For our system configuration, we exploited the presence of the words in each category as a single feature, thus obtaining 17 single features, one for each HurtLex category.

[4] `https://www.nltk.org/`

[5] `https://www.parolacce.org/2016/12/20/dati-frequenza-turpiloquio/` and `https://it.wikipedia.org/wiki/Turpiloquio_nella_lingua_italiana`

Task A			Task B		
	English	**Italian**		**English**	**Italian**
Misogynistic	1,785/460	1,828/512	Stereotype	179/140	668/175
			Dominance	148/124	71/61
			Derailing	92/11	24/2
			Sexual Harassment	352/44	431/170
			Discredit	1,014/141	634/104
			Active	1,058/401	1,721/446
			Passive	727/59	96/66
Not misogynistic	2,215/540	2,172/488	No class	2,215/540	2,172/488
Total				4,000/1,000	4,000/1,000

Table 1: Dataset label distribution (training/test).

Category	Description
PS	Ethnic Slurs
RCI	Location and Demonyms
PA	Profession and Occupation
DDP	Physical Disabilities and Diversity
DDF	Cognitive Disabilities and Diversity
DMC	Moral Behavior and Defect
IS	Words Related to Social and Economic antage
OR	Words Related to Plants
AN	Words Related to Animals
ASM	Words Related to Male Genitalia
ASF	Words Related to Female Genitalia
PR	Words Related Prostitution
OM	Words Related Homosexuality
QAS	Descriptive Words with Potential Negative Connotations
CDS	Derogatory Words
RE	Felonies and Words Related to Crime and Immoral Behavior
SVP	Words Related to the Seven Deadly Sins of the Christian Tradition

Table 2: HurtLex Categories.

4 Experimental Setup

We experimented with different sets of features and kernels to find the best configuration of the two SVM classifiers (one for each language of the task). A 10-fold cross validation was carried out to tune our systems based on accuracy. Our submitted systems configuration can be seen in Table 3.

Run #3 for both languages uses the same configuration of our best system at the IberEval task. (Fersini et al., 2018a).

The best result on the English training set has been obtained by run #1, where we used the RBF kernel (0.765 accuracy), while for Italian the best result has been obtained by runs #2 and #3 with the Linear kernel (0.893 accuracy). Different sets of categories from *Hurt* were able to improve the classifier performance, depending on the language.

In order to classify the category and target of misogyny (Task B), we adopted the same set of features as Task A. Therefore, we did not build new systems specifically for Task B.

We experimented with different selections of categories from the HurtLex lexicon, and identified the most useful for the purpose of misogyny identification. As it can be seen in Table 3, the main categories are: physical disabilities and diversity (DDP), words related to prostitution (PR), words referring to male genitalia (ASM) and female genitalia (ASF). But also: derogatory words (CDS), words related to felonies and crime, and also immoral behavior (RE).

Language	**English**			**Italian**		
Systems	run1	run2	**run3**	run1	run2	**run3**
Accuracy	0.765	0.72	0.744	0.786	0.893	0.893
Bag of Word	-	✓	-	-	✓	✓
Bag of Hashtags	-	-	-	-	-	✓
Bag of Emojis	-	-	-	-	-	✓
S.W. Count	✓	-	✓	✓	-	-
S.W. Presence	✓	-	✓	✓	-	-
Sexist Slurs	✓	✓	✓	✓	✓	-
Woman Word	✓	✓	✓	✓	✓	-
Hashtag	-	-	✓	-	✓	-
Link Presence	✓	✓	✓	-	-	-
Upper Case Count	✓	-	-	✓	✓	-
Text Length	-	✓	-	✓	-	-
ASF Count	✓	✓	-	✓	✓	✓
PR Count	-	-	-	✓	✓	✓
OM Count	✓	✓	-	-	-	-
DDF Count	-	-	-	-	-	-
CDS Count	✓	✓	-	✓	✓	-
DDP Count	✓	✓	-	-	-	✓
AN Count	✓	✓	-	-	-	-
ASM Count	-	-	-	✓	✓	-
DMC Count	-	-	-	-	-	-
IS Count	✓	✓	-	-	-	-
OR Count	-	-	-	-	-	-
PA Count	✓	✓	-	-	-	-
PS Count	-	-	-	-	-	-
QAS Count	-	-	-	-	-	-
RCI Count	-	-	-	-	-	-
RE Count	-	-	-	✓	✓	-
SVP Count	-	-	-	-	-	-
Kernel	RBF	Linear	RBF	RBF	Linear	Linear

Table 3: Feature Selection for all the submitted systems.

5 Results

Table 4 shows our system performance based on the test sets. Our best system in Task A ranked 3^{rd} in Italian (0.839 in accuracy for run3) and 13^{th} in English (0.621 in accuracy for run3). Interestingly, our best result on both languages were obtained by the best configuration submitted at the IberEval campaign. However, our English system performance was way worse compared to the result of IberEval (accuracy = 0.814). We will try to analyze this problem in the Section 6.

ITALIAN

Rank	Team	Accuracy
1	bakarov.c.run2	0.844
2	bakarov.c.run1	0.842
3	14-exlab.c.run3	0.839
4	bakarov.c.run3	0.836
5	14-exlab.c.run2	0.835
6	StopPropagHate.c.run1	0.835
7	*AMI-BASELINE*	0.830
8	StopPropagHate.u.run2	0.829
9	SB.c.run1	0.824
10	RCLN.c.run1	0.824
11	SB.c.run3	0.823
12	SB.c.run	0.822

ENGLISH

Rank	Team	Accuracy
1	hateminers.c.run1	0.704
2	hateminers.c.run3	0.681
3	hateminers.c.run2	0.673
4	resham.c.run3	0.651
5	bakarov.c.run3	0.649
6	resham.c.run1	0.648
7	resham.c.run2	0.647
8	ITT.c.run2.tsv	0.638
9	ITT.c.run1.tsv	0.636
10	ITT.c.run3.tsv	0.636
11	himani.c.run2.tsv	0.628
12	bakarov.c.run2	0.628
13	14-exlab.c.run3	0.621
14	himani.c.run1.tsv	0.619
15	himani.c.run3.tsv	0.614
16	14-exlab.c.run1	0.614
17	SB.c.run2.tsv	0.613
18	bakarov.c.run1	0.605
19	*AMI-BASELINE*	0.605
20	StopPropagHate.c.run1.tsv	0.593
21	SB.c.run1.tsv	0.592
22	StopPropagHate.u.run3.tsv	0.591
23	StopPropagHate.u.run2.tsv	0.590
24	RCLN.c.run1	0.586
25	SB.c.run3.tsv	0.584
26	14-exlab.c.run2	0.500

Table 4: Official Results for Subtask A.

In Task B, most of the submitted systems struggled to classify the misogynous tweets into the five categories and discriminate whether the target is active or passive. Both subtasks for both languages have very low baselines (below 0.4 for English and around 0.5 for Italian). Several under-represented classes such as DERAILING and DOMINANCE are very difficult to be detected in category classification (See Table 1 for details). Similarly, the label distribution was very unbalanced for target classification, where most of the misogynous tweets are attacking a specific target (ACTIVE).

ITALIAN

Rank	Team	Avg.	Cat.	Targ.
1	bakarov.c.run1	0.493	0.555	0.432
2	*AMI-BASELINE*	0.487	0.534	0.440
3	14-exlab.c.run3	0.485	0.552	0.418
4	14-exlab.c.run2	0.482	0.550	0.415
5	bakarov.c.run3	0.478	0.536	0.421
6	bakarov.c.run2	0.463	0.499	0.426
7	SB.c.run.tsv	0.449	0.485	0.414
8	SB.c.run1.tsv	0.448	0.483	0.414
9	RCLN.c.run1	0.448	0.473	0.422
10	SB.c.run2.tsv	0.446	0.480	0.411
11	14-exlab.c.run1	0.292	0.164	0.420

ENGLISH

Rank	Team	Avg.	Cat.	Targ.
1	himani.c.run3.tsv	0.406	0.361	0.451
2	himani.c.run2.tsv	0.377	0.323	0.431
3	*AMI-BASELINE*	0.370	0.342	0.399
4	hateminers.c.run3	0.369	0.302	0.435
5	hateminers.c.run1	0.348	0.264	0.431
6	SB.c.run2.tsv	0.344	0.282	0.407
7	himani.c.run1.tsv	0.342	0.280	0.403
8	SB.c.run1.tsv	0.335	0.282	0.389
9	hateminers.c.run2	0.329	0.229	0.430
10	SB.c.run3.tsv	0.328	0.269	0.387
11	resham.c.run2	0.322	0.246	0.399
12	resham.c.run1	0.316	0.235	0.397
13	bakarov.c.run1	0.309	0.260	0.357
14	resham.c.run3	0.283	0.214	0.353
15	RCLN.c.run1	0.280	0.165	0.395
16	ITT.c.run2.tsv	0.276	0.173	0.379
17	bakarov.c.run2	0.275	0.176	0.374
18	14-exlab.c.run1	0.260	0.124	0.395
19	bakarov.c.run3	0.254	0.151	0.356
20	14-exlab.c.run3	0.239	0.107	0.371
21	ITT.c.run1.tsv	0.238	0.140	0.335
22	ITT.c.run3.tsv	0.237	0.138	0.335
23	14-exlab.c.run2	0.232	0.205	0.258

Table 5: Official Results for Subtask B.

Several features which focus on the use of offensive words were proven to be useful in English. For Italian, a simple tweet representation which involves Bag of Words, Bag of Hashtags, and Bag of Emojis already produced a better result than the baseline. Some of the HurtLex categories that were improving the system's performance during training did not help the prediction on the test set (ASF, OM, CDS, DDP, AN, IS, PA for English and CDS, ASM for Italian). However, similarly to the Spanish case, the system configuration which utilized ASF, PR, and DDP obtained the best result in Italian.

6 Discussion

We performed an error analysis on the gold standard test set, and analyzed 160 Italian tweets that our best system configuration mislabelled. The label "misogynistic" was wrongly assigned to 147 instances (false positives, 91.9% of the errors), while the contrary happened only 13 times (false negatives, 8.1% of the errors). The same situation happened in the English dataset, but with a less striking impact, with 228 false positives (60.2% of the errors), 151 false negatives (39.8% of the errors). In this section we conduct a qualitative error analysis, identifying and discussing several factors that contribute to the misclassification.

Presence of swear words. We encountered a lot of "bad words" in the dataset of this shared task for both English and Italian. In case of abusive context, the presence of swear words can help to spot abusive content such as misogyny. However, they could also lead to false positives when the swear word is used in a casual, not offensive context (Malmasi and Zampieri, 2018; Van Hee et al., 2018; Nobata et al., 2016). Consider the following two examples containing the swear word "bitch" in different contexts:

> 1. Im such a fucking cunt bitch and i dont even mean to be goddammit
>
> 2. Bitch you aint the only one who hate me, join the club, stand in the corner, and stfu.

In Example 1, the swear word "bitch" is used just to arouse interest/show off, thus not directly insulting the other person. This is a case of *idiomatic swearing* (Pinker, 2007). In Example 2, the swear word "bitch" is used to insult a specific target in an abusive context, an instance of *abusive swearing* (Pinker, 2007). Resolving swearing context is still a challenging task for automatic system which contributing to the difficulties of this task.

Reported speech. Tweets may contain misogynistic content as an indirect quote of someone else's words, such as in the following example:

> 3. Quella volta che mia madre mi ha detto quella cosa le ho risposto "Mannaggia! Non sarò mai una brava donna schiava zitta e lava! E adesso?!" Potrei morire per il dispiacere.
> → *That time when my mom told me that thing and I answered "Holy s**t! I will never be a good slave who shuts up and cleans! What now?"*

According to task guidelines this should not be labeled as a misogynistic tweet, because it is not the user himself who is misogynistic. Therefore, instances of this type tend to confuse a classifier based on lexical features.

Irony and world knowledge. In Example 3, the sentence "Potrei morire per il dispiacere."[6] is ironic. Humor is very hard to model for automatic systems — sometimes, the presence of figurative language even baffles human annotators. Moreover, external world knowledge is often required in order to infer whether an utterance is ironic (Wallace et al., 2014).

Preprocessing and tokenization. In computer-mediated communication, and specifically on Twitter, users often resort to a language type that is closer to speech, rather than written language. This is reflected in less-than-clean orthography, with forms and expressions that imitate the verbal face-to-face conversation.

> 4. @ @ @ @ x me glob prox2aa colpiran tutti incluso nemicinterno.. esterno colpopiúduro saràculograndе che bevetropvodka e inoltre x questiondisoldi progetta farmezzofallirsudfinitestampe: ciò nnvàben xrchèindebolis
> → *4 me glob next2aa will hit everyone included internalenemy.. external harderhit willbebigass who drinkstoomuchvodka and also 4 mattersofmoney isplanning tomakethesouthfailwithprintings: dis notgood causeweaken*

In Example 4, preprocessing steps like tokenization and stemming are particularly hard to perform, because of the lack of spaces between one word and the other and the confused orthography. Consequently all the classification pipeline is compromised and error-prone.

Gender of the target. As defined in the Introduction, we know that misogyny is a specific type of hateful language, targeting women. However, detecting the gender of the target is a challenging task in itself, especially in Twitter datasets.

> 5. @realDonaldTrump shut the FUCK up you infected pussy fungus.
>
> 6. @TomiLahren You're a fucking skank!

Both examples use bad words to abuse their targets. However, the first example is labeled as not misogyny since the target is Donald Trump (man), while the second example is labeled as misogyny with the target Tomi Lahren (woman).

[6] Translation: I could die for heartbreak.

7 Conclusions

Here we draw some considerations based on the results of our participation to the EVALITA 2018 AMI shared task. In order to test the multilingual potential of our model, one of the systems we submitted for Italian at EVALITA (run #3) was based on our best model for Spanish at IberEval. Based on the official results, this system performed well for Italian, consisting of features such as: BoW, BoE, BoH and several HurtLex categories specifically related to the hate against women. Concerning English, we obtained lower results in EVALITA in comparison to IberEval with the same system configuration. It is worth mentioning that even if the training set for the AMI EVALITA task was substantially bigger, in absolute terms all the AMI's participants at EVALITA obtained worse scores than the ones obtained by the IberEval's teams.

Acknowledgments

Valerio Basile and Viviana Patti were partially supported by Progetto di Ateneo/CSP 2016 (*Immigrants, Hate and Prejudice in Social Media-IhatePrejudice*, S1618_L2_BOSC_01).

References

Maria Anzovino, Elisabetta Fersini, and Paolo Rosso. 2018. Automatic Identification and Classification of Misogynistic Language on Twitter. In *Proc. of the 23rd Int. Conf. on Applications of Natural Language & Information Systems*, pages 57–64. Springer.

Jamie Bartlett, Richard Norrie, Sofia Patel, Rebekka Rumpel, and Simon Wibberley. 2014. Misogyny on twitter. *Demos*.

Elisa Bassignana, Valerio Basile, and Viviana Patti. 2018. Hurtlex: A Multilingual Lexicon of Words to Hurt. In *Proc. of the 5th Italian Conference on Computational Linguistics (CLiC-it 2018)*, Turin, Italy. CEUR.org.

Romolo Giovanni Capuano. 2007. *Turpia: sociologia del turpiloquio e della bestemmia*. Riscontri (Milano, Italia). Costa & Nolan.

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine learning*, 20(3):273–297.

Tullio De Mauro. 2016. Le parole per ferire. *Internazionale*. 27 settembre 2016.

Fabio Fasoli, Andrea Carnaghi, and Maria Paola Paladino. 2015. Social acceptability of sexist derogatory and sexist objectifying slurs across contexts. *Language Sciences*, 52:98–107.

Elisabetta Fersini, Maria Anzovino, and Paolo Rosso. 2018a. Overview of the Task on Automatic Misogyny Identification at IberEval. In *Proceedings of 3rd Workshop on Evaluation of Human Language Technologies for Iberian Languages (IberEval 2018)*, pages 57–64. CEUR-WS.org, September.

Elisabetta Fersini, Debora Nozza, and Paolo Rosso. 2018b. Overview of the evalita 2018 task on automatic misogyny identification (ami). In *Proceedings of the 6th evaluation campaign of Natural Language Processing and Speech tools for Italian (EVALITA'18)*, Turin, Italy. CEUR.org.

Thorsten Joachims. 1998. Text categorization with support vector machines: Learning with many relevant features. In *European conference on machine learning*, pages 137–142. Springer.

Shervin Malmasi and Marcos Zampieri. 2018. Challenges in discriminating profanity from hate speech. *Journal of Experimental & Theoretical Artificial Intelligence*, 30(2):187–202.

Chikashi Nobata, Joel Tetreault, Achint Thomas, Yashar Mehdad, and Yi Chang. 2016. Abusive language detection in online user content. In *Proceedings of the 25th international conference on world wide web*, pages 145–153.

Endang Wahyu Pamungkas, Alessandra Teresa Cignarella, Valerio Basile, and Viviana Patti. 2018. 14-ExLab@ UniTo for AMI at IberEval2018: Exploiting Lexical Knowledge for Detecting Misogyny in English and Spanish Tweets. In *Proc. of 3rd Workshop on Evaluation of Human Language Technologies for Iberian Languages (IberEval 2018)*.

Steven Pinker. 2007. *The stuff of thought: Language as a window into human nature*. Penguin.

Bailey Poland. 2016. *Haters: Harassment, Abuse, and Violence Online*. Potomac Press.

Anna Schmidt and Michael Wiegand. 2017. A survey on hate speech detection using natural language processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media*, pages 1–10.

Cynthia Van Hee, Gilles Jacobs, Chris Emmery, Bart Desmet, Els Lefever, Ben Verhoeven, Guy De Pauw, Walter Daelemans, and Véronique Hoste. 2018. Automatic detection of cyberbullying in social media text. *arXiv preprint arXiv:1801.05617*.

Byron C. Wallace, Laura Kertz, Eugene Charniak, et al. 2014. Humans require context to infer ironic intent (so computers probably do, too). In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 512–516.

CrotoneMilano for AMI at Evalita2018. A performant, cross-lingual misogyny detection system.

Angelo Basile
Symanto Research
angelo.basile@symanto.net

Chiara Rubagotti
Independent Researcher
chiara.rubagotti@gmail.com

Abstract

We present our systems for misogyny identification on Twitter, for Italian and English. The models are based on a Support Vector Machine and they use n-grams as features. Our solution is very simple and yet we achieve top results on Italian Tweets and excellent results on English Tweets. Furthermore, we experiment with a single model that works across languages by leveraging abstract features. We show that a single multilingual system yields performances comparable to two independently trained systems. We achieve accuracy results ranging from 45% to 85%. Our system is ranked first out of twelve submissions for sub-task B on Italian and second for sub-task A.

In questo articolo presentiamo i nostri modelli per il riconoscimento automatico di testi misogini su Twitter: abbiamo addestrato lo stesso sistema prima su un corpus italiano e poi su uno inglese. Il modello si basa su una macchina a vettori supporto e usa n-grammi come feature. La nostra soluzione è molto semplice e tuttavia ci permette di raggiungere lo stato dell'arte sull'italiano e ottimi risultati sull'inglese. Presentiamo inoltre un sistema che funziona con entrambe le lingue sfruttando una serie di feature astratte. Il nostro livello raggiunge livelli di accuratezza tra il 45% e l'85%: con questi risultati ci piazziamo primi nel task B per l'italiano e secondi nel task A.

1 Introduction

With awareness of violence against women growing in the public discourse and the spread of unfiltered and possibly anonymous communication on social media in our digital culture, the issue of misogyny online has become compelling. Violence against women has been described by the UN as a "Gender-based [..] form of discrimination that seriously inhibits women's ability to enjoy rights and freedoms on a basis of equality with men"[1]. On the web this often takes the form of female-discriminating attacks of different types, which undermine the women's rights of freedom of expression and participation[2]. Following erjavec2012you's understanding of *hate speech*, reported in (Pamungkas et al., 2018) as "any type of communication that is abusive, insulting, intimidating, harassing, and/or incites to violence or discrimination, and that disparages a person or a group on the basis of some characteristics such as race, colour, ethnicity, gender, sexual orientation, nationality, religion, or other characteristics", we can define *misogynist speech* as any kind of aggressive discourse which targets women because they are women. Within the larger context of hate speech, online misogyny — or *cybersexism* — stands out as a large and complex phenomenon which reflects other forms of offline abuse on women (Poland, 2016). This holds true for the Italian case as well, where bouts of misogynistic tweets have been linked to episodes of femicides [3]. In recent years the NLP commu-

[1] http://www.un.org/womenwatch/daw/cedaw/recommendations/recomm.htm

[2] https://www.amnesty.org/en/latest/research/2018/03/online-violence-against-women-chapter-3/.

[3] http://www.voxdiritti.it/wp-content/uploads//2018/06/mappa-intolleranza-3-donne.jpg

nity has addressed the issue of automatic detection of hate speech in general (Schmidt and Wiegand, 2017) and misogyny in particular (Anzovino et al., 2018). This effort to detect and contain verbal violence on social media (or any kind of text) demonstrates how NLP tools can also be used for ethically beneficial purposes and should be considered in the newborn and crucial discourse on Ethics in NLP (Hovy and Spruit, 2016; Hovy et al., 2017; Alfano et al., 2018). We are therefore proud to take up the AMI challenge (Fersini et al., 2018) and present our contribution to the cause of stopping misogynistic speech on Twitter. In this paper we propose a simple linear model using n-grams: we show that such a simple setup can still yield good results. We decided to propose a simple model for three reasons: first, it has been shown that linear SVM can easily outperform more complex deep neural networks (Plank, 2017; Medvedeva et al., 2017); second, training and testing our model does not require expensive hardware but a common laptop is enough to replicate our experiments; third, we experiment with a transformation of the input (i.e. we extract abstract features) and a linear model allows for an easier interpretation of the contribution of this transformation.

To summarise, the following are the contributions of this paper:

- We propose a simple and yet strong misogyny detection system for English and Italian (ranked first out of twelve systems for misogynistic category detection)
- We show how a single system can be trained to work across languages
- We release all the code[4] and our trained systems for reproducibility and for a quick implementation of language technology systems that can help detect and mitigate cybersexism phenomena.

Task Description The AMI task is combined binary and multi-label, short text classification task. Given a Tweet, we have to predict whether it contains or not misogyny (**Task A**) and if it does, we have to classify the misogynistic behaviour and predict who is the subject being targeted (**Task B**). The misogynistic behaviour's space consists of five different labels:

- `Stereotype & Objectification`
- `Dominance`
- `Derailing`
- `Sexual Harassment & Threats of Violence`
- `Discredit`

The target can be either `Active` when the message refers to a specific person or `Passive` when the message expresses generic misogyny. The setup is the same for both Italian and English.

2 Data

We use only the data released by the task organisers: they consist of Italian and English Tweets. The organisers report that the corpus has been manually labelled by several annotators. We provide an overview of the data set in Table 1. As it can be seen from the table, the data for Task A is more or less balanced, while the data for Task B is highly skewed.

3 Experiments

In this section we describe the feature extraction process and the model that we built.

3.1 Pre-processing

We decide not to pre-process the data in any way, since we do not have linguistic (or non-linguistic) reasons for doing so. To tokenize the text we simply split at every white space.

3.2 Model and Features

We built a sparse linear model for approaching this task.

We use n-grams extracted at the word level as well as at the character level. We use 3-10 n-grams and binary tf-idf. We feed these features to a Support Vector Machine (SVM) model with a linear kernel; we use the implementation included in `scikit-learn` (Pedregosa et al., 2011). Furthermore, we experiment with feature abstraction: we follow the bleaching approach recently proposed by (van der Goot et al., 2018). First, we transform each word in a list of symbols that 1) represents the shape of the individual characters

[4]The code can be found at https://github.com/anbasile/AMI/.

	ITALIAN					ENGLISH				
	SO	DO	DE	ST	DI	SO	DO	DE	ST	DI
Active	625	61	21	428	586	54	78	24	207	695
Passive	40	9	3	2	43	125	70	68	145	319
Non-Misogynous	2172					2215				

Table 1: Data Set Overview, showing the label distribution across the five misogynistic behaviours: Stereotype & Objectification (**SO**), Dominance (**DO**), Derailing (**DE**), Sexual Harassment & Threats of Violence (**ST**) and Discredit (**DI**).

	SHAPE	FREQ	LEN	ALPHA
This	Ccvc	46	04	True
is	vc	650	02	True
an	vc	116	02	True
example	vcvcccv	1	07	True
.	.	60	01	False
😎	😎	1	01	False

Table 2: An illustration of the bleaching process.

and 2) abstracts from meaning by still approximating the vowels and characters that compose the word; then, we compute the length of the word and its frequency (while taking care of padding the first one with a zero in order to avoid feature collision); finally, we use a Boolean label for explicitly distinguishing words from non-alphanumeric token (e.g. emojis). Table 2 shows an example of this feature abstraction process.

(van der Goot et al., 2018) proposed this bleaching approach for modelling gender across languages, by leveraging the language-independent nature of these features: here, we try to re-use the technique for classifying misogynist text across languages. We slightly modify the representation proposed by (van der Goot et al., 2018) by merging the shape feature (e.g. `Xxx`) with the vowel-consonant approximation feature (e.g. `CVC`) into one single feature (e.g. `Cvc`).

We propose three different multi-lingual experiments:

- `TRAIN` Italian → `TEST` English
- `TRAIN` English → `TEST` Italian
- `TRAIN` Ita. & Eng. → `TEST` Ita. & Eng.

For the last experiment, we use half the data set for each language. We report scores obtained by training on the whole training set and testing on the official test set, using the gold labels released by the organisers after the evaluation period.

4 Evaluation and Results

Since the data set labels for the sub-task B are not evenly distributed across the classes, we use f1-score to evaluate our model. First we report results obtained via a 10-fold cross-validation on the training set; then, we report results from the official test set, whose labels have been released. The official evaluation does not take into account the joint prediction of the labels, however here we report results considering the 0 label: since we train different models for the different label sets, we make sure that the models trained on Task B are able to detect if a message is misogynistic in the first place.

4.1 Development Results

We report the development results obtained by using different text representations. Table 3 presents an overview of these results. We note that all four representations — words, characters, a combination of these two and the bleached representation — all yield comparable results. The combination of words and characters seems to be the best format. Overall, we note that the system performs better on the Italian corpus than on the English corpus.

4.1.1 Cross-lingual Results

In Table 4 we present the results of our cross-lingual experiments. We train and test different systems using lexical and abstract features. We note that the abstract model trained on Italian outperforms the fully lexicalized model when tested on English, but the opposite is not true. The English data set seems particularly hard for both the abstract and the lexicalized model. Interestingly, the abstract model trained on both corpora shows good results.

	ENGLISH			ITALIAN		
	MIS.	CAT.	TGT.	MIS.	CAT.	TGT.
Words (W)	0.68	0.29	0.57	0.88	0.60	0.59
Chars (C)	0.71	0.30	0.61	0.88	0.59	0.58
W+C	0.70	0.31	0.59	0.88	0.62	0.59
Bleaching	0.68	0.27	0.57	0.85	0.55	0.56

Table 3: An overview of the development f1-macro scores obtained via cross-validation.

	TEST →	IT	EN
TRAIN	IT lex	0.85	0.51
	IT abs	0.83	0.52
	EN lex	0.47	0.62
	EN abs	0.45	0.52
	IT + EN lex	0.83	0.60
	IT + EN abs	0.81	0.58

Table 4: Pair-wise accuracy results for Task A. We compare lexicalized vs. abstract models. The combined IT+EN data set is built by randomly sampling 50% of instances from both corpora.

4.2 Test Results

In Table 5 we present official test results (Fersini et al., 2018). We submitted only one, constrained run; a run is considered *constrained* when only the data released by the organisers are used. We submitted the model using the combined representation with word- and character-ngrams, trained once on the English corpus and once on the Italian corpus. We achieve the top and the second position for the tasks B and A respectively on the Italian data set. On the English data set our system is ranked 15th and 4th on the tasks A and B respectively.

	TASK A	TASK B		
		CATEGORY	TARGET	AVG.
IT	0.843	0.579	0.423	0.501
EN	0.617	0.293	0.444	0.369

Table 5: Official test results. Task A is measured using accuracy and Task B is measured using f1-score. We reach the first position on Task B for Italian.

5 Discussion and Conclusions

A warning to the reader: this section contains explicit language. In the attempt to understand better the big difference in performance between the English and Italian models, we show the importance of words as learned by the model: we print the ten most important words, ranked by their learned weights. The result is shown in Table 6. From the output we see that the model trained on Italian learned meaningful words for identifying a misogynist message, such as *zitta* [shut up], *tua* [your] and *muori* [die!]: these words stand out from the rest of the profanity for directly referring to someone, while the rest of the words and almost all the most important English words could be used as interjections or could be more generic insults.

RANK	ITA	ENG
1	zitta	woman
2	bel	hoe
3	pompinara	she
4	puttanona	hoes
5	tua	women
6	muori	whore
7	baldracca	her
8	troie	bitches
9	culona	womensuck
10	tettona	bitch

Table 6: Top ten words ranked by their positive weights learned during training.

The results of the abstract system are satisfactory for eventually building a light, portable model that could be adapted to different language. In the future we will try training on English and Italian and testing on a third corpus (such as the Spanish version of the AMI data set).

In this paper we described our participation to the AMI - Automatic Misogyny Identification for Italian and English. We proposed a very simple solution that can be implemented quickly and we scored a state-of-the-art result for classification of misogynistic behaviours in five classes.

Acknowledgements

The authors would like to thank the two anonymous reviewers who helped improve the quality of this paper. The first author has conducted this research as he was still part of the Erasmus Mundus master in Language and Communication Technology, a shared master program between the University of Groningen (NL) and the University of Malta (MT).

References

Mark Alfano, Dirk Hovy, Margaret Mitchell, and Michael Strube. 2018. Proceedings of the second acl workshop on ethics in natural language processing. In *Proceedings of the Second ACL Workshop on Ethics in Natural Language Processing*.

Maria Anzovino, Elisabetta Fersini, and Paolo Rosso. 2018. Automatic identification and classification of misogynistic language on twitter. In *International Conference on Applications of Natural Language to Information Systems*, pages 57–64. Springer.

Elisabetta Fersini, Debora Nozza, and Paolo Rosso. 2018. Overview of the evalita 2018 task on automatic misogyny identification (ami). In Tommaso Caselli, Nicole Novielli, Viviana Patti, and Paolo Rosso, editors, *Proceedings of the 6th evaluation campaign of Natural Language Processing and Speech tools for Italian (EVALITA'18)*, Turin, Italy. CEUR.org.

Dirk Hovy and Shannon L Spruit. 2016. The social impact of natural language processing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 591–598.

Dirk Hovy, Shannon Spruit, Margaret Mitchell, Emily M Bender, Michael Strube, and Hanna Wallach. 2017. Proceedings of the first acl workshop on ethics in natural language processing. In *Proceedings of the First ACL Workshop on Ethics in Natural Language Processing*.

Maria Medvedeva, Martin Kroon, and Barbara Plank. 2017. When sparse traditional models outperform dense neural networks: the curious case of discriminating between similar languages. In *Proceedings of the Fourth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial)*, pages 156–163.

Endang Wahyu Pamungkas, Alessandra Teresa Cignarella, Valerio Basile, and Viviana Patti. 2018. 14-exlab@unito for ami at ibereval2018: Exploiting lexical knowledge for detecting misogyny in english and spanish tweets. In *Proceedings of the Third Workshop on Evaluation of Human Language Technologies for Iberian Languages (IberEval 2018)*, pages 234–241.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Barbara Plank. 2017. All-in-1 at ijcnlp-2017 task 4: Short text classification with one model for all languages. *Proceedings of the IJCNLP 2017, Shared Tasks*, pages 143–148.

Bailey Poland. 2016. *Haters: Harassment, Abuse, and Violence Online*. University of Nebraska Press.

Anna Schmidt and Michael Wiegand. 2017. A survey on hate speech detection using natural language processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media*, pages 1–10.

Rob van der Goot, Nikola Ljubešić, Ian Matroos, Malvina Nissim, and Barbara Plank. 2018. Bleaching text: Abstract features for cross-lingual gender prediction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 383–389.

Vector Space Models for Automatic Misogyny Identification

Amir Bakarov
National Research University Higher School of Economics, Moscow, Russia
amirbakarov at gmail.com

Abstract

English. The problem of hate speech and, especially, of misogynous language is one of the most crucial problems of contemporary Internet communities. Therefore, automatic detection of such language becomes one of the most actual natural language processing tasks. The most ubiquitous tools for resolving this task are based on vector space models of texts. In this paper we describe our system that exploits such tools and have shown the best performance on the Italian AMI task of EVALITA 2018.

Italiano. *Il problema dell'uso di discorsi che incitano l'odio, e specialmente dell'uso di linguaggio misogino, è uno dei problemi più cruciali delle comunità di internet al giorno d'oggi. Pertanto, il rilevamento automatico di tale linguaggio diventa uno degli obiettivi più attuali per l'elaborazione del linguaggio naturale. I sistemi più diffusi atti ad affrontare questo obiettivo sfruttano l'ipotesi distributiva. In questo articolo, descriviamo il sistema proposto basato su quest'ipotesi che hanno dimostrato le migliori performance nel task AMI di EVALITA 2018 nella lingua italiana.*

1 Introduction

As the Internet community and several online discussions grow, the number of manifestations of *hate speech* on open web resources also increases. Such type of speech (also called *abusive language* or *textual harassment*) could get different forms depending on its focus on the person's ethnicity, gender identity, religion, or sexual orientation. Probably, one of the most destructive forms of hate speech is the one that abuses a person's gender identity. Such form of hate speech is called *misogynous language* since misogyny is a specific case of hate whose targets are women. Misogyny on the Internet (*cybermisogyny*, or *online sexual harassment*) is one of the crucial problems of contemporary Internet communities, especially from the perspective of the societal impact of this phenomenon.

Thus, the problem of automatic misogyny identification could be considered as one of the most important branches of a hate speech detection task. The successful solution of this problem could lead to the significant limitation of the diffusion for the hate speech against women. The problem of automatic misogynous language detection got attention from the research community fairly recently, and the shared task on *automatic misogyny identification* held as a part of the EVALITA-2018 campaign is one of the first works trying to deal with this problem (Fersini et al., 2018b). The aim of this task is to automatically identify misogynous content in tweets for the Italian and English languages.

This paper describes our system that has outperformed all other systems for the Italian language and also has shown fairly good results for the English language. This system is based on using semantic features of tweets as an input of a supervised classifier. The semantic features are considered as latent vectors produced by a vector space model.

Our work is organized as follows. Section 2 briefly describes related work on the proposed task. Section 3 describes the setup of our system, while Section 4 discusses the results and proposes an analysis of them. Section 5 concludes the paper.

	Task A (Italian)	Task B (Italian)	Task A (English)	Task B (English)
Baseline	0.830	0.487	0.605	0.370
TFIDF+LR	0.842	0.443	**0.649**	0.241
TFIDF+XGB	0.836	**0.493**	0.604	**0.309**
TFIDF+SVD+LR	**0.844**	0.478	0.628	0.275
TFIDF+SVD+XGB	0.833	0.463	0.605	0.254

Table 1: Performance of each of the compared vectorizers and supervised classifiers on each of the tasks. Task A reports accuracy, Task B reports macro F1-measure.

2 Related Work

The first notorious works of the task of automatic misogyny identification were described as shared task proposed at IverEval 2018 workshop (Fersini et al., 2018a) (a shared task organized jointly with SEPLN-2018 Conference for Iberian languages), and SemEval-2019[1]. These tasks proposed certain baselines based on ubiquitous text classification techniques (for example, SVM). The automatic misogyny identification task considered in our research is the third shared task on this topic (Anzovino et al., 2018). We are also aware of certain other attempts to computationally resolve the task of automatic misogyny identification, but most of them were published only as some exploratory analysis (Hewitt et al., 2016). Most of the state-of-the art approaches to this problem were described as system reports for the aforementioned IberEval-2018 shared task. As far as we know, there were no other scholarly works trying to resolve or to formalize this task.

In the natural language processing community very similar tasks were also considered in other hate speech online challenges and scholarly works (Davidson et al., 2017). An extensive overview of all the research related to hate speech detection goes beyond the scope of this work, and an interested reader could be referred to a survey paper specialized on this topic (Schmidt and Wiegand, 2017).

Apart from computational linguistics and natural language processing, the problem of misogynous speech was also a focus of some linguistic and social science articles (Fulper et al., 2014). Most of such scholarly works were trying to understand the nature of misogynous hate speech and patterns appearing in this type of language (Poland, 2016). We think that from the perspective of natural language processing, such papers could be useful for the systems that are highly grounded to linguistic knowledge and manually crafted resources.

3 Experimental Setup

In the shared task we had two datasets (for English and for Italian) of 5000 tweets each. 4000 tweets in each dataset were considered as a training sample, and the evaluation of the system was done on 1000 tweets (their labels were hidden until the end of the competition). The classification task has included both binary and multi-label classification.

In our work we have used vectors from term-document matrix with TF-IDF values. We propose the text classification based on using semantic features obtained from vector space models of texts. We considered the terms as word n-grams, and used a factorization of the term-document matrix (we used a method of singular value decomposition, SVD) and a normalization of factorized values (in the table with the results we call it **TFIDF+SVD**). From this perspective, our approach is very close to the method of Latent Semantic Analysis (Landauer et al., 1998) (and we have also tried to resolve this task using not-factorized TF-IDF matrix, called **TFIDF** in the table). As a supervised classifier we have used a Logistic Regression classifier, therefore, our system is based on using TF-IDF n-gram word features and a Logistic Regression (LR).

For all the methods of vectorization we used a basic pipeline of text pre-processing (tokenization, lemmatization and stop-word removal based on NLTK build-in tools and resources).

We have also compared it with other classifiers (for instance, a Gradient Boosting classifier,

[1] https://competitions.codalab.org/competitions/19935

XGB in the table) and got worse results on the certain tasks. All in all, we have compared four different models. The exact hyperparameters of the models used in our system, and all the code for reproducing the experiments could be found at our Gitlab repository: `https://gitlab.com/bakarov/ami-evalita`.

4 Results and Discussion

The system evaluation was done on two subtasks. The first subtask had proposed a binary classification to identify whether the text is either misogynous or not misogynous (Task A). The second subtask (Task B) was to classify the misogynous tweets according to both the misogynistic behavior (multi-label classification) and the target of the message (binary-classification). The results of the system for the English and Italian subtasks for the misogyny identification task are described in Table 1. It is notable that our system has outperformed the baseline put by organizers in most of the cases, and different combinations of vectorizers and models have shown different performance in different tasks.

After an error analysis conducted on the system, we have found out that the system fails on examples where misogyny is expressed without (or with a very little use of) offensive lexis, or, vice versa, such lexis is used not in misogynous context (for example, *you pussy boy*). This could be explained by the fact that the system is too much focused on the lexicon and does not takes into account syntactic patterns or thematic roles.

5 Conclusions

The proposed work has described the system that has shown the best results for the Italian track on all the subtasks (and have also got fairly good results on English). Our system is based on a vector space model of character n-grams and a supervised gradient boosting classifier.

The system described in this paper is one of the first attempts to the problem of detecting misogynistic language for the Italian language in the natural language processing community. We think that the description of the implementation of our system could help other researchers to resolve such important and actual task. We consider this value as a main contribution of our research.

In future we plan to give more attention to some other linguistic features based on analysis of patterns that people tend to use in misogynous language. We would also like to try out more promising approaches to text classification based on deep learning (for example, convolutional neural networks).

References

Anzovino, M., Fersini, E., and Rosso, P. (2018). Automatic identification and classification of misogynistic language on twitter. In *International Conference on Applications of Natural Language to Information Systems*, pages 57–64. Springer.

Davidson, T., Warmsley, D., Macy, M., and Weber, I. (2017). Automated hate speech detection and the problem of offensive language. *arXiv preprint arXiv:1703.04009*.

Fersini, E., Anzovino, M., and Rosso, P. (2018a). Overview of the task on automatic misogyny identification at ibereval. In *Proceedings of the Third Workshop on Evaluation of Human Language Technologies for Iberian Languages (IberEval 2018), co-located with 34th Conference of the Spanish Society for Natural Language Processing (SEPLN 2018). CEUR Workshop Proceedings. CEUR-WS. org, Seville, Spain*.

Fersini, E., Nozza, D., and Rosso, P. (2018b). Overview of the evalita 2018 task on automatic misogyny identification (ami). In Caselli, T., Novielli, N., Patti, V., and Rosso, P., editors, *Proceedings of the 6th evaluation campaign of Natural Language Processing and Speech tools for Italian (EVALITA'18)*, Turin, Italy. CEUR.org.

Fulper, R., Ciampaglia, G. L., Ferrara, E., Ahn, Y., Flammini, A., Menczer, F., Lewis, B., and Rowe, K. (2014). Misogynistic language on twitter and sexual violence. In *Proceedings of the ACM Web Science Workshop on Computational Approaches to Social Modeling (ChASM)*.

Hewitt, S., Tiropanis, T., and Bokhove, C. (2016). The problem of identifying misogynist language on twitter (and other online social spaces). In *Proceedings of the 8th ACM Conference on Web Science*, pages 333–335. ACM.

Landauer, T. K., Foltz, P. W., and Laham, D. (1998). An introduction to latent semantic analysis. *Discourse processes*, 25(2-3):259–284.

Poland, B. (2016). *Haters: Harassment, abuse, and violence online*. U of Nebraska Press.

Schmidt, A. and Wiegand, M. (2017). A survey on hate speech detection using natural language processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media*, pages 1–10.

Tweetaneuse @ AMI EVALITA2018: Character-based Models for the Automatic Misogyny Identification Task

Davide Buscaldi
LIPN, Université Paris 13
Sorbonne Paris Cité
99, Avenue Jean-Baptiste Clément
93430 Villetaneuse (France)
buscaldi@lipn.fr

Abstract

English. This paper presents the participation of the RCLN team with the Tweetaneuse system to the AMI task at Evalita 2018. Our participation was focused on the use of language-independent, character-based methods.

Italiano. *Quest'articolo presenta la partecipazione del team RCLN con il sistema Tweetaneuse al challenge AMI di Evalita 2018. La nostra partecipazione era orientata sull'utilizzo di metodi multilingue e basati sui caratteri.*

1 Introduction

The language used on social media and especially Twitter is particularly noisy. The reasons are various; among them, the abuse of abbreviations induced by the limitations on the size of the messages, and the use of different ways to refer to the same event or concept, strengthened by the availability of hashtags (for instance: *World Cup in Russia*, *#WorldCup2018*, *#WC18* all refer to the same event).

Recently, some character-level neural network based models have been developed to take into account these problems for tasks such as sentiment analysis (Zhang et al., 2017) or other classification tasks (Yang et al., 2016). Another advantage of these methods, apart the robustness to the noisy text that can be found in tweets, is that they are completely language independent and they don't need lexical information to carry out the classification task.

The Automatic Misogyny Identification task at Evalita2018 (Fersini et al., 2018) presented an interesting and novel challenge. Misogyny is a type of hate speech that targets specifically women in different ways. The language used in such messages is characterised by the use of profanities, specific hashtags, threats and other intimidating language. This task is an ideal test bed for character-based models, and (Anzovino et al., 2018) already reported that character n-grams play an important role in the misogyny identification task.

We participated to the French Sentiment Analysis challenge DEFT 2018 (Paroubek et al., 2018) earlier this year with language-independent character-based models, based both on neural networks and classic machine learning algorithms. For our participation to AMI@Evalita2018 our objective was to verify whether the same models could be applied to this task while keeping a comparable accuracy.

The rest of the paper is structured as follows: in Section 2 we describe the two methods that were developed for the challenge; in Section 3 we present and discuss the obtained results, and finally in Section 4 we draw some conclusions about our experience and participation to the AMI challenge.

2 Methods

2.1 Locally-weighted Bag-of-Ngrams

This method is based on a Random Forest (RF) classifier (Breiman, 2001) with character n-grams features, scored on the basis of their relative position in the tweet. One of the first parameters to choose was the size of the n-grams to work with. According to our previous experience, we chose to use all the character n-grams (excluding spaces) of size 3 to 6, with a minimum frequency of 5 in the training corpus.

The weight of each n-gram n in tweet t is calculated as:

$$s(n,t) = \begin{cases} 0 & if absent \\ \sum_{i=1}^{occ(n,t)} 1 + \frac{pos(n_i)}{len(t)} & otherwise \end{cases}$$

where $occ(n,t)$ is the number of occurences of n-gram n in t, $pos(n_i)$ indicates the position of the first character of the i-th occurrence of the n-gram n and $len(t)$ is the length of the tweet as number of characters. The hypothesis behind the use of this positional scoring scheme is that the presence of some words (or symbols) at the end or the beginning of a tweet may be more important that the mere presence of the symbol. For instance, in some cases the conclusion is more important than the first part of the sentence, especially when people are evaluating different aspects of an item or they have mixed feelings: *I liked the screen, but the battery duration is horrible*.

2.2 Char and Word-level bi-LSTM

This method was only tested before and after the participation, since we observed that it performed worse than the Random Forest method.

In this method we use a recurrent neural network to implement a LSTM classifier (Hochreiter and Schmidhuber, 1997), which are now widely used in Natural Language Processing. The classification is carried out in three steps:

First, the text is split on spaces. Every resulting text fragment is read as a character sequence, first from left to right, then from right to left, by two recurrent NN at character level. The vectors obtained after the training phase are summed up to provide a character-based representation of the fragment (compositional representation). For a character sequence $s = c_1 \dots c_m$, we compute for each position $h_i = LSTM_o(h_{i-1}, e(c_i))$ et $h'_i = LSTM_{o'}(h'_{i+1}, e(c_i))$, where e is the embedding function, and $LSTM$ indicates a LSTM recurrent node. The fragment compositional representation is then $c(s) = h_m + h'_1$.

Subsequently, the sequence of fragments (i.e., the sentence) is read again from left to right and vice versa by other two recurrent NNs at word level. These RNNs take as input the compositional representation obtained in the previous step for the fragments to which a vectorial representation is concatenated. This vectorial representation is obtained from the training corpus and is considered only if the textual fragment has a frequence ≥ 10. For a sequence of textual fragments $p = s_1 \dots s_n$, we calculate $l_i = LSTM_m(l_{i-1}, c(s_i) + e(s_i))$, $l'_i = LSTM_{m'}(l_{i+1}, c(s_i) + e(s_i))$, where c is the compositional representation introduced above and e the embedding function. The final states obtained after the bi-directional reading are added and they are required to represent the input sentence, $r(p) = l_n + l'_1$.

Finally, these vectors are used as input to a multi-layer perceptron which is responsible for the final classification: $o(p) = \sigma(O \times \max(0, (W \times r(p) + b)))$, where σ is the *softmax* operator, W, O are matrices and b a vector. The output is interpreted as a probability distribution on the tweets' categories.

The size of character embeddings is 16, those of the text fragments 32, the input layer for the perceptron is of size 64 and the hidden layer 32. The output layer is size 2 for subtask A and 6 for subtask B. We used the DYNET[1] library.

3 Results

We report in Table 1 the results on the development set for the two methods.

Italian		
Subtask	**lw RF**	**bi-LSTM**
Misogyny identification	0.891	0.872
Behaviour classification	0.692	0.770
English		
Misogyny identification	0.821	0.757
Behaviour classification	0.303	0.575

Table 1: Results on the dev test (macro F1). In both cases, the results were obtained on a random 90%-10% split of the dev dataset.

From these results we could see that the locally weighted n-grams model using Random Forest was better in the identification tasks, while the bi-LSTM was more accurate for the misogynistic behaviour classification sub-task. However, a closer look to these results showed us that the bi-LSTM was classifying all instances but two in the majority class. Finally, due to these problems, we decided to participate to the task with just the locally weighted n-grams model.

The official results obtained by this model are detailed in Table 2. We do not consider the *derailing* category for which the system obtained 0 accuracy.

We also conducted a "post-mortem" test with the bi-LSTM model for which we obtained the following results:

[1] https://github.com/clab/dynet

Overall		
Subtask	**Italian**	**English**
Misogyny identification	0.824	0.586
Behaviour classification	0.473	0.165
Per-class accuracy (sub-task B)		
discredit	**0.694**	0.432
dominance	0.250	0.184
sexual_harassment	0.722	0.169
stereotype	0.699	0.040
active	0.816	0.541
passive	0.028	0.248

Table 2: Official results on the test set (macro F1) obtained by the locally-weighted bag of n-grams model.

Subtask	**Italian**	**English**
Misogyny identification	0.821	0.626
Behaviour classification	0.355	0.141

Table 3: Unofficial results on the test set (macro F1) obtained by the bi-LSTM character model.

As it can be observed, the results confirmed those obtained in the dev test for the misogyny identification sub-task, and in any case we observed that the "deep" model performed overall worse than its more "classical" counterpart.

The results obtained by our system were in general underwhelming and below the expectations, except for the *discredit* category, for which our system was ranked 1st and 3rd in Italian and English respectively. An analysis of the most relevant features according to information gain (Lee and Lee, 2006) showed that the 5 most informative n-grams are *tta*, *utt*, *che*, *tan*, *utta* for Italian and *you*, *the*, *tch*, *itc*, *itch* for English. They are clearly part of some swear words that can appear in different forms, or conjunctions like *che* that may indicate some linguistic phenomena such as emphasization (for instance, as in *"che brutta!"* - "what a ugly girl!"). On the other hand, another category for which some keywords seemed particularly important is the *dominance* one, but in that case the information gain obtained by sequences like *stfu* in English or *zitt* in Italian (related to the "shut up" meaning) was marginal. We suspect that the main problem may be related to the unbalanced training corpus in which the discredit category is dominant, but without knowing whether the other participants adopted some balancing technique it is difficult to analyze our results.

4 Conclusions

Our participation to the AMI task at EVALITA 2018 was not as successful as we hoped it to be; our systems in particular were not able to repeat the excellent results that they obtained at the DEFT 2018 challenge, although for a different task, the detection of messages related to public transportation in tweets. In particular, the bi-LSTM model underperformed and was outclassed by a simpler Random Forest model that uses locally weighted n-grams as features. At the time of writing, we are not able to assess if this was due to a misconfiguration of the neural network, or to the nature of the data, or the dataset. We hope that this participation and the comparison to the other systems will allow us to better understand where we have failed and why in view of future participations. The most positive point of our contribution is that the systems that we proposed are completely language-independent and we did not make any adjustment to adapt the systems that participated in a French task to the Italian or English language that were targeted in the AMI task.

Acknowledgments

We would like to thank the program "Investissements d'Avenir" overseen by the French National Research Agency, ANR-10-LABX-0083 (Labex EFL) for the support given to this work.

References

Maria Anzovino, Elisabetta Fersini, and Paolo Rosso. 2018. Automatic identification and classification of misogynistic language on twitter. In *Natural Language Processing and Information Systems - 23rd International Conference on Applications of Natural Language to Information Systems, NLDB 2018, Paris, France, June 13-15, 2018, Proceedings*, pages 57–64.

Leo Breiman. 2001. Random forests. *Machine learning*, 45(1):5–32.

Elisabetta Fersini, Debora Nozza, and Paolo Rosso. 2018. Overview of the evalita 2018 task on automatic misogyny identification (ami). In Tommaso Caselli, Nicole Novielli, Viviana Patti, and Paolo Rosso, editors, *Proceedings of the 6th evaluation campaign of Natural Language Processing and Speech tools for Italian (EVALITA'18)*, Turin, Italy. CEUR.org.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Changki Lee and Gary Geunbae Lee. 2006. Information gain and divergence-based feature selection for machine learning-based text categorization. *Information processing & management*, 42(1):155–165.

Patrick Paroubek, Cyril Grouin, Patrice Bellot, Vincent Claveau, Iris Eshkol-Taravella, Amel Fraisse, Agata Jackiewicz, Jihen Karoui, Laura Monceaux, and Torres-Moreno Juan-Manuel. 2018. Deft2018: recherche d'information et analyse de sentiments dans des tweets concernant les transports en île de france. In *14ème atelier Défi Fouille de Texte 2018*.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489.

Shiwei Zhang, Xiuzhen Zhang, and Jeffrey Chan. 2017. A word-character convolutional neural network for language-agnostic twitter sentiment analysis. In *Proceedings of the 22Nd Australasian Document Computing Symposium*, ADCS 2017, pages 12:1–12:4, New York, NY, USA. ACM.

Merging datasets for hate speech classification in Italian

Paula Fortuna[1] **Ilaria Bonavita**[2] **Sérgio Nunes**[1,3]

(1) INESC TEC and (3) FEUP, University of Porto
Rua Dr. Roberto Frias, s/n 4200-465 Porto PORTUGAL
paula.fortuna@fe.up.pt, sergio.nunes@fe.up.pt
(2) Eurecat, Centre Tecnològic de Catalunya
Carrer de Bilbao, 72, 08005 Barcelona

Abstract

This paper presents an approach to the shared task HaSpeeDe within Evalita 2018. We followed a standard machine learning procedure with training, validation, and testing phases. We considered word embedding as features and deep learning for classification. We tested the effect of merging two datasets in the classification of messages from Facebook and Twitter. We concluded that using data for training and testing from the same social network was a requirement to achieve a good performance. Moreover, adding data from a different social network allowed to improve the results, indicating that more generalized models can be an advantage.

ll manoscritto presenta un approccio per la risoluzione dello shared task HaSpeeDe organizzato all'interno di Evalita 2018. La classificazione è stata condotta con caratteristiche del testo estratte con word embedding e utilizzando algoritmi di deep learning. Abbiamo voluto sperimentare l'effetto dell'integrazione di messaggi di Facebook e Twitter ha e abbiamo ottenuto due risultati. 1) Addestrare modelli con un dataset integrato migliora le performance di classificazione in datasets provenienti dai singoli social network suggerendo una migliore capacità di generalizzazione del modello. 2) Tuttavia, utilizzare modelli addestrati su datasets provenienti da un social network per classificare messaggi provenienti da un altro social network comporta un peggioramento delle performance indicando che è indispensabile includere nel train set messaggi dello stesso social network che si è interessati a classificare nel test set.

1 Introduction

In the last few years, there is a growing attention to the automatic detection of hate speech in text. This appears as an answer to the increased spreading of online abuse in social networks. Several evaluation initiatives have been presenting different yet related classification tasks, e.g. TRAC (Kumar et al., 2018). Shared initiatives such as this, have the advantage of promoting the development of different but comparable solutions for the same problem, within a short period of time. In this paper, we describe the participation of the "Stop PropagHate" team in the HaSpeeDe task within Evalita 2018 (Bosco et al., 2018).

The goal of this task is to improve the automatic classification of hate speech in Italian. More specifically, there were three sub-tasks, promoting the development of features that would work independently of social network. For the task HaSpeeDe-FB, only the Facebook dataset could be used to train the model and classify Facebook data; for HaSpeeDe-TW, only the Twitter dataset could be used to classify Twitter data; and for the Cross-HaSpeeDe, only the Facebook dataset could be used to classify the Twitter and vice versa.

In our approach, we focused on understanding the effects of merging the two provided datasets. As features, we used word embeddings and deep learning for classification with a simple dense neural network. In this paper, we present the details of our approach, our results and conclusions.

2 Related Work

Previous research in the field of automatic detection of hate speech can give us insight into how to approach this problem. Two surveys summarize previous research and conclude that the approaches rely frequently on Machine Learning and classification (Schmidt and Wiegand, 2017; Fortuna and Nunes, 2018).

Regarding the automatic classification of messages, one first step is the gathering of training data. Several studies published datasets considering hate speech with different classification systems (Ross et al., 2017; Waseem and Hovy, 2016; Davidson et al., 2017; Nobata et al., 2016; Jigsaw, 2018). Although these could be useful datasets, the annotated language is not Italian. Regarding this language, the two existent datasets are used in this task (Del Vigna et al., 2017; Poletto et al., 2017; Sanguinetti et al., 2018).

After data collection, one of the most important steps when using classification is the process of feature extraction (Schmidt and Wiegand, 2017). Different methods are used, for instance word and character n-grams (Liu and Forss, 2014), perpetrator characteristics (Waseem and Hovy, 2016), othering language (Burnap and Williams, 2016) or word embedings (Djuric et al., 2015). Regarding the classification algorithms, the more common are, for instance, SVM (Del Vigna et al., 2017) or Random forests (Burnap and Williams, 2014). Another popular approach, due to its good results, is deep learning (Yuan et al., 2016; Gambäck and Sikdar, 2017; Park and Fung, 2017).

Different studies proved that deep learning algorithms outperform previous approaches. This was the case when using character or token-based n-grams with Recurrent Neural Network Language Model (RNN) (Mehdad and Tetreault, 2016); user behavioral characteristics with neural network composed of multiple Long-Short-Term-Memory (LSTM) (Park and Fung, 2017); Convolutional Neural Networks (CNN), LSTM and FastText (Badjatiya et al., 2017); morpho-syntactical features, sentiment polarity and word embedding lexicons with LSTM (Del Vigna et al., 2017); users' tendency towards racism or sexism with RNN (Pitsilis et al., 2018); abusive behavioral norms, available metadata, patterns within the text with RNN (Founta et al., 2018); n-grams, tf-idf, POS, sentiment, misspellings, emojis, special punctuation, capitalization, hashtags with CNN and GRU (Zhang et al., 2018); and word2vec with Convolutional Neural Networks (CNN) (Gambäck and Sikdar, 2017).

In this work, we propose an innovative approach in hate speech detection by merging different datasets, in the sequence of a previous experiment (Fortuna et al., 2018). We merged two datasets for aggression classification and the results showed that, although training with similar data is an advantage, adding data from different platforms allowed slightly better results.

Regarding the specificities of our approach in this contest, the main research question of our work concerns the effects of merging new datasets on the performance of models for hate speech classification. Accordingly with the previous study, we hypothesize that merging datasets will lead to a better performance. Additionally, we want to investigate how models perform when only data from different sources was used in the training.

In the next sections, we present our methodology and approach to this problem.

3 Methodology

3.1 Data

The data proposed for this task results of joining a collection of Facebook comments from 2016 (Del Vigna et al., 2017) with a Twitter corpus developed in 2018 (Poletto et al., 2017; Sanguinetti et al., 2018). Both consist of a total amount of 4,000 comments/tweets, randomly split into development and test set, of 3,000 and 1,000 messages respectively. The data format is the same with three tab-separated columns, each one representing the ID of the message, the text and the class (1 if the text contains hate speech, and 0 otherwise).

3.2 Text pre-processing

As a first step, we load the messages, remove the retweet marker "RT" in case of the tweets, and also the URL links present in the text.

3.3 Feature extraction and classification

We follow a methodology of classification with training, testing and validation. Keeping 30% of the data for validation allows us to estimate the results we would achieve in the contest. We use word embeddings and deep learning as presented in previous literature (Chollet and Allaire, 2018). We use the keras R package (Allaire et al., 2018) and make our approach available in a public repository[1].

3.3.1 Word embedding

In the procedure of feature extraction, we vectorize the text. We start by tokenizing the

[1] https://github.com/StopPropagHate/experiment_evalita_HaSpeeDe

data, considering only the top 10,000 words in the dataset. Additionally, we consider only the first 100 words of the tweets. We use the functions text_tokenizer, fit_text_tokenizer, texts_to_sequences and pad_sequences in our extraction.

3.3.2 Deep Learning

For the classification we use 10 fold cross-validation and apply a simple dense neural network. We use binary_crossentropy for loss with the rmsprop optimizer, we define the custom metric F1, so that it would be in according to the contest used metric. Regarding the model, we instantiate an empty model and we customize it:

- First we add an embedding layer where we specify the input_length (100, the maximum length of the messages) and give the dimensionality of the input data (dimensional space of 10,000). We add a dropout of 0.25.
- We flatten the output and add a dense layer, specified with 256 unit, with "relu" as a parameter. We add a dropout of 0.25.
- We add a dense layer with just a single neuron to serve as the output layer. Aiming for a single output, we use a sigmoid activation.

We use keras_compile function to compile and fit the model. We use batch size 128 and we tune the number of epochs starting by using 10. We also feed the model with the classes weights, corresponding to the frequencies of the classes in the training set. We average the F1 and loss results of the 10 folds for each epoch. For the epoch number, we kept the maximum number before overfitting to happen (the results only improving in the training set, but not in the test set). We save the final model and apply it to the validation data, with the function keras_predict. We conduct a permutation test in order to have a p-value associated to the F1.

4 Tasks and runs description

We conduct three different experiments following the procedure described in Section 3.

Task HaSpeeDe-FB In the HaSpeeDe-FB run1, we train and test with Facebook data. In the HaSpeeDe-FB run2, we mix Facebook with the Twitter provided data and see the effect in predicting hate speech in Facebook.

Task HaSpeeDe-TW We follow a similar procedure, but we switched the roles of Facebook and Twitter data. For theHaSpeeDe-TW run1 only Twitter data is used. In a second run HaSpeeDe-TW run2, we mix data for training and use Twitter for testing.

Task Cross-HaSpeeDe This is a proposed out-of-domain task. In the Cross-HaSpeeDe-FB, only the Facebook dataset can be used to classify Twitter data. In the Cross-HaSpeeDe-TW, only the Twitter dataset is used to classify Facebook data.

5 Results and Discussion

We separated the conditions with testing data from Facebook from Twitter and we compared three different conditions: the training data is from the same social network (1), the training data is both from and not from the social network (2), and the training data is not from the social network (3).

5.1 Results for Tuning and Validation

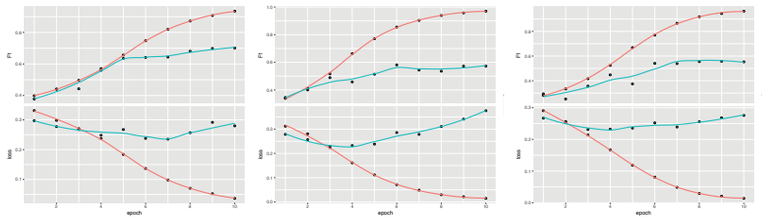
For each of the runs in our experiment we tuned the epoch parameter and we analyzed the average of the 10 folds for each of the 10 epochs (Figure 1). The decided number of epochs for each run is presented in the Table 1. We concluded that using mixed data for training (Condition 2) has a better performance (F1) than using data only from the social network (Condition 1). Additionally, using data only from other social network (Condition 3) provided poor results. Finally, classifying Facebook data was easier than Twitter data.

C.	system	epoch	F1	*p-value*
1	HaSpeeDe-FB run1	7	0.723	0.001
2	**HaSpeeDe-FB run2**	**3**	**0.738**	**0.001**
3	Cross-HaSpeeDe-TW	4	0.284	0.001
1	HaSpeeDe-TW run1	6	0.630	0.001
2	**HaSpeeDe-TW run2**	**4**	**0.679**	**0.001**
3	Cross-HaSpeeDe-FB	6	0.434	1

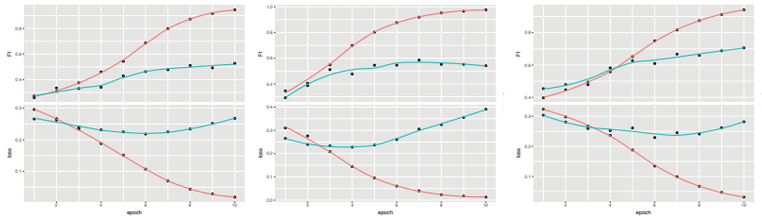
Table 1: F1 and respective *p-value* achieved in the validation set and respective Condition (C.).

5.2 Contest Results

Regarding the contest results (Table 2), similarly to the validation results we verified again that using mixed data for training (Condition 2) is better. Also in this case we verified that using only data from a different social network provided much worse results (Condition 3). Opposing to the validation results we found here that generally classifying Facebook data was more difficult than Twitter data.



(a) HaSpeeDe-FB run1 (b) HaSpeeDe-FB run2 (c) Cross-HaSpeeDe-TW



(d) HaSpeeDe-TW run1 (e) HaSpeeDe-TW run2 (f) Cross-HaSpeeDe-FB

Figure 1: Average of the 10 folds, for the metric F1 and loss, both for the training folds (blue) and validation fold (red). The results present each of the runs submitted by the team.

			Not HS			HS			
Test data	**C.**	**Run**	**Precision**	**Recall**	**F-score**	**Precision**	**Recall**	**F-score**	**Macro-Avg F-score (P.)**
Facebook	1	HaSpeeDe-FB run1	0,478	0,7089	0,571	0,8195	0,6307	0,7128	0,6419 (13)
	2	**HaSpeeDe-FB run2**	**0,4923**	**0,6965**	**0,5769**	**0,8195**	**0,6573**	**0,7295**	**0,6532 (12)**
	3	Cross-HaSpeeDe_TW	0,3606	0,9133	0,517	0,8461	0,2274	0,3585	0,4378 (11)
Twitter	1	HaSpeeDe-TW run1	0,7952	0,8964	0,8428	0,7058	0,5185	0,5978	0,7203 (11)
	2	**HaSpeeDe-TW run2**	**0,8628**	**0,7721**	**0,8149**	**0,6101**	**0,7438**	**0,6703**	**0,7426 (10)**
	3	Cross-HaSpeeDe_FB	0,6579	0,3727	0,4759	0,3128	0,5956	0,4102	0,443 (12)

Table 2: Macro Averaged F score and position (P.) achieved in the contest, respective Condition (C.) and Run. Precision, Recall and F-score are also provided for each of the classes hate speech (HS) and not hate speech (Not HS).

Regarding the main finding of this experiment, the results show that in this contest adding new data from a different social network brought improved performance. However, in the scope of this work it was not possible to investigate the reasons for this. One possibility may be the increased number of instances in the training when adding new datasets. Also using data from a different social network may bring less overfitting from training with only a dataset.

6 Conclusion

Throughout our approach to this shared task, our goal was to measure the effects of merging new datasets on hate speech classification. Supported by a previous experiment, we expected that adding data would help the classification. Indeed, we verified that merging datasets allowed us to have a small improvement of the results.

Complementary to this result, we tried the same approach following the same method and idea, in the Evalita 2018 AMI task. Merging datasets did not help for misoginy classification. In this case, we found that merging extra misogynistic or hate speech data kept the mysoginy classification with similar performance.

The reason why merging datasets worked in one case and not in the other remains unclear, and requires exploration in future studies. Possible variables interfering are the number of messages used for training and also the number of distinct words in the data.

Acknowledgments

This work was partially funded by the Google DNI grant Stop PropagHate.

References

J. J. Allaire, Francois Chollet, Yuan Tang, Daniel Falbel, Wouter Van Der Bijl, and Martin Studer. 2018. R interface to 'keras'. *Computer software manual](R package version 2.1.6). Retrieved from https://CRAN. R-project. org/package= keras*.

Pinkesh Badjatiya, Shashank Gupta, Manish Gupta, and Vasudeva Varma. 2017. Deep learning for hate speech detection in tweets. In *Proceedings of the 26th International Conference on World Wide Web Companion*, pages 759–760. International World Wide Web Conferences Steering Committee.

Cristina Bosco, Felice DellOrletta, Fabio Poletto, Manuela Sanguinetti, and Maurizio Tesconi. 2018. Overview of the EVALITA 2018 Hate Speech Detection Task. In Tommaso Caselli, Nicole Novielli, Viviana Patti, and Paolo Rosso, editors, *Proceedings of the 6th evaluation campaign of Natural Language Processing and Speech tools for Italian (EVALITA'18)*, Turin, Italy. CEUR.org.

Peter Burnap and Matthew L. Williams. 2014. Hate speech, machine classification and statistical modelling of information flows on Twitter: Interpretation and communication for policy decision making. In *Proceedings of Internet, Policy & Politics*, pages 1–18.

Pete Burnap and Matthew L. Williams. 2016. Us and them: identifying cyber hate on Twitter across multiple protected characteristics. *EPJ Data Science*, 5(1):11.

Francois Chollet and J. J. Allaire. 2018. *Deep Learning with R*. Manning Publications Co., Greenwich, CT, USA, 1st edition.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated Hate Speech Detection and the Problem of Offensive Language. In *Proceedings of ICWSM*.

Fabio Del Vigna, Andrea Cimino, Felice Dell'Orletta, Marinella Petrocchi, and Maurizio Tesconi. 2017. Hate me, hate me not: Hate speech detection on facebook. In *Proceedings of the First Italian Conference on Cybersecurity*, pages 86–95.

Nemanja Djuric, Jing Zhou, Robin Morris, Mihajlo Grbovic, Vladan Radosavljevic, and Narayan Bhamidipati. 2015. Hate speech detection with comment embeddings. In *Proceedings of the 24th International Conference on World Wide Web*, pages 29–30. ACM2.

Paula Fortuna and Sérgio Nunes. 2018. A survey on automatic detection of hate speech in text. *ACM Computing Surveys (CSUR)*, 51(4):85.

Paula Fortuna, José Ferreira, Luiz Pires, Guilherme Routar, and Sérgio Nunes. 2018. Merging datasets for aggressive text identification. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018)*, pages 128–139.

Antigoni-Maria Founta, Despoina Chatzakou, Nicolas Kourtellis, Jeremy Blackburn, Athena Vakali, and Ilias Leontiadis. 2018. A unified deep learning architecture for abuse detection. *arXiv preprint arXiv:1802.00385*.

Björn Gambäck and Utpal Kumar Sikdar. 2017. Using Convolutional Neural Networks to Classify Hate-speech. In *Proceedings of the First Workshop on Abusive Language Online*, pages 85–90.

Jigsaw. 2018. Toxic comment classification challenge identify and classify toxic online comments. Available in `https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge`, accessed last time in 23 May 2018.

Ritesh Kumar, Atul Kr. Ojha, Shervin Malmasi, and Marcos Zampieri. 2018. Benchmarking Aggression Identification in Social Media. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbulling (TRAC)*, Santa Fe, USA.

Shuhua Liu and Thomas Forss. 2014. Combining n-gram based similarity analysis with sentiment analysis in web content classification. In *International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management*, pages 530–537.

Yashar Mehdad and Joel Tetreault. 2016. Do characters abuse more than words? In *Proceedings of the SIGdial 2016 Conference: The 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 299–303.

Chikashi Nobata, Joel Tetreault, Achint Thomas, Yashar Mehdad, and Yi Chang. 2016. Abusive language detection in online user content. In *Proceedings of the 25th International Conference on World Wide Web*, pages 145–153. International World Wide Web Conferences Steering Committee.

Ji Ho Park and Pascale Fung. 2017. One-step and Two-step Classification for Abusive Language Detection on Twitter. In *Proceedings of the First Workshop on Abusive Language Online*.

Georgios K Pitsilis, Heri Ramampiaro, and Helge Langseth. 2018. Detecting offensive language in tweets using deep learning. *arXiv preprint arXiv:1801.04433*.

Fabio Poletto, Marco Stranisci, Manuela Sanguinetti, Viviana Patti, and Cristina Bosco. 2017. Hate speech annotation: Analysis of an italian Twitter corpus. In *CEUR WORKSHOP PROCEEDINGS*, volume 2006, pages 1–6. CEUR-WS.

Bjorn Ross, Michael Rist, Guillermo Carbonell, Benjamin Cabrera, Nils Kurowsky, and Michael Wojatzki. 2017. Measuring the reliability of hate speech annotations: The case of the european refugee crisis. *arXiv preprint arXiv:1701.08118*.

Manuela Sanguinetti, Fabio Poletto, Cristina Bosco, Viviana Patti, and Marco Stranisci. 2018. An italian Twitter corpus of hate speech against immigrants. In *Proceedings of LREC*.

Anna Schmidt and Michael Wiegand. 2017. A survey on hate speech detection using natural language processing. *SocialNLP 2017*, page 1.

Zeerak Waseem and Dirk Hovy. 2016. Hateful symbols or hateful people? predictive features for hate speech detection on Twitter. In *Proceedings of NAACL-HLT*, pages 88–93.

Shuhan Yuan, Xintao Wu, and Yang Xiang. 2016. A two phase deep learning model for identifying discrimination from tweets. In *International Conference on Extending Database Technology*, pages 696–697.

Ziqi Zhang, David Robinson, and Jonathan Tepper. 2018. Detecting hate speech on Twitter using a convolution-gru based deep neural network. In *European Semantic Web Conference*, pages 745–760. Springer.

HanSEL: Italian Hate Speech detection through Ensemble Learning and Deep Neural Networks

Marco Polignano
University of Bari Aldo Moro
Dept. Computer Science
via E. Orabona 4, 70125 Bari, Italy
marco.polignano@uniba.it

Pierpaolo Basile
University of Bari Aldo Moro
Dept. Computer Science
via E. Orabona 4, 70125 Bari, Italy
pierpaolo.basile@uniba.it

Abstract

English. The detection of hate speeches, over social media and online forums, is a relevant task for the research area of natural language processing. This interest is motivated by the complexity of the task and the social impact of its use in real scenarios. The task solution proposed in this work is based on an ensemble of three classification strategies, mediated by a majority vote algorithm: Support Vector Machine (Hearst et al., 1998) (SVM with RBF kernel), Random Forest (Breiman, 2001), Deep Multilayer Perceptron (Kolmogorov, 1992) (MLP). Each classifier has been tuned using a greedy strategy of hyper-parameters optimization over the "F1" score calculated on a 5-fold random subdivision of the training set. Each sentence has been pre-processed to transform it into word embeddings and TF-IDF bag of words. The results obtained on the cross-validation over the training sets have shown an F1 value of 0.8034 for Facebook sentences and 0.7102 for Twitter. The code of the system proposed can be downloaded from GitHub: https://github.com/marcopoli/haspeede_hate_detect

Italiano. *L'individuazione di discorsi di incitamento all'odio sui social media e sui forum on-line è una sfida rilevante per l'area di ricerca riguardante l'elaborazione del linguaggio naturale. Tale interesse è motivato della complessità del processo e dell'impatto sociale del suo utilizzo in scenari reali. La soluzione proposta in questo lavoro si basa su un insieme di tre strategie di classificazione mediate da un algoritmo di votazione per maggioranza: Support Vector Machine (Hearst et al., 1998) (SVM con kernel RBF), Random Forest (Breiman, 2001), Deep Multilayer Perceptron (Kolmogorov, 1992) (MLP). Ogni classificatore è stato configurato utilizzando una strategia greedy di ottimizzazione degli iper-parametri considerando il valore di "F1" calcolato su una suddivisione casuale in 5-fold del set di training. Ogni frase è stata pre-elaborata affinchè fosse trasformarta in formato word embeddings e TF-IDF. I risultati ottenuti tramite cross-validation sul training set hanno mostrato un valore F1 pari a 0.8034 per le frasi estratte da Facebook e 0.7102 per quelle di Twitter. Il codice sorgente del sistema proposto può essere scaricato tramite GitHub:* https://github.com/marcopoli/haspeede_hate_detect

1 Introduction and background

In the current digital era, characterized by the large use of the Internet, it is common to interact with others through chats, forums, and social networks. Common is also to express opinions on public pages and online squares. These places of discussion are frequently transformed into "fight clubs" where people use insults and strong words in order to support their ideas. The unknown identity of the writer is used as an excuse to fell free of consequences derived by attacking people only for their gender, race or sexual inclinations. A general absence of automatic moderation of contents can cause the diffusion of this phenomenon. In particular, consequences on the final user could be psychological problems such as depression, relational disorders and in the most critical situations also suicidal tendencies.

A recent survey of state of the art approaches for hate speech detection is provided by (Schmidt and Wiegand, 2017). The most common systems of speech detection are based on algorithms of text classification that use a representation of contents based on "surface features" such as them available in a bag of words (BOW) (Chen et al., 2012; Xu et al., 2012; Warner and Hirschberg, 2012; Sood et al., 2012). A solution based on BOW is efficient and accurate especially when n-grams have been extended with semantic aspects derived by the analysis of the text. (Chen et al., 2012) describe an increase of the classification performances when features such as the number of URLs, punctuations and not English words are added to the vectorial representation of the sentence. (Van Hee et al., 2015) proposed, instead, to add as a feature the number of positive, negative and neutral words found in the sentence. This idea demonstrated that the polarity of sentences positively supports the classification task. These approaches suffer from the lack of generalization of words contained into the bag of words, especially when it is created through a limited training set. In particular, terms found in the test sentences are often missing in the bag. More recent works have proposed word embeddings (Le and Mikolov, 2014) as a possible distributional representation able to overcome this problem. This representation has the advantage to transform semantically similar words into a similar numerical vector. Word embeddings are consequently used by classification strategies such as Support Vector Machine and recently by deep learning approaches such as deep recurrent neural networks (Mehdad and Tetreault, 2016). The solution proposed in this work reuse the findings of (Chen et al., 2012; Mehdad and Tetreault, 2016) for creating an ensemble of classifiers, including a deep neural network, which works with a combined representation of word embeddings and a bag of words.

2 Task and datasets description

The hate speech detection strategy proposed in *HAnSEL* has been developed for HaSpeeDe (Hate Speech Detection) task organized within Evalita 2018 (Caselli et al., 2018), which is going to be held in Turin, Italy, on December 12th-13th, 2018 (Bosco et al., 2018). HaSpeeDe consists in the annotation of messages from social networks (Twitter and Facebook) with a boolean label (0;1) that indicates the presence and absence of hate speeches. The task is organized into three subtasks, based on the dataset used for training and testing the participants' systems:

- **Task 1: HaSpeeDe-FB**, where only the Facebook dataset can be used to classify the Facebook test set
- **Task 2: HaSpeeDe-TW**, where only the Twitter dataset can be used to classify the Twitter test set
- **Task 3: Cross-HaSpeeDe**, which can be further subdivided into two sub-tasks:
  1. **Task 3.1: Cross-HaSpeeDe_FB**, where only the Facebook dataset can be used to classify the Twitter test set
  2. **Task 3.2: Cross-HaSpeeDe_TW**, where only the Twitter dataset can be used to classify the Facebook test set

The Facebook and Twitter datasets released for the task consist of a total amount of 4,000 comments/tweets, randomly split into development and test set, of 3,000 and 1,000 messages respectively. Data are encoded in a UTF-8 with three tab-separated columns, each one representing the sentence id, the text and the class (Fig. 1).

id	text	hs
8	Io votero NO NO E NO	0
36	Matteo serve un colpo di stato.	1

Table 1: Examples of annotated sentences.

3 Description of the system

The system proposed in this work is HanSEL: a system of Hate Speech detection through Ensemble Learning and Deep Neural Networks. We decided to approach the problem using a classic natural language processing pipeline with a final task of sentences classification into two exclusive classes: hate speech and not hate speech. The data provided by task organizers are obtained crawling social network, in particular, Facebook and Twitter. The analysis of the two data sources showed many possible difficulties to face in the case of using approaches based on Italian lexicons of hate speeches. In particular, we identified the following issues:

- **Repeated characters**: many words includes characters repeated many times for emphasizing the semantic meaning of the word. As an example, the words "nooooo", "Grandeeeee", "ccanaleeeeeeeeeeeeeee" are found in the training of Facebook messages.
- **Emoji**: sentences are often characterized by emoji such as hearts and smiley faces that are often missing in external lexicons.
- **Presence of links, hashtags and mentions**: this particular elements are typical of the social network language and can introduce noise in the data processing task.
- **Length of the sentences**: many sentences are composed by only one word or in general, they are very short. Consequently, they are not expressive of any semantic meaning.

The complexity of the writing style used in hate speech sentences guided us through the idea to do not use an approach based on standard lexicons and to prefer supervised learning strategies on the dataset provided by the task organizers.

Sentence processing.
We decide to represent each sentence as a concatenation of a 500 features word embedding vector and a 7,349 size bag of words for Facebook messages and 24,866 size bag of words for Twitter messages. In particular, the word-embedding procedure used is word2vec introduced by Mikolov (Mikolov et al., 2013). This model learns a vector representation for each word using a neural network language model and can be trained efficiently on billions of words. Word2vec allows being a very efficient data representation in text classification due to its capability to create very similar vectors for words strongly semantically related. The Italian word embeddings used in this work are provided by Tripodi (Tripodi and Li Pira, 2017). The author trained the model on a dump of the Italian Wikipedia (dated 2017.05.01), from which only the body text of each article is used. The corpus consists of 994,949 sentences that result in 470,400,914 tokens. The strategy of the creation of word embeddings is CBOW with the size of the vectors equal to 500, the window size of the words contexts set to 5, the minimum number word occurrences equal to 5 and the number of negative samples set to 10.

We follow the same step of pre-processing applied by Tripodi (Tripodi and Li Pira, 2017) to transform the sentence of the task datasets into word embeddings. In particular, we applied the following Natural Language Processing pipeline:

- Reduction of repeated characters: we scan each sentence of the datasets (both training and test). For each sentence, we obtain words merely splitting it by space. Each word is analyzed, and characters repeated three times or more are reduced to only two symbols, trying to keep intact word that naturally includes doubles.
- Data cleansing: we transformed the words is lowercase and following we removed from each sentences links, hashtags, entities, and emoji

The normalized sentences are consequently tokenized using the TweetTokenizer of the NLTK library [1]. For each sentence we averaged the word2vec vectors correspondent of each token, removing during each sum the centroid of the whole distributional space. This technique is used for mitigating the problems of loss of information due to the operation of averaging the semantic vectors.

The two bags of words (Facebook and Twitter) are, instead, created directly on the sentences without any pre-processing, also if during the tuning of the architecture we had tried some configurations that include bag of words without stop words, with lowercase letters and processed by Snowball stemmer algorithm [2] without obtaining breaking results. The n-gram size considered for the construction of the bag is in the range of 1 to 3. The final representation of each sentence of the dataset is consequently obtained concatenating the word2vec vector and the correspondent bag of words. Sentences too shorts that cannot be transformed into word2vec as a consequence of the absence of all the tokens of the sentence have been classified using only the bag of words representation.

Classification strategy.
HAnSEL is based on a classification process that uses three different classification strategies mediated by a hard majority vote algorithm. A stacking of classifiers with a Random Forest blender

[1] https://www.nltk.org/data.html
[2] http://snowball.tartarus.org/texts/quickintro.html

has also been considered during the design of HAnSEL architecture but, the internal evaluation runs on 5-fold cross-validation of the training set showed us low performances of the approach. This analysis is not detailed more in this work due to the page limitations of it. In order to design the ensemble, we analyzed the performances of some of the most popular classification algorithms for the text categorization task. In particular, we considered:

- Logistic regression with stochastic gradient descendent training (SGD). It has the advantage to be very efficient with large datasets considering, during the training, one instance per time independent by others. It uses the gradient descendent as optimization function for learning the optimal weight of the separation function of the distributional space of items. In literature, it has been successfully used for tasks of text classification, especially with binary classification problems.

- C-Support Vector Classification (SVC). It is the standard Support Vector Machine algorithm applied for the classification task. It is a powerful approach that supports linear and non-linear classification function. Moreover, through the C parameter, it is possible to decide how much the margin of classification could be significant and consequently sensitive to outliers. The implementation is based on libsvm, and we evaluated different configurations of the algorithm: polynomial function with 2 and 3 degree, RBF kernel and different values of the C parameters.

- K-nearest neighbors vote (KNN). This classic and versatile algorithm is based on the concept of similarity among items according to a distance metric. In particular, for an unseen item, the k most similar items of the training set are retrieved, and the class, provided as output, is obtained by the majority vote of the neighborhoods. Despite the simplicity of the algorithm it is often used in tasks of text classification.

- A decision tree classifier (DT). This approach is another popular strategy of classification used especially when it is required to visualize the model. The DT algorithm works splitting items into a different path of the tree according to their feature values. In order to classify an unseen item, the three is navigated until reaching the leaf and then the ration of training items of class k in that leaf is used as a class probability.

- Random forest classifier (RF). It is an ensemble of Decision Trees trained on different batches of the dataset that uses averaging to improve the predictive accuracy and control over-fitting. A typical parameter is the number of threes to use in order to balance the precision of the algorithm and the randomness to obtain a good level of generalization of the model.

- Multi-layer Perceptron classifier (MLP). This model is a classical architecture of a deep neural network. It is composed by one layer of inputs, one layer of linear threshold units (LTU) as output and many hidden layers of an arbitrary number of LTU plus one bias neuron fully connected each other. The weights learned by each neuron (perceptron) are updated through back-propagation using a gradient descendent strategy. Important parameters to configuring are the number of hidden layers, the number of training epochs and the L2 penalty (regularization term) parameter.

We evaluated the performance of the algorithms just described using a default configuration and a 5-fold cross validation over the Facebook training set. Moreover, we set the random seed equal to 42 for obtaining at each run always the same folder subdivision.

Tab. 3 shows the results obtained by the different classification algorithms during their preliminary analysis considering the macro F1 score as in the task specifications. The values obtained do not point out significant statistical differences among the approaches, but we decided to investigate more the top three scored algorithms: SVM with an RBF kernel, Random Forest with 300 trees, MLP with 2,000 hidden layers. In general, we observed that linear algorithms obtain a high score for the task supporting our idea that linguistic features are enough for defining a clear separation among the sentences of hate and not hate speeches. In order to identify an optimal configuration of the algorithms, we trained our models using a greedy search approach. For each algorithm, we performed 100 training runs with pa-

	Not HS			HS				
	Precision	**Recall**	**F1-score**	**Precision**	**Recall**	**F1-score**	**Macro F1**	**Pos.**
Task 1	0,6981	0,6873	0,6926	0,8519	0,8581	0,855	*0,7738*	7
Task 2	0,7541	0,8801	0,8122	0,6161	0,4012	0,4859	*0,6491*	14
Task 3.1	0,7835	0,2677	0,3991	0,3563	0,8456	0,5013	*0,4502*	11
Task 3.2	0,3674	0,8235	0,5081	0,7934	0,3234	0,4596	*0,4838*	8

Table 2: Final scores obtained during the HASpeeDe challenge

Algorithm	**Macro F1 score**
LR	0.780444109
SVC-rbf - C= 1	*0.789384136*
SVC-poly 2 C=1	0.758599844
SVC-poly 3 C=1	0.667374386
KNN - 3	0.705064332
KNN - 5	0.703990867
KNN - 10	0.687719117
KNN - 20	0.663451598
DT	0.68099986
RF-50	0.75219596
RF-100	0.764247578
RF-300	*0.787778421*
RF-500	0.768494151
MLP-1000	0.766835616
MLP-2000	*0.791230474*
MLP-3000	0.76952709

Table 3: Classification algorithms on Facebook training set using 5-fold cross validation.

rameters randomly selected from a range of values preliminary defined. Each run has been evaluated, considering the macro F1 score, on the training set using the same strategy of cross-validation already described before. At the end of the 100 runs the model that achieve the best results has been stored and later used in our final ensemble of classifiers. The final configurations obtained for the three strategies are the following:

- SVC(C=1.76800710431488, gamma=0.1949764030136127, kernel='rbf')
- RandomForestClassifier(bootstrap=False, max_depth=30,max_features='sqrt', min_samples_leaf=2,min_samples_split=2, n_estimators=200, warm_start=False)
- MLPClassifier(alpha=0.5521952082781035, early_stopping=False, hidden_layer_sizes=2220, learning_rate_init=0.001, max_iter=184,solver='adam', warm_start=False)

The models are consequently used in a voting classifier configured for using a hard majority vote algorithm. The ensemble obtains an F1 value of 0.8034 for Facebook sentences and 0.7102 for Twitter using the 5-fold subdivision of the training sets. The implementation of the system has been realized into Python language and using the scikit-learn 0.20 machine learning library [3].

4 Results and discussion

HanSEL has been used for classifying the data provided as a test set for each of the three specialized tasks of HaSpeeDe competition. Tab. 2 shows the final results obtained by our system in the challenge. It is possible to observe that the system well performed for Task 1 and Task 3.2 which involve the classification of Facebook messages. In particular, it emerges that HanSEL performs better fot hate speeches sentences than for not hate speeches probably a consequence of the presence of many clear hate words used in this type of messages such as "sfigati" and "bugiardo" in that category of textual sentences. A symmetrical situation is obtained for Task 2 and Task 3.2 that involves Twitter messages. In this scenario, the significant use of specific hashtags, irony, and entities instead of clear hate words has made difficult the identification of hate speeches. The cross-classification task has, moreover, stressed the generalization of the system. It has been observed that the writing style of the two social networks strongly influences the classification performance, especially when the models are trained on a small training set, as in our case. Finally, the optimization of the models inside the ensemble has been stressed more on the Facebook dataset consequently overfitting on the characteristics of that type of messages. The outcomes achieved for the challenge

[3] http://scikit-learn.org/stable/

allow us to deduce important consideration for further developments of the system. In particular, we consider essential to mix the two datasets in order to allow the models to generalize better considering the two different sources of data. Moreover, extra features regarding hashtags, entities, and links can be helpful for obtaining better results with Twitter messages.

5 Conclusion

The HaSpeeDe competition has been a perfect scenario for developing and testing solutions for the social problem of hate speeches on social media and, in particular, for them in the Italian language. In our work, we presented *HAnSEL* a system based on an ensemble of classifiers that includes the Support Vector Machine algorithm, Random Forests, and a Multilayers Perceptron Deep Neural Network. We formalize messages as a concatenation of word2vec sentence vectors and a TF-IDF bag of words. Results showed the efficacy of the solution in a scenario that uses clear offensive words such as Facebook messages. On the contrary, there is a large margin of improvements for the classification of Tweets. The future direction of the work will surely investigate the use of more data and semantic features for allowing classification methods to create a more general model.

References

Cristina Bosco, Felice Dell'Orletta, Fabio Poletto, Manuela Sanguinetti, and Maurizio Tesconi. 2018. Overview of the EVALITA 2018 HaSpeeDe Hate Speech Detection (HaSpeeDe) Task. In Tommaso Caselli, Nicole Novielli, Viviana Patti, and Paolo Rosso, editors, *Proceedings of the 6th evaluation campaign of Natural Language Processing and Speech tools for Italian (EVALITA'18)*, Turin, Italy. CEUR.org.

Leo Breiman. 2001. Random forests. *Machine learning*, 45(1):5–32.

Tommaso Caselli, Nicole Novielli, Viviana Patti, and Paolo Rosso. 2018. EVALITA 2018: Overview of the 6th Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. In Tommaso Caselli, Nicole Novielli, Viviana Patti, and Paolo Rosso, editors, *Proceedings of Sixth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2018)*, Turin, Italy. CEUR.org.

Ying Chen, Yilu Zhou, Sencun Zhu, and Heng Xu. 2012. Detecting offensive language in social media to protect adolescent online safety. In *Privacy, Security, Risk and Trust (PASSAT), 2012 International Conference on and 2012 International Confernece on Social Computing (SocialCom)*, pages 71–80. IEEE.

Marti A. Hearst, Susan T Dumais, Edgar Osuna, John Platt, and Bernhard Scholkopf. 1998. Support vector machines. *IEEE Intelligent Systems and their applications*, 13(4):18–28.

Věra Kolmogorov. 1992. Kolmogorov's theorem and multilayer neural networks. *Neural networks*, 5(3):501–506.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International Conference on Machine Learning*, pages 1188–1196.

Yashar Mehdad and Joel Tetreault. 2016. Do characters abuse more than words? In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 299–303.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Anna Schmidt and Michael Wiegand. 2017. A survey on hate speech detection using natural language processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media*, pages 1–10.

Sara Sood, Judd Antin, and Elizabeth Churchill. 2012. Profanity use in online communities. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1481–1490. ACM.

Rocco Tripodi and Stefano Li Pira. 2017. Analysis of italian word embeddings. *arXiv preprint arXiv:1707.08783*.

Cynthia Van Hee, Els Lefever, Ben Verhoeven, Julie Mennes, Bart Desmet, Guy De Pauw, Walter Daelemans, and Véronique Hoste. 2015. Detection and fine-grained classification of cyberbullying events. In *International Conference Recent Advances in Natural Language Processing (RANLP)*, pages 672–680.

William Warner and Julia Hirschberg. 2012. Detecting hate speech on the world wide web. In *Proceedings of the Second Workshop on Language in Social Media*, pages 19–26. Association for Computational Linguistics.

Jun-Ming Xu, Kwang-Sung Jun, Xiaojin Zhu, and Amy Bellmore. 2012. Learning from bullying traces in social media. In *Proceedings of the 2012 conference of the North American chapter of the association for computational linguistics: Human language technologies*, pages 656–666. Association for Computational Linguistics.

Comparing Different Supervised Approaches to Hate Speech Detection

Michele Corazza[†], **Stefano Menini**[‡], **Pinar Arslan**[†], **Rachele Sprugnoli**[‡]
Elena Cabrio[†], **Sara Tonelli**[‡], **Serena Villata**[†]
[†]Université Côte d'Azur, CNRS, Inria, I3S, France
[‡]Fondazione Bruno Kessler, Trento, Italy
{michele.corazza,pinar.arslan}@inria.fr
{menini,sprugnoli,satonelli}@fbk.eu
{elena.cabrio,serena.villata}@unice.fr

Abstract

English. This paper reports on the systems the InriaFBK Team submitted to the EVALITA 2018 - Shared Task on Hate Speech Detection in Italian Twitter and Facebook posts (HaSpeeDe). Our submissions were based on three separate classes of models: a model using a recurrent layer, an ngram-based neural network and a LinearSVC. For the Facebook task and the two cross-domain tasks we used the recurrent model and obtained promising results, especially in the cross-domain setting. For Twitter, we used an ngram-based neural network and the LinearSVC-based model.

Italiano. *Questo articolo descrive i modelli del team InriaFBK per lo Shared Task on Hate Speech Detection in Italian Twitter and Facebook posts (HaSpeeDe) di EVALITA 2018. Tre classi di modelli differenti sono state utilizzate: un modello che usa un livello ricorrente, una rete neurale basata su ngrammi e un modello basato su LinearSVC. Per Facebook e i due task cross-domain, si è scelto un modello ricorrente che ha ottenuto buoni risultati, specialmente per quanto riguarda i task cross-domain. Per Twitter, sono stati utilizzati la rete neurale basata su ngrammi e il modello basato su LinearSVC.*

1 Introduction

In this paper, we describe the submitted systems for each of the four subtasks organized within the HaSpeeDe evaluation exercise at EVALITA 2018 (Bosco et al., 2018): Hate speech detection on Facebook comments (Task 1: HaSpeeDe-FB), Hate speech detection on tweets (Task 2: HaSpeeDe-TW), Cross-domain task hate speech detection from Facebook to Twitter posts (Task 3.1: Cross-HaSpeeDe_FB) and Cross-domain task hate speech detection from Twitter to Facebook posts (Task 3.2: Cross-HaSpeeDe_TW). We build our models for these binary classification subtasks testing recurrent neural networks, ngram-based neural networks[1] and a LinearSVC (Support Vector Machine) approach[2]. In HaSpeeDe-TW, which has comparatively short sequences with respect to HaSpeeDe-FB, an ngram-based neural network and a LinearSVC model were used, while for HaSpeeDe-FB and the two cross-domain tasks recurrent models were used.

2 System Description

We adopt a supervised approach and, to select the best model for each task, we perform grid search over different machine learning classifiers such as Neural Networks (NN), Support Vector Machines (SVM) and Logistic Regression (LR). Both ngram-based (unigram and bigram) and recurrent models using embeddings were tested, but only the ones that were submitted for the tasks will be described. A LinearSVC model from scikit-learn (Pedregosa et al., 2011a) was also tested, and it showed good performance on the Twitter dataset. In order to perform a grid search over the parameters and models, the training set released by the task organisers was partitioned in three: 60% of it was used for training, 20% for validation and 20% for testing.[3]

2.1 Preprocessing

Since misspellings, neologisms, acronyms and jargon are common in social media interactions, it was necessary to carefully preprocess the data, in

[1] https://gitlab.com/ashmikuz/creep-cyberbullying-classifier

[2] https://github.com/0707pinar/Hate-Speech-Detection/

[3] To split the data we use the scikit-learn `train_test_split` function, using 42 as seed value.

order to normalize it without losing information. For this reason, we first replace URLs with the word "url" and "@" user mentions with "username" by using regular expressions.

Since hashtags often provide important semantic content, they are normalized by splitting them into the words composing them. To this end, we adapted to Italian the Ekphrasis tool (Baziotis et al., 2017), using as ngram model the Italian Google ngrams starting from year 2000. In addition to the aforementioned normalizations, for the LinearSVC model we also stemmed Italian words via the Snowball Stemmer (Bird and Loper, 2004) and we removed stopwords.

2.2 Feature Description

We used the following text-derived features:

- **Word Embeddings**: Italian fastText embeddings (Bojanowski et al., 2016)[4] employed in the recurrent models (Section 2.3);
- **Ngrams**: unigrams and bigrams, used for the ngram-based neural network and the linearSVC (Sections 2.4, 2.5);
- **Social-network specific features**: the number of hashtags and mentions, the number of exclamation and question marks, the number of emojis, the number of words that are written in uppercase.
- **Sentiment and Emotion features**: the word-level emotion and sentiment tags for Italian words extracted from the EmoLex (Mohammad and Turney, 2013; Mohammad and Turney, 2010) resource.

2.3 Recurrent Neural Network Model

In order to classify hate speech in social media interactions, we believe that recurrent neural networks are a useful tool, given their ability to remember the sequence of inputs while considering their order, differently from the feed-forward models. In the context of our classifier, this allows the model to remember the whole sequence of words in the order they appear in.

More specifically, our recurrent models, implemented using Keras (Chollet and others, 2015), combine both sequences of word embeddings and social media features. In order to achieve that, an asymmetric topology is used for the neural network: the sequences of word embeddings are fed to a recurrent layer, whose output is then concatenated with the social features. The concatenated vector is then fed to one or two feed forward fully connected layers that use the Rectified Linear Unit (ReLU) as their activation function. The output layer is a single neuron with a sigmoid activation, while binary cross-entropy is used as the loss function for the model.

Batch normalization and various kinds of dropout have been tested to reduce the variance of the models. Experimental results suggested that applying the former to the output of the recurrent layer had a negative effect on performance. For this reason, batch normalization was applied only to the output of the hidden layers. As for dropout, we tried three different mechanisms. A simple dropout layer (Srivastava et al., 2014) is applied to the output of the hidden layers, as applying dropout to the output of the recurrent layer introduces too much noise and does not improve performance. We also tested a dropout on the embeddings (Gal and Ghahramani, 2016) that effectively skips some of the word embeddings in the sequence, as dropping part of the embedding vector causes a loss of information, while dropping entire words can help reduce overfitting. In addition, a recurrent dropout (Gal and Ghahramani, 2016) was also tested. While evaluating the models, we tested both a Long Short Term Memory (LSTM) (Gers et al., 1999) and a Gated Recurrent Unit (GRU) (Cho et al., 2014) as recurrent layers. The latter is functionally very similar to an LSTM but by using less weights it can sometimes reduce the variance of the model, improving its performance.

2.4 Ngram-based Neural Networks

Ngram-based neural networks are structurally similar to the recurrent models. We first compute the unigrams and bigrams over the lemmatized social media posts. The resulting vector is then normalized by using tf-idf from scikit-learn and concatenated to the social-specific features. One or two hidden feed-forward layers are then used, and the same output layer as in the recurrent models is used. The same dropout and batch normalization techniques used in the recurrent models have been tested for the ngram-based neural networks as well. For the first submitted run of Task

[4] https://github.com/facebookresearch/fastText

2: HaSpeeDe-TW, we used unigrams and bigrams along with the required preprocessing steps based on tf-idf model.

2.5 Linear SVC System

We implemented a Linear Support Vector Classification system (i.e., LinearSVC) (Fan et al., 2008) based on bag-of-words (i.e., unigrams), using scikit-learn (Pedregosa et al., 2011b) for the first submitted run in Task 2: HaSpeeDe-TW. We chose this system as it scales well for large-scale samples, and it is efficient to solve text classification problems. To deal with imbalanced labels, we set the `class_weight` parameter as "balanced". To mitigate overfitting, penalty parameter `C` was scaled as 0.7.

3 Submitted Runs and Results

In this Section we describe the single runs submitted for each task and we present the results. The official ranking reported for each run is given in terms of macro-average F-score.

3.1 Task 1: HaSpeeDe-FB

For Task 1: HaSpeeDe-FB, two recurrent models were used. The first submitted run used a single fully connected layer of size 200 and a GRU of size 100 as the recurrent layer. Recurrent dropout was applied to the GRU with value 0.2. The second submitted run used two fully connected layers of size 500 and a GRU of size 300 as the recurrent layer. Simple dropout was applied to the output of the feed-forward layers with value 0.5. The first run ranked third and the second ranked fourth out of 18 submissions (Table 1). As shown in Table 1, both runs yield a better performance on the hate speech class.

First Run				
Category	**P**	**R**	**F1**	**Instances**
Non Hate	0.763	0.687	0.723	323
Hate	0.858	0.898	0.877	677
Macro AVG	0.810	0.793	**0.800**	1000
Second Run				
Non Hate	0.716	0.703	0.709	323
Hate	0.859	0.867	0.863	677
Macro AVG	0.788	0.785	0.786	1000

Table 1: Results on HaSpeeDe-FB

3.2 Task 2: HaSpeeDe-TW

In the first submitted run for Task 2: HaSpeeDe-TW, we used the LinearSVC-based model described in subsection 2.5. This run was ranked sixth out of 19 submissions. As our second run on the Task 2: HaSpeeDe-TW, an ngram-based neural network was used having a single fully connected hidden layer with size 200. Simple dropout was applied to the hidden layer with value 0.5. This run ranked fourth. Both runs show better performance when classifying the non hate speech class as displayed in Table 2.

First Run				
Category	**P**	**R**	**F1**	**Instances**
Non Hate	0.873	0.827	0.850	676
Hate	0.675	0.750	0.711	324
Macro AVG	0.774	0.788	0.780	1000
Second Run				
Non Hate	0.842	0.899	0.870	676
Hate	0.755	0.648	0.698	324
Macro AVG	0.799	0.774	**0.784**	1000

Table 2: Results on HaSpeeDe-TW

3.3 Task 3.1: Cross-HaSpeeDe_FB

For Task 3.1: Cross-HaSpeeDe_FB two recurrent models were used. In the first submitted run, two hidden layers of size 500 were used. An LSTM of size 200 was adopted as the recurrent layer. Embeddings dropout was applied with value 0.5 and a simple dropout was applied to the output of the feed-forward layers with value 0.5. The recurrent model for the second run had one hidden layer of size 500. A GRU of size 200 was used as the recurrent layer and no dropout was applied. The first run ranked second out of 17 submissions while the second run registered the best score in the Task 3.1: Cross-HaSpeeDe_FB. In both runs, the models showed good performance over the non hate speech class, whereas the precision on the hate speech class does not exceed 0.5 (see Table 3).

First Run				
Category	**P**	**R**	**F1**	**Instances**
Non Hate	0.810	0.675	0.736	676
Hate	0.497	0.670	0.570	324
Macro AVG	0.653	0.672	0.653	1000
Second Run				
Non Hate	0.818	0.660	0.731	676
Hate	0.494	0.694	0.580	324
Macro AVG	0.656	0.677	**0.654**	1000

Table 3: Results on Cross-HaSpeeDe_FB

3.4 Task 3.2: Cross-HaSpeeDe_TW

For Task 3.2: Cross-HaSpeeDe_TW two recurrent models were used. In the first submitted run, two

hidden layers of size 500 were used together with a GRU of size 200 as the recurrent layer. Simple dropout was applied to the output of the feed-forward layers with value 0.2, whereas the recurrent dropout has value 0.2. In the second submitted run, one hidden layer of size 200 was used adopting an LSTM of size 200 as the recurrent layer. Embeddings dropout was applied with value 0.5. The first run ranked fourth out of 17 submissions, while the other run ranked second. Table 4 shows that in both cases the system showed good performance over the hate speech class, while detecting negative instances proved difficult, in particular in terms of precision over the non hate speech class.

First Run				
Category	**P**	**R**	**F1**	**Instances**
Non Hate	0.493	0.703	0.580	323
Hate	0.822	0.656	0.730	677
Macro AVG	0.658	0.679	0.655	1000
Second Run				
Non Hate	0.537	0.653	0.589	323
Hate	0.815	0.731	0.771	677
Macro AVG	0.676	0.692	**0.680**	1000

Table 4: Results on Cross-HaSpeeDe_TW

4 Error Analysis and Discussion

Although all our runs obtained satisfactory results in each task, there is still room for improvement. In particular, we noticed that our models have problems in classifying social media messages containing the following specific phenomena: (i) dialects (e.g. "un se ponno sentì...ma come se fà...") or bad orthography (e.g. "Io no nesdune delle due.....momti pesanti"); (ii) sarcasm, "Dopo i campi rom via pure i centri sociali. L'unico problema sarà distinguere gli uni dagli altri"; (iii) references to world knowledge, typically used for an indirect attack not containing an explicit insult (e.g. "un certo Adolf sarebbe utile ancora oggi con certi soggetti"); (iv) metaphorical expressions, usually referring to ways to physically eliminate the targets of hate speech messages (e.g. "Ruspali").

As for false positives, some errors come from the misclassification of messages containing the lemmas "terrorista", "terrorismo", "immigrato" that are extremely frequent in particular in the Twitter dataset. These lemmas are associated to the hate speech class even when they appear in messages reporting the title of a news, eg. "Il Giappone senza immigrati a corto di forza lavoro".

In Task 2: HaSpeeDe-TW, when the classifier relies on sentiment and emotion features, we registered several misclassified instances containing relevant content words not covered by EmoLex. This is due to the fact that for every English word, EmoLex provides only one translated entry, thus limiting the overall coverage. For instance, "to kill" is translated in Italian with "uccidere" not considering synonyms such as "ammazzare" often used in the dataset.

Finally, we noticed some inconsistencies in the gold standard. For example, the message "Al solo vederle danno il voltastomaco!" is annotated as hate speech while, the almost equivalent, "Appena le ho viste ho vomitato" is considered a non hate speech instance while our models identify it as hate speech. Similarly, an insult like "ridicoli" is annotated as non hate speech in "CERTO CHE GLI ONOREVOLI DEL PD SI RICONOSCONO A KILOMETRI ... RIDICOLI" but as hate speech in "Ci vorrebbe anche qua Putin, invece di quei RIDICOLI...PAROLACCE PAROLACCE".

5 Conclusions

In this paper we presented an overview of the runs submitted for the four subtasks of HaSpeeDe evaluation exercise. We implemented a number of different models, comparing recurrent neural networks, ngram-based neural networks and linear SVC. While RNNs perform better in three of four tasks, classification on Twitter data achieves a better ranking using the ngram based neural network. Our system was ranked first among all the teams in one of the cross-domain task, i.e. Cross-HaSpeeDe_FB. This is probably due to the fact that considering the whole sequence of inputs with a recurrent neural networks and using a pre-learned representation by using word embeddings help the model to learn some common traits of hate speech across different social media.

Acknowledgments

Part of this work was funded by the CREEP project (`http://creep-project.eu/`), a Digital Wellbeing Activity supported by EIT Digital in 2018. This research was also supported by the HATEMETER project (`http://hatemeter.eu/`) within the EU Rights, Equality and Citizenship Programme 2014-2020.

References

Christos Baziotis, Nikos Pelekis, and Christos Doulkeridis. 2017. DataStories at SemEval-2017 Task 4: Deep LSTM with Attention for Message-level and Topic-based Sentiment Analysis. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 747–754, Vancouver, Canada, August. Association for Computational Linguistics.

Steven Bird and Edward Loper. 2004. Nltk: the natural language toolkit. In *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*, page 31. Association for Computational Linguistics.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching Word Vectors with Subword Information. *arXiv preprint arXiv:1607.04606*.

Cristina Bosco, Felice Dell'Orletta, Fabio Poletto, Manuela Sanguinetti, and Maurizio Tesconi. 2018. Overview of the EVALITA 2018 HaSpeeDe Hate Speech Detection (HaSpeeDe) Task. In Tommaso Caselli, Nicole Novielli, Viviana Patti, and Paolo Rosso, editors, *Proceedings of the 6th evaluation campaign of Natural Language Processing and Speech tools for Italian (EVALITA18)*, Turin, Italy, December. CEUR.org.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

François Chollet et al. 2015. Keras. `https://keras.io`.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Liblinear: A library for large linear classification. *Journal of machine learning research*, 9(Aug):1871–1874.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in neural information processing systems*, pages 1019–1027.

Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. 1999. Learning to forget: Continual prediction with LSTM.

Saif M Mohammad and Peter D Turney. 2010. Emotions evoked by common words and phrases: Using mechanical turk to create an emotion lexicon. In *Proceedings of the NAACL HLT 2010 workshop on computational approaches to analysis and generation of emotion in text*, pages 26–34. Association for Computational Linguistics.

Saif M Mohammad and Peter D Turney. 2013. Crowdsourcing a word–emotion association lexicon. *Computational Intelligence*, 29(3):436–465.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011a. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011b. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Hate Speech Detection using Attention-based LSTM

Gretel Liz De la Peña Sarracén[1], **Reynaldo Gil Pons**[2], **Carlos Enrique Muñiz Cuza**[2], **Paolo Rosso**[1]
[1]PRHLT Research Center, Universitat Politècnica de València, Spain
gredela@posgrado.upv.es
prosso@dsic.upv.es
[2]CERPAMID, Cuba
{rey,carlos}@cerpamid.co.cu

Abstract

English. This paper describes the system we developed for EVALITA 2018, the 6th evaluation campaign of Natural Language Processing and Speech tools for Italian, on Hate Speech Detection (HaSpeeDe). The task consists in automatically annotating Italian messages from two popular micro-blogging platforms, Twitter and Facebook, with a boolean value indicating the presence or not of hate speech. We propose an Attention-based in Long Short-Term Memory Recurrent Neural Network where the attention layer helps to calculate the contribution of each part of the text towards targeted hateful messages.

Italiano. *In questo articolo descriviamo il sistema che abbiamo sviluppato per il task di Hate Speech Detection (HaSpeeDe), presso EVALITA 2018, la sesta campagna di valutazione dellelaborazione del linguaggio naturale. Il task consiste nellannotare automaticamente testi italiani da due popolari piattaforme di micro-blogging, Twitter e Facebook, con un valore booleano indicando la presenza o meno di incitamento allodio. Il nostro approccio usa una rete neurale ricorrente LSTM attention-based, in cui il layer di attenzione aiuta a calcolare il contributo di ciascuna porzione del testo verso messaggi di odio mirati.*

1 Introduction

In recent years, Hate Speech (HS) has become a major issue as a hot topic in the domain of social media. Some key aspects (such as virality, or presumed anonymity) that characterize it, distinguish it from offline communication and make it potentially more dangerous and hurtful. Therefore, the identification of HS is an important step for dealing with the urgent need for effective counter measures to this issue.

The evaluation campaign EVALITA 2018[1] launched this year the HaSpeeDe (Hate Speech Detection) task[2] (Bosco et al., 2018). It consists in automatically annotating messages from two popular micro-blogging platforms, Twitter and Facebook, with a boolean value indicating the presence (or not) of HS.

Deep neural network are greatly studied due to their flexibility in capturing nonlinear relationships. Long Short-Term Memory units (LSTM) (Hochreiter and Schmidhuber, 1997) are one of the most used in Natural Language Processing (NLP). They are able to learn the dependencies in lengths of considerably large chains. Moreover, attention models have become an effective mechanism to obtain better results (Yang et al., 2017; Zhang et al., 2017; Wang et al., 2016; Lin et al., 2017; Rush et al., 2015). In (Yang et al., 2016), the authors use a hierarchical attention network for document classification. The model has two levels of attention mechanisms applied at the word and sentence-level, enabling it to attend differentially to more and less important content when constructing the document representation. The experiments show that the architecture outperforms previous methods by a substantial margin. In this paper, we propose a similar Attention-based LSTM for HaSpeeDe. The attention layer is applied on the top of a Bidirectional LSTM to generate a context vector for each word embedding which is then fed to another LSTM network to detect the presence or not of hate in the text. The paper is organized as follows. Section 2 describes our system.

[1]http://www.evalita.it/2018
[2]http://www.di.unito.it/tutreeb/haspeede-evalita18/index.html

Experimental results are then discussed in Section 3. Finally, we present our conclusions with a summary of our findings in Section 4.

2 System

2.1 Preprocessing

In the preprocessing step, the text is cleaned. Firstly, the emoticons are recognized and replaced by corresponding words that express the sentiment they convey. Also, all links and urls are removed. Afterwards, text is morphologically analyzed by FreeLing (Padró and Stanilovsky, 2012). In this way, for each resulting token, its lemma is assigned. Then, the texts are represented as vectors with a word embedding model. We used pretrained word vectors in Italian from fastText (Bojanowski et al., 2016).

2.2 Method

We propose a model that consists in a Bidirectional LSTM neural network (Bi-LSTM) at the word level as Figure 1 shows. At each time step t the Bi-LSTM gets as input a word vector x_t with syntactic and semantic information, known as word embedding (Mikolov et al., 2013). Afterward, an attention layer is applied over each hidden state $\hat{h_t}$. The attention weights are learned using the concatenation of the current hidden state h_t of the Bi-LSTM and the past hidden state s_{t-1} of the Post-Attention LSTM (Pos-Att-LSTM). Finally, the presence of hate (or not) in a text is predicted by this final Pos-Att-LSTM network.

2.3 Bidirectional LSTM

In NLP problems, standard LSTM receives sequentially (left to right order) at each time step a word embedding x_t and produces a hidden state h_t. Each hidden state h_t is calculated as follow:

$$
\begin{aligned}
input\_gate_t &= \sigma(W^{(i)}x_t + U^{(i)}h_{t-1} + b^{(i)}) \\
forget\_gate_t &= \sigma(W^{(f)}x_t + U^{(f)}h_{t-1} + b^{(f)}) \\
output\_gate_t &= \sigma(W^{(o)}x_t + U^{(i)}h_{t-1} + b^{(o)}) \\
new\_mem_t &= \sigma(W^{(u)}x_t + U^{(u)}h_{t-1} + b^{(u)}) \\
final\_mem_t &= i_t \otimes u_t + f_t \otimes c_{t-1} \\
h_t &= o_t \otimes tanh(c_t)
\end{aligned}
$$

Where all W_*, U_* and b_* are parameters to be learned during training. The function σ is the sigmoid function and $\otimes$ stands for element-wise multiplication.

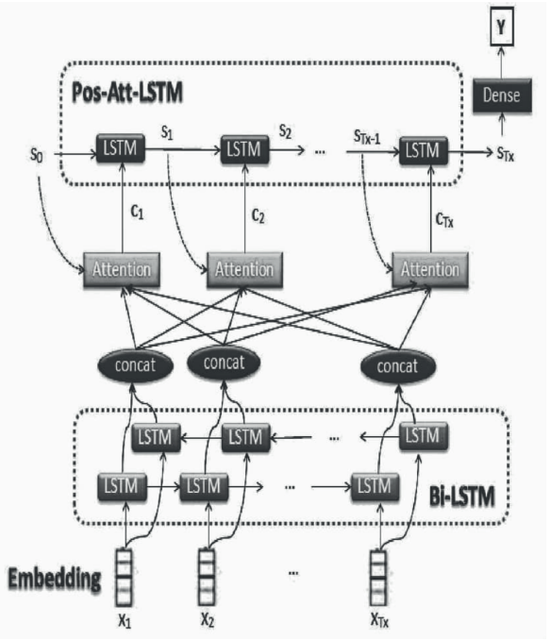


Figure 1: General architecture

The bidirectional LSTM, on the other hand, makes the same operations as standard LSTM but, processes the incoming text in a left-to-right and a right-to-left order in parallel. Thus, the output is a two hidden state at each time step $\overrightarrow{h_t}$ and $\overleftarrow{h_t}$.

The proposed method uses a Bidirectional LSTM network which considers each new hidden state as the concatenation of these two $\hat{h_t} = [\overrightarrow{h_t}, \overleftarrow{h_t}]$. The idea of this Bi-LSTM is to capture long-range and backwards dependencies.

2.4 Attention Layer

With an attention mechanism we allow the Bi-LSTM to decide which part of the sentence should "attend". Importantly, we let the model learn what to attend on the basis of the input sentence and what it has produced so far. Figure 2 shows the general attention mechanism.

Let $H \in R^{2*N_h \times T_x}$ the matrix of hidden states $[\hat{h_1}, \hat{h_2}, ..., \hat{h_{T_x}}]$ produced by the Bi-LSTM, where N_h is the size of the hidden state and T_x is the length of the given sentence. The goal is then to derive a context vector c_t that captures relevant information and feeds it as an input to the next level (Pos-Att-LSTM). Each c_t is calculate as follow:

$$c_t = \sum_{t'=1}^{T_x} \alpha_{t,t'} \hat{h_{t'}}$$

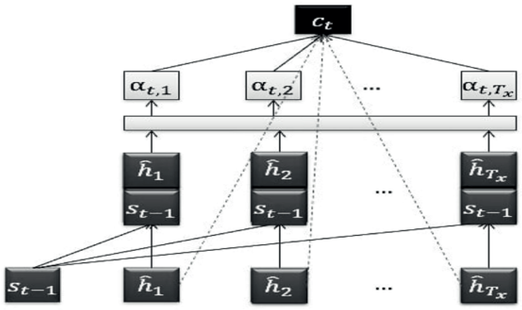


Figure 2: Attention layer

$$\alpha_{t,t'} = \frac{\beta_{t,t'}}{\sum_{i=1}^{T_x} \beta_{t,i}}$$

$$\beta_{t,t'} = tanh(W_a * [\hat{h_t}, s_{t-1}] + b_a)$$

Where W_a and b_a are the trainable attention weights, s_{t-1} is the past hidden state of the Pos-Att-LSTM and $\hat{h_t}$ is the current hidden state. The idea of the concatenation layer is to take into account not only the input sentence but also the past hidden state to produce the attention weights.

2.5 Post-Attention LSTM

The goal of the Post-Att-LSTM is to predict whether the text is hateful or not. This network at each time step receives the context vector c_t which is propagated until the final hidden state s_{T_x}. This vector is a high level representation of the text and is used in the final softmax layer as follow:

$$\hat{y} = softmax(W_g * s_{T_x} + b_g)$$

Where W_g and b_g are the parameters for the softmax layer. Finally, cross entropy is used as the loss function, which is defined as:

$$L = -\sum_i y_i * log(\hat{y_i})$$

y_i is the true classification of the i-th text.

3 Results

Table 1 shows the results obtained by different variants of the proposed method with the 5-fold cross-validation in terms of F1-score, precision and recall on the training set. The models are: M1 - LSTM+Att+LSTM (run1), M2 - LSTM+Att+LSTM (run2), M3 - Bi-LSTM+Att+LSTM (run1) and M4 - Bi-LSTM+Att+LSTM (run2).

	Twitter			Facebook		
	F1	P	R	F1	P	R
SVM	0.748	0.772	0.737	0.780	0.787	0.781
M1	0.869	0.881	0.863	0.865	0.872	0.863
M2	0.865	0.867	0.865	0.894	0.895	0.894
M3	0.853	0.860	0.854	0.864	0.873	0.864
M4	**0.877**	**0.891**	**0.871**	**0.899**	**0.903**	**0.899**

Table 1: 5-fold cross-validation results on the training corpus (Twitter and Facebook) in terms of F1-score (F1), Precision (P) and Recall (R). The best results are in bold. run2 in M2 and M4, identifies models that take dictionaries into account.

As run1 in M1 and M3, we first evaluated the model described before which is compound for the Bi-LSTM, the Attention layer and the LSTM (Bi-LSTM+Att+LSTM). Also, a variation in this model originated a new model for analizing the contribution of the Bi-LSTM layer. Therefore, we substituted the Bi-LSTM for a LSTM (LSTM+Att+LSTM).

Then, we processed the training sets to generate resources that we called the hate words dictionaries. For each train set we generated a dictionary of the most common words in the texts labeled as hateful. Taking into account this dictionaries, we added a linguistic characteristic to texts which defines if it contains a word into the correspondent dictionary. Thus, run 2 of the model is obtained considering this linguistic characteristic.

We used a SVM as baseline to compare the results of the different variants of the model and all variants achieved better results than this baseline.

The results show that the original model outperforms the results of the variant where the Bi-LSTM is not used. It is important to note that this occurs for run2 where the linguistic characteristic is taken into account. In fact, when this feature is not used the results decrease and the original model obtains the worst results in most cases. Therefore, taking into account the run2 of each variant, the results suggest that the best option is to use the Bi-LSTM with the linguistic characteristic.

The HaSpeeDe task was three sub-tasks, based on the dataset used. First, only the Facebook dataset could be used to classify the Facebook test set (HaSpeeDe-FB), where our system takes macro-average F1-score of 0.7147 and 0.7144, reaching the 11th and 10th positions for run1 and run2 of the model respectively. Another subtask

was HaSpeeDe-TW, here only the Twitter dataset can be used to classify the Twitter test set, where our system takes scores of 0.6638 and 0.6567, reaching the 12th and 13th positions for run1 and run2 of the model respectively. Finally, two other tasks consisted of using one of the datasets to train and the other to classify (Cross-HaSpeeDe). Here our system takes scores of 0.4544 and 0.5436, reaching places 10th and 7th in Cross-HaSpeeDe-FB and scores of 0.4451 and 0.318, for places 10th and 12th in Cross-HaSpeeDe-TW.

We think that these results can be improved with a more careful tunning of the model parameters. In addition, it may be necessary to enrich the system with linguistic resources for the treatment of the Italian language.

4 Conclusion

We propose an Attention-based Long Short-Term Memory Network Recurrent Neural Network for the EVALITA 2018 task on Hate Speech Detection (HaSpeeDe). The model consists of a bidirectional LSTM neural network with an attention mechanism that allows to estimate the importance of each word and then, this context vector is used with another LSTM model to estimate whether a text is hateful or not. The results showed that the use of a linguistic characteristic based on the occurrence of hateful words in the texts allows to improve the performance of the model. In addition, experiments performed on the training sets with 5-fold cross-validation suggest that the use of the Bi-LSTM layer is important when this linguistic characteristic is taken into account.

Acknowledgments

The work of the fourth author was partially supported by the SomEMBED TIN2015-71147-C2-1-P research project (MINECO/FEDER).

References

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Cristina Bosco, Felice Dell'Orletta, Fabio Poletto, Manuela Sanguinetti, and Maurizio Tesconi. 2018. Overview of the Evalita 2018 Hate Speech Detection Task. In Tommaso Caselli, Nicole Novielli, Viviana Patti, and Paolo Rosso, editors, *Proceedings of Sixth Evaluation Campaign of Natural Language Processing and Speech tools for Italian (EVALITA 2018)*, Turin, Italy. CEUR.org.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Kai Lin, Dazhen Lin, and Donglin Cao. 2017. Sentiment analysis model based on structure attention mechanism. In *UK Workshop on Computational Intelligence*, pages 17–27. Springer.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Lluís Padró and Evgeny Stanilovsky. 2012. Freeling 3.0: Towards wider multilinguality. In *LREC2012*.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685*.

Yequan Wang, Minlie Huang, Li Zhao, et al. 2016. Attention-based lstm for aspect-level aentiment classification. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 606–615.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489.

Min Yang, Wenting Tu, Jingxuan Wang, Fei Xu, and Xiaojun Chen. 2017. Attention based lstm for target dependent sentiment classification. In *AAAI*, pages 5013–5014.

Yu Zhang, Pengyuan Zhang, and Yonghong Yan. 2017. Attention-based lstm with multi-task learning for distant speech recognition. *Proc. Interspeech 2017*, pages 3857–3861.

Detecting Hate Speech for Italian Language in Social Media

Valentino Santucci, Stefania Spina
University for Foreigners of Perugia
{valentino.santucci, stefania.spina}@unistrapg.it

Alfredo Milani
University of Perugia
alfredo.milani@unipg.it

Giulio Biondi, Gabriele Di Bari
University of Florence
{giulio.biondi, gabriele.dibari}@unifi.it

Abstract

English. In this report we describe the hate speech detection system for the Italian language developed by a joint team of researchers from the two universities of Perugia (University for Foreigners of Perugia and University of Perugia). The experimental results obtained in the HaSpeeDe task of the Evalita 2018 evaluation campaign are analyzed. Finally, a suggestion for future research directions is provided in the conclusion.

Italiano. *In questo documento descriviamo il sistema di hate speech detection per la lingua Italiana sviluppato da una squadra di ricercatori dell'Università per Stranieri di Perugia e dell'Università degli Studi di Perugia. I risultati sperimentali ottenuti nel task HaSpeeDe, organizzato nell'ambito di Evalita 2018, sono riportati e analizzati. Infine, una possibile direzione di ricerca è fornita nelle conclusioni.*

1 Introduction

In the recent years there was an exponential growth of social media that has revolutionized communication and content publishing. However, social media are also increasingly exploited for the propagation of hate speech. This issue motivates the recent research on hate speech detection systems (Zhang and Luo, 2018; Waseem and Hovy, 2016; Del Vigna et al., 2017; Davidson et al., 2017; Badjatiya et al., 2017; Gitari et al., 2015).

In this paper, we provide the description of our hate speech detection system for the Italian language. The system, namely HSD4I_PG, has been developed by a joint team of researchers from the University for Foreigners of Perugia and the University of Perugia. The code of HSD4I_PG is provided online at the url https://github.com/Gabriele91/HSD4I_PG.

The rest of the paper is organized as follows. The main system architecture is provided in Section 2, while the single software components are described in Sections 3-6. Experimental results are provided in Section 7, while conclusion and future lines of research are depicted in Section 8.

2 Architecture of the Hate Speech Detector

The hate speech detector we have developed, namely HSD4I_PG, is composed by several software components:

- a *tokenizer* for Italian posts from social media,
- the popular *FastText* tool (Bojanowski et al., 2016) used to generate a word embedding model,
- a *features generator* that generates a vector of numeric features for each post to be classified,

- a (trainable) *classifier* that, for each post, predicts its class label.

Moreover, the following resources have been adopted:

- the *Ita_Twitter* corpus (Spina, 2016) that includes 1,234,865 tweets extracted from the Italian timeline in a time span of seven months (November 2012 - May 2013). The tweets were extracted randomly, 2,000 per day, using the R package TwitteR (https://cran.r-project.org/web/packages/twitteR/);
- the *Italian Lexicon of Hate Speech* that was collected based on an Italian monolingual dictionary, Il Nuovo De Mauro, which is also available online (https://dizionario.internazionale.it);
- the *Sentix* italian lexicon for sentiment analysis (Basile and Nissim, 2013);
- the training sets of 3,000 Facebook posts and 3,000 tweets available for the "Haspeede" task of Evalita 2018.

As any other supervised classifier system, HSD4I_PG requires a training stage, that is depicted in Figure 1. The word embedding model is trained by FastText using the Ita_twitter corpus. Numeric features are obtained by aggregating the FastText features and by generating some ad-hoc extra-features. These numeric features are finally fed to a Support Vector Machine (SVM) (Cortes and Vapnik, 1995) in order to generate a classifier model.

After the SVM classifier has been trained, the prediction of (unlabeled) posts is performed following the scheme depicted in Figure 2.

3 The Tokenizer

A tokenizer for the Italian language adopted in social media has been designed by modifying the output produced by the "TweetTokenizer" class of the popular Python library NLTK (Bird et al., 2009).

A variety of corrections have been introduced. The most important ones are:

1. two or more consecutive occurrences of the same vowel have been replaced by a single occurrence (e.g., "ciaooo" is replaced with "ciao"),
2. alternative spellings of some bad words have been normalized (e.g., "vaffa" is replaced with its most popular form),
3. some common mispellings and abbreviations have been corrected (e.g., "cmq" is replaced with "comunque"),
4. hashtags have been split into multiple tokens using the Python library "compound-word-splitter",
5. apostrophes have been considered as token separators,
6. tokens composed by digits characters have been replaced with the token `NUM`,
7. tokens corresponding to Twitter mentions have been replaced with the token `MEN`,
8. tokens corresponding to web links have been replaced with the token `URL`,
9. emojis have been kept as tokens on their own, while other punctuation characters have been removed,
10. all the textual tokens have been replaced with their stemmed form by using the NLTK implementation of the Snowball stemming algorithm for the Italian language (Porter, 1980).

Moreover, in order to provide additional experimental results, we have also tried a lighter variant of the tokenizer that only perform the tasks numbered from 5 to 10.

4 The Word Embedding Model

A word embedding model is generated by FastText (Bojanowski et al., 2016) using the skipgram technique.

Fed with the Ita_Twitter corpus, FastText produces a numeric vector representation for every n-gram contained in the corpus' posts in such a way that the n-grams belonging to tokens appearing in similar contexts are close to each other in the continuous numerical space.

After the model has been generated, a numeric representation for a given token w can be simply computed by summing up the numeric representations of the n-grams that compose w.

Since out-of-vocabulary words are quite common in social media texts, we think that the subwords information contained in the n-grams is particularly useful in our scenario.

5 The Features Generator

The word embedding model allows to generate a numeric representation for every token. Therefore, in order to produce a (constant length) numeric representation of the whole post, we need to aggregate the vectors corresponding to the tokens of the post. Six different aggregation functions have been considered: average (`avg`), standard deviation (`std`), minimum (`min`), maximum (`max`), median (`med`), and sum (`sum`). Any combination of these aggregators can be adopted, thus the features generator requires an experimental tuning (see Section 7).

Moreover, 20 additional extra-features have been introduced:

- number of hateful tokens, computed using the Italian Lexicon of Hate Speech (Spina, 2016),
- average sentiment polarity and intensity, computed using the Sentix lexicon (Basile and Nissim, 2013),
- number of web links,
- number of mentions,
- a boolean flag to indicate if it is a reply tweet or not,
- number of hashtags,
- maximum length of an hashtag (in characters),
- a boolean flag to indicate if it is a retweet or not,
- the percentage of capital letters,
- the percentage of tokens whose letters are all in capital case,
- number of exclamation marks,
- number of tokens composed by three or more dots,
- number of punctutation characters,
- number of emojis,
- number of repeated consecutive vowels,
- percentage of tokens representing a correct Italian word,
- post length in number of characters,
- post length in number of tokens.

As an illustrative example, let consider that: FastText has generated numeric vectors of size 300 for every single token w of a post p, and that the combination of the three aggregators `sum, min, max` has been chosen. Then, the numeric vector representing p has $300 \times 3 + 20 = 920$ dimensions and it is formed by concatenating the three vectors, each one of size 300, given by every chosen aggregator together with the 20 extra-features.

Finally, in the case the number of features is too large for the classifier, during the training phase we are able to reduce the dimensionality to a given number k by selecting the features having the largest mutual information with respect to the class labels.

6 The Classifier

After some preliminary experiments, we have decided to adopt a Support Vector Machine (SVM) classifier (Cortes and Vapnik, 1995). SVM is a supervised technique for training a classifier model by efficiently computing a separation hyperplane (between the two classes to be predicted) in a (implicitly) higher dimensional space (with respect to the features dimensionality). The SVM implementation of the Python's library Scikit-Learn (Pedregosa et al, 2011) has been used.

Compared to the popular neural network model, the SVM technique has less parameters to be tuned, it is computationally more efficient, and it generally obtains comparable performances.

Finally, it is important to note that, before the training phase, all the training features have been standardized in such a way that their means and variances, across all the training instance, are, respectively, 0 and 1.

7 Experiments

7.1 Experimental Setting

The parameters of the different software components of HSD4I_PG have been tuned using a grid search approach and a 10-folds cross-validation scheme.

FastText parameters have been chosen in the following ranges: number of epochs `epoch` $\in \{5, 20, 50, 100\}$, the initial learning rate `lr` $\in$

$\{0.05, 0.1\}$, the negative sampling `neg` $\in \{5, 20, 50\}$, the window size `ws` $\in \{5, 10\}$. Moreover, the skipgram model has been considered, while other FastText parameters that have been set to constant values are: `dim = 300`, `minCount` $= 1$, `minn` $= 3$, and `maxn` $= 6$.

Regarding the features generator (see Section 5), a combination of the six aggregators has to be chosen. Importantly, for combinations resulting in more than 1,000 features, the filtering procedure described at the end of Section 5 is performed.

After some preliminary experiments, we have decided to use the following ranges in order to tune the SVM parameters: `kernel` $\in \{$`rbf`, `linear`$\}$, `C` $\in \{1.8, 2, 2.2, 2.4\}$. Moreover, the `gamma` and `class_weight` parameters have been set to, respectively, `auto` and `balanced`.

The best parameter setting resulting from the experimental tuning is provided in Table 1.

	Parameter	Value
FastText	`epoch`	50
	`lr`	0.05
	`ns`	50
	`ws`	5
Features Generator	aggregators	`sum`
		`min`
		`max`
SVM	`kernel`	`rbf`
	`C`	2.2

Table 1: Tuned parameter setting

This setting has been used to generate the results submitted as "run 2" at the Haspeede task of Evalita 2018 by the team "Perugia1". For a mistake, we have submitted a wrong file as "run 1". Anyway, in the following section we also provide the results of three additional executions of HSD4I_PG:

Execution A) It uses the same setting of Table 1 except that $C = 2$,

Execution B) It uses the same setting of Table 1 except that the lighter variant of the tokenizer (see Section 3) has been adopted,

Execution C) It uses the same setting of Table 1 except that $C = 2$ and the lighter variant of the tokenizer (see Section 3) has been adopted.

7.2 Experimental Results

Table 2 provides the results obtained by HSD4I_PG in the four proposed tasks. In particular, the Macro-Average F1 score for each subtask is shown, along with the difference from the best competitor in the subtask.

SubTask	HSD4I_PG	Distance from best
HaSpeeDe-FB	0.7841	0.0447
HaSpeeDe-TW	0.7744	0.0249
Cross-HaSpeeDe-FB	0.6279	0.0262
Cross-HaSpeeDe-TW	0.5545	0.1440

Table 2: Subtask results of HSD4I_PG

Table 2 shows that HSD4I_PG achieved results comparable to the best competitors, except in the task Cross-HaSpeeDe-TW. The complete results for all the tasks are available in (Bosco et al., 2018). Besides, in Tables 3 and 4, three additional rows corresponding to the new executions A,B,C previously discussed (and performed after the official HaSpeeDe evaluation) are provided.

Interestingly, the results in Table 4 show that HSD4I_PG, tuned with different parameter settings, would have ranked 3rd in the HaSpeeDe-TW subtask (see (Bosco et al., 2018)).

8 Conclusion and Future Work

In this paper we have introduced a system for the hate speech detection of social media texts in Italian language. The results we have obtained for the HaSpeeDe task of the Evalita 2018 campaign are provided.

It is worth to point out that the results of most participants are very similar and quite far from being fully accurate. The question is whether hate annotation is objective or subjective. Few of the posts in the datasets looks to be difficult to annotate even for a human being. Indeed, we think that different people can produce different annotations. Therefore, it can be interesting to model the subjective perception of hatefulness and exploit such information in the detection task, perhaps, taking inspiration by recommender system techniques.

References

Pinkesh Badjatiya, Shashank Gupta, Manish Gupta, and Vasudeva Varma. 2017. Deep Learning for Hate Speech Detection in Tweets. In *Proceedings of the 26th International Conference on World Wide Web*

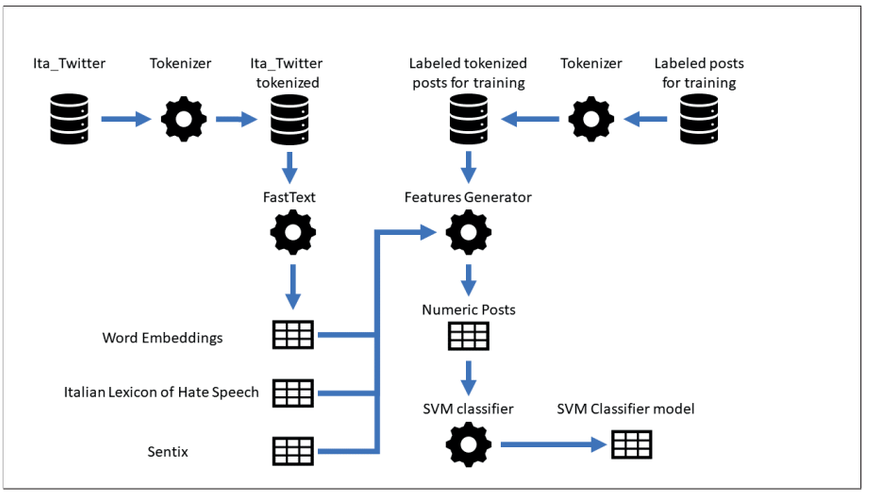


Figure 1: Training in HSD4I_PG

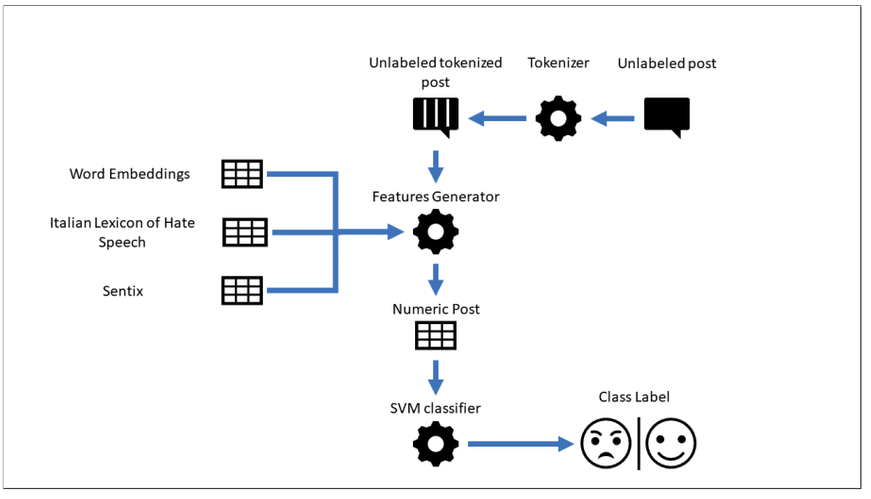


Figure 2: Classification in HSD4I_PG

	Not HS			HS			Macro-Avg F-score
	Precision	Recall	F-score	Precision	Recall	F-score	
A	**0.7261**	**0.6811**	**0.7029**	**0.8522**	**0.8774**	**0.8646**	**0.7838**
B	**0.7219**	**0.6749**	**0.6976**	**0.8496**	**0.8759**	**0.8625**	**0.7801**
C	**0.7166**	**0.6811**	**0.6984**	**0.8514**	**0.8715**	**0.8715**	**0.7799**

Table 3: Additional results in the subtask HaSpeeDe-FB

	Not HS			HS			Macro-Avg F-score
	Precision	Recall	F-score	Precision	Recall	F-score	
A	**0.8489**	**0.8728**	**0.8607**	**0.7180**	**0.6759**	**0.6963**	**0.7785**
B	**0.8545**	**0.8950**	**0.8743**	**0.7568**	**0.6821**	**0.7175**	**0.7959**
C	**0.8575**	**0.8905**	**0.8737**	**0.7517**	**0.6914**	**0.7203**	**0.7970**

Table 4: Additional results in the subtask HaSpeeDe-TW

Companion - WWW '17 Companion, pages 759–760, New York, New York, USA. ACM Press.

Valerio Basile and Malvina Nissim. 2013. Sentiment Analysis on Italian Tweets. In *In Proceedings of the 4th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis, Atlanta, Georgia, 14 June 2013*.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python*. O'Reilly Media, Inc., 1st edition.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching Word Vectors with Subword Information. 7.

Cristina Bosco, Felice Dell'Orletta, Fabio Poletto, Manuela Sanguinetti, and Maurizio Tesconi. 2018. Overview of the Evalita 2018 Hate Speech Detection Task. In Tommaso Caselli, Nicole Novielli, Viviana Patti, and Paolo Rosso, editors, *Proceedings of the 6th evaluation campaign of Natural Language Processing and Speech tools for Italian (EVALITA'18)*, Turin, Italy. CEUR.org.

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine Learning*, 20(3):273–297.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated Hate Speech Detection and the Problem of Offensive Language. 3.

Fabio Del Vigna, Andrea Cimino, Felice Dell'Orletta, Marinella Petrocchi, and Maurizio Tesconi. 2017. Hate me, hate me not: Hate speech detection on Facebook. In *CEUR Workshop Proceedings*.

Njagi Dennis Gitari, Zhang Zuping, Hanyurwimfura Damien, and Jun Long. 2015. A lexicon-based approach for hate speech detection. *International Journal of Multimedia and Ubiquitous Engineering*.

Fabian Pedregosa et al. 2011. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.*, 12:2825–2830.

M.F. Porter. 1980. An algorithm for suffix stripping. *Program*, 14(3):130–137, 3.

Stefania Spina. 2016. *Fiumi di parole. Discorso e grammatica delle conversazioni scritte in Twitter*. StreetLib, Loreto, Italy.

Zeerak Waseem and Dirk Hovy. 2016. Hateful Symbols or Hateful People? Predictive Features for Hate Speech Detection on Twitter. In *Proceedings of the NAACL Student Research Workshop*.

Ziqi Zhang and Lei Luo. 2018. Hate Speech Detection: A Solved Problem? The Challenging Case of Long Tail on Twitter. 2.

RuG @ EVALITA 2018: Hate Speech Detection In Italian Social Media

Xiaoyu Bai[*], **Flavio Merenda**[*∓], **Claudia Zaghi**[*], **Tommaso Caselli**[*], **Malvina Nissim**[*]
[*] Rikjuniversiteit Groningen, Groningen, The Netherlands
[∓] Università degli Studi di Salerno, Salerno, Italy
f.merenda|t.caselli|m.nissim@rug.nl x.bai.5|c.zaghi@student.rug.nl

Abstract

English. We describe the systems the RuG Team developed in the context of the Hate Speech Detection Task in Italian Social Media at EVALITA 2018. We submitted a total of eight runs, participating in all four subtasks. The best macro-F1 score in all subtasks was obtained by a Linear SVM, using hate-rich embeddings. Our best system obtains competitive results, by ranking 6th (out of 14) in HaSpeeDe-FB, 3rd (out of 15) in HaSpeeDe-TW, 8th (out of 13) in Cross-HaSpeeDe_FB, and 6th (out of 13) in Cross-HaSpeeDe_TW.

Italiano. *Illustriamo i dettagli dei due sistemi che il Team RuG ha sviluppato nell'ambito dell'esercizio di valutazione su riconoscimento di messagi d'odio in testi da Social Media per l'italiano. Abbiamo partecipato a tutti e quattro i sottotask, inviando un totale di otto predizioni. La migliore macro-F1, è ottenuta da un SVM che usa embedding polarizzati, costruiti sfruttando contenuto ricco di odio. Il nostro miglior sistema ha ottenuto dei risultati competitivi, classificandosi 6° (su 14) in HaSpeeDe-FB, 3° (su 15) in HaSpeeDe-TW, 8° (su 13) nel Cross-HaSpeeDe_FB, e 6° (su 13) in Cross-HaSpeeDe_TW.*

1 Introduction

The use of "bad" words and "bad" language has been the battleground for freedom of speech for centuries. The spread of Social Media platforms, and especially of micro-blog platforms (e.g. Facebook and Twitter), has favoured the growth of online hate speech. Social media sites and platforms have been urged to deal with and remove offensive and/or abusive content but the phenomenon is so pervasive that developing systems that automatically detect and classify offensive on-line content has become a pressing need (Bleich, 2014; Nobata et al., 2016; Kennedy et al., 2017).

The Natural Language Processing and Computational Social Science communities have been receptive to such urgency, and the automatic detection of abusive and/or offensive language, trolling, and cyberbulling (Waseem et al., 2017; Schmidt and Wiegand, 2017) has seen a growing interest. This has taken various forms: datasets in multiple languages[1], thematic workshops[2], and shared evaluation exercises, such as the GermEval 2018 Shared Task (Wiegand et al., 2018), and the SemEval 2019 Task 5: HateEval[3] and Task 6: OffensEval[4]. The EVALITA 2018 Hate Speech Detection task (haspeede)[5] (Bosco et al., 2018) also falls in the latter category, and focuses on the automatic identification of hate messages from Facebook comments and tweets in Italian. We participated in this shared task with two different models, exploiting the concept of *polarised embeddings* (Merenda et al., 2018). The details of our participation are the core of this paper. Code and outputs are available at https://github.com/tommasoc80/evalita2018-rug.

2 Task

The haspeede task derives from the harmonization process of originally separate annotation efforts from two research groups, converging onto a uniform label granularity (Del Vigna et al., 2017; Poletto et al., 2017; Sanguinetti et al., 2018). For details on the data see Section 3.1, and the task

[1] http://bit.ly/2RZUlKH
[2] https://sites.google.com/view/alw2018
[3] http://bit.ly/2EEC7Me
[4] http://bit.ly/2P7pTQ9
[5] http://di.unito.it/haspeedeevalita18

overview paper (Bosco et al., 2018).

The hate detection task is articulated in four binary (hate vs non-hate) sub-tasks, two in-domain, two cross-domain. The in-domain sub-tasks require training and test data to belong to the same text type, either Facebook (HaSpeeDe-FB) or Twitter (HaSpeeDe-TW), while the cross-domain sub-tasks require training on one text type and testing on the other: Facebook-Twitter (Cross-HaSpeeDe_FB) and Twitter-Facebook (Cross-HaSpeeDe_TW).

3 Data and Resources

All of our runs for all subtasks are based on supervised approaches, where data (and features) play a major role for the final results of a system. Furthermore, our contribution adopted a closed-task setting, i.e. we did not include any training data beyond what was provided within the task. We did however build enhanced distributed representations of words exploiting additional data (see Section 3.2). This section illustrates the datasets and language resources used in our submissions.

3.1 Resources Provided by the Organisers

The organizers provided a total of 6,000 labeled Italian messages for training, split as follows: 3,000 comments from Facebook, and 3,000 messages from Twitter. For test, they subsequently made available 1000 instances for each text type. Table 1 illustrates the distribution of the classes in the different text types both in training and test data. Note that the distribution of labels in the test data is unknown at developing time.

Table 1: Distribution of the labeled samples in the training and test data per text type.

Text type	Class	Training	Test
Facebook	non-hate	1,618	323
	hate	1,382	677
Twitter	non-hate	2,028	676
	hate	972	324

Although the task organisers have balanced the datasets with respect to size, and have adopted the same annotation granularity (hate vs. non-hate), the two datasets are very different both in terms of class distribution (i.e. 46.06% of messages labelled as hateful in Facebook vs. 32.40% in Twitter in training) and with regard to their contents. For instance, the Facebook data is concerned with *general topics* that may contain hateful messages such as immigration, religion, politics, gender issues, while the Twitter dataset is focused on *specific targets*, i.e., categories or groups of individuals who are likely to become victims of hate speech (migrants, Muslims, and Roma[6]). It is also interesting to note that the label distribution in the Facebook test data is flipped compared to training, with a strong majority of hateful comments.

3.2 Additional Resources: Source-Driven Embeddings

We addressed the task by adopting a closed-task setting. However, as a strategy to potentially increase the generalization capabilities of our systems and tune them towards better recognition of hate content, we developed hate- and offense-sensitive word embeddings.

To do so, we scraped comments from a list of selected Facebook pages likely to contain offensive and/or hateful content in the form of comments to posts, extracting over 1M comments. We built word embeddings over the acquired data with the `word2vec` tool skip-gram model (Mikolov et al., 2013), using 300 dimensions, a context window of 5, and minimum frequency 1. In the remainder of this paper we refer to these representations as "hate-rich embeddings". More details on the creation process, including the complete list of Facebook pages used, and a preliminary evaluation of these specialised representations can be found in (Merenda et al., 2018).

4 Systems and Runs

We detail in this section our final submissions. The models have been developed in parallel to our participating systems at the GermEval 2018 Shared Task (Bai et al., 2018), sharing with them some core aspects.

4.1 Run 1: Binary SVM

Our first model is a Linear Support Vector Machine (SVM), built using the LinearSVC scikit learn implementation (Pedregosa et al., 2011).

We performed minimal pre-processing by removing stop words using the Python module `stop-words`[7], and lowercasing the tokens.

[6]The Romani, Romany, or Roma are an ethnic group of traditionally itinerant people who originated in northern India and are nowadays subject to ethnic discrimination.

[7]`https://pypi.org/project/stop-words/`

We used two groups of surface features, namely: i.) word n-grams in the range 1–3; and ii.) character n-grams in the range 2–4. The sparse vector representation of each (training) instance is then concatenated with its dense vector representation, as follows: for every word *w* in an instance *i*, we derived a 300 dimension representation, $\vec{w}$, by means of a look-up in the hate-rich embeddings. We performed max pooling over these word embeddings, $\vec{w}$, to obtain a 300 dimension representation of the full instance, $\vec{i}$. Words not covered in the hate-oriented embeddings are ignored. Finally, class weights are balanced and SVM parameters use default values ($C = 1$).

4.2 Run 2: Binary Ensemble Model

Our second submission uses a binary ensemble model, which combines a Convolutional Neural Network (CNN) system and the linear SVM (Section 4.1), with a logistic regression meta-classifier on top. Predictions on training data are obtained via ten-fold cross-validation.

In the ensemble model, each input instance to the meta-classifier is represented by the concatenation of four features: a) the class predictions for that instance made by the SVM, b) the predictions of the CNN, and c) two additional surface-level features: the instance's length in terms of characters and the percentage of offensive terms in the instance. This latter feature is obtained via a look-up in a list of offensive terms in Italian obtained from the article *Le Parole per ferire* by Tullio De Mauro[8] and the "bad words" category in the Italian Wiktionary. The feature is expressed by the ratio between the frequency of any of the instance's tokens comprised in the list and the instance's length in terms of tokens. Figure 1 shows the features fed to the ensemble meta-classifier.

The CNN is an adaptation of available architectures for sentence classification (Kim, 2014; Zhang and Wallace, 2015), using Keras (Chollet and others, 2015), and is composed of: i.) a word embeddings input layer using the hate-rich embeddings; ii.) a single convolutional layer; iii.) a single max-pooling layer; iv.) a single fully-connected layer; and v.) a sigmoid output layer.

The max-pooling layer output is flattened, concatenated, and fed to the fully-connected layer composed of 50 hidden-units with the ReLU activation function. The final output layer with the sigmoid activation function computes the distribution of the two labels. (Other network hyperparameters: `Number of filters`: 6; `Filter sizes`: 3, 5, 8; `Strides`: 1). We used binary cross-entropy as loss function and Adam as optimiser. In training, we set a batch size of 64 and ran it for 10 epochs. We also applied two dropouts: 0.6 between the embeddings and the convolutional layer, and 0.8 between the max-pooling and the fully-connected layer.

[8] `https://bit.ly/2J4TPag`

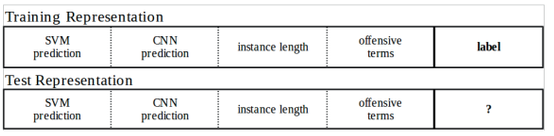


Figure 1: Feature representation of each sample fed to the ensemble model. On top, the representation of a training sample, on bottom, the representation of a test sample.

5 Results and Ranking

Table 2 reports the results and ranking for our runs for all four subtasks. We also include the scores of the CNN (not submitted to the official competition), marked with a $*$.[9]

Table 2: System results and ranking, including the out-of-competition runs for CNN alone.

Subtask	Model[10]	Rank	Macro F1
HaSpeeDe-FB	SVM	6/14	**0.7751**
	Ensemble	9/14	0.7428
	CNN*	n/a	0.7138
HaSpeeDe-TW	SVM	3/15	**0.7934**
	Ensemble	9/15	0.7530
	CNN*	n/a	0.7363
Cross-HaSpeeDe_FB	SVM	8/13	**0.5409**
	Ensemble	9/13	0.4845
	CNN*	n/a	0.4692
Cross-HaSpeeDe_TW	SVM	6/13	**0.6021**
	Ensemble	7/13	0.5545
	CNN*	n/a	0.6093

The SVM models obtain, by far, better results than the Ensemble models. It is likely that the Ensemble systems suffer from the lower performances of

[9] Being allowed to submit a maximum of two runs per subtask, we based our choice of models on the results of a 10-fold cross validation of the three architectures on the training data.

[10] The SVM correposnds to run id 1 and the Ensemble model to run id 3 in the official submitted runs - see Submissions-Haspeede in the GitHub repository `https://github.com/tommasoc80/evalita2018-rug/tree/master/Submissions-Haspeede`

the CNN. We also observe differences in performance on the two datasets across the subtasks.

Table 3: SVM's performance per class

Subtask	non-hate		hate	
	P	R	P	R
HaSpeeDe-FB	0.6990	0.6904	0.8531	0.8581
HaSpeeDe-TW	0.8577	0.8831	0.7401	0.6944
CrossHaSpeeDe_FB	0.8318	0.4023	0.3997	0.8302
CrossHaSpeeDe_TW	0.4375	0.6934	0.7971	0.5745

In-domain, in absolute terms, we do better on Twitter (.7934) than on Facebook (.7751), and this is even truer in relative terms, as performance overall in the competition is better on Facebook (best: 0.8288) than on Twitter (best: 0.7993). Our high score on HaSpeeDe-TW comes from high precision and recall on non-hate, while for HaSpeeDe-FB, we do well on the hate class. This can be due to label distribution (hate is always minority class, but more balanced in Facebook), but also to the fact that we use Facebook-based hate-rich embeddings, which might push towards better hate detection.

Cross-domain, results are globally lower, as expected, with best scores on Cross-HaSpeeDe_FB and Cross-HaSpeeDe_TW of 0.6541 and 0.6985, respectively (Bosco et al., 2018). Our models experience a more substantial loss when trained on Facebook and tested on Twitter (in Cross-HaSpeeDe_FB we lose over 25 percentage points compared to HaSpeeDe-TW, where the Twitter test set is the same), than viceversa (we lose ca. 17 percentage points on the Facebook test set).

6 Discussion

The drop in performance in the cross-domain settings is likely due to topics, and data collection strategies (general topics on Facebook, specific targets on Twitter). In other words, despite the use of hate-rich embeddings as a strategy to make the systems generalize better, our models remain too sensitive to training data, which is strongly represented as word and character n-grams.

The impact of the hate-rich embeddings is most strongly seen in HaSpeeDe-FB and Cross-HaSpeeDe_FB, with recall for the hate class being substantially higher than for the non-hate class. This could be due to the fact that the hate-rich embeddings have been generated from comments in Facebook pages, that is, the same text type as the training data in the two tasks, so that possibly some jargon and topics are shared. While this has a positive effect when training and testing on Facebook (HaSpeeDe-FB), it has instead a detrimental effect when testing on Twittter (Cross-HaSpeeDe_FB), since this dataset has a large majority of non-hate instances, and we tend to overpredict the hate class (see Table 3).

In HaSpeeDe-TW and Cross-HaSpeeDe_TW (training on Twitter) the impact of the hate-rich embeddings is a lot less clear. Indeed, recall for the hate class is always lower than non-hate, with the large majority of errors (more than 50% in all runs) being hate messages wrongly classified as non-hateful, thus seemingly just following the class imbalance of the Twitter trainset.

In both datasets, hate content is expressed either in a direct way, by means of "bad words" or direct insults to the target(s), or more implicitly and subtly. This latter type of hate messages is definitely the main source of errors for our systems in all subtasks. Finally, we observe that in some cases the annotation of messages as hateful is subject to disagreement and debate. For instance, all messages containing the word *rivoluzione* [revolution] are marked as hateful, even though there is a lack of linguistic evidence.

7 Conclusion and Future Work

Developing our systems for the Hate Speech Detection in Italian Social Media task at EVALITA 2018, we focused on the generation of distributed representations of text that could not only enhance the generalisation power of the models, but also better capture the meaning of words in hate-rich contexts of use. We did so exploiting Facebook on-line communities to generate *hate-rich embeddings* (Merenda et al., 2018).

A Linear SVM system outperformed a meta-classifer that used predictions from the SVM itself, and a CNN, due to the low performance of the CNN component. Major errors of the systems are due to implicit hate messages, where even the hate-rich embeddings fail. A further aspect to consider in this task is the difference in text type and class balance of the two datasets. Both of these aspects have a major impact on system performance in the cross-genre settings.

Finally, to better generalize to unseen data and genres, future work will focus on developing systems able to further abstract from the actual lexical content of the messages by capturing general

writing patterns of haters. One avenue to explore in this respect is "bleaching" text (van der Goot et al., 2018), a newly suggested technique used to fade the actual strings into more abstract, signal-preserving representations of tokens.

References

Xiaoyu Bai, Flavio Merenda, Claudia Zaghi, Tommaso Caselli, and Malvina Nissim. 2018. RuG at GermEval: Detecting Offensive Speech in German Social Media. In Josef Ruppenhofer, Melanie Siegel, and Michael Wiegand, editors, *Proceedings of the GermEval 2018 Workshop*.

Erik Bleich. 2014. Freedom of expression versus racist hate speech: Explaining differences between high court regulations in the usa and europe. *Journal of Ethnic and Migration Studies*, 40(2):283–300.

Cristina Bosco, Fabio Poletto Dell'Orletta, Felice, Manuela Sanuguinetti, and Maurizio Tesconi. 2018. Overview of the EVALITA Hate Speech Detection Task. In Tommaso Caselli, Nicole Novielli, Viviana Patti, and Paolo Rosso, editors, *Proceedings of the 6th evaluation campaign of Natural Language Processing and Speech tools for Italian (EVALITA'18)*, Turin, Italy. CEUR.org.

François Chollet et al. 2015. Keras. `https://keras.io`.

Fabio Del Vigna, Andrea Cimino, Felice Dell'Orletta, Marinella Petrocchi, and Maurizio Tesconi. 2017. Hate me, hate me not: Hate speech detection on facebook. In *Proceedings of the First Italian Conference on Cybersecurity (ITASEC17), Venice, Italy, January 17-20, 2017*, pages 86–95.

George Kennedy, Andrew McCollough, Edward Dixon, Alexei Bastidas, John Ryan, Chris Loo, and Saurav Sahay. 2017. Technology solutions to combat online harassment. In *Proceedings of the First Workshop on Abusive Language Online*, pages 73–77.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Flavio Merenda, Claudia Zaghi, Tommaso Caselli, and Malvina Nissim. 2018. Source-driven Representations for Hate Speech Detection, proceedings of the 5th italian conference on computational linguistics (clic-it 2018). Turin, Italy.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Chikashi Nobata, Joel Tetreault, Achint Thomas, Yashar Mehdad, and Yi Chang. 2016. Abusive language detection in online user content. In *Proceedings of the 25th International Conference on World Wide Web*, pages 145–153. International World Wide Web Conferences Steering Committee.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Fabio Poletto, Marco Stranisci, Manuela Sanguinetti, Viviana Patti, and Cristina Bosco. 2017. Hate speech annotation: Analysis of an italian twitter corpus. In *CEUR WORKSHOP PROCEEDINGS*, volume 2006, pages 1–6. CEUR-WS.

Manuela Sanguinetti, Fabio Poletto, Cristina Bosco, Viviana Patti, and Marco Stranisci. 2018. An Italian Twitter Corpus of Hate Speech against Immigrants. In Nicoletta Calzolari (Conference chair), Khalid Choukri, Christopher Cieri, Thierry Declerck, Sara Goggi, Koiti Hasida, Hitoshi Isahara, Bente Maegaard, Joseph Mariani, Hlne Mazo, Asuncion Moreno, Jan Odijk, Stelios Piperidis, and Takenobu Tokunaga, editors, *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan, May 7-12, 2018. European Language Resources Association (ELRA).

Anna Schmidt and Michael Wiegand. 2017. A survey on hate speech detection using natural language processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media. Association for Computational Linguistics, Valencia, Spain*, pages 1–10.

Rob van der Goot, Nikola Ljubešić, Ian Matroos, Malvina Nissim, and Barbara Plank. 2018. Bleaching text: Abstract features for cross-lingual gender prediction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 383–389.

Zeerak Waseem, Thomas Davidson, Dana Warmsley, and Ingmar Weber. 2017. Understanding abuse: A typology of abusive language detection subtasks. In *Proceedings of the First Workshop on Abusive Language Online*, pages 78–84, Vancouver, BC, Canada, August. Association for Computational Linguistics.

Michael Wiegand, Melanie Siegel, and Josef Ruppenhofer. 2018. Overview. In Josef Ruppenhofer, Melanie Siegel, and Michael Wiegand, editors, *Proceedings of the GermEval 2018 Workshop*.

Ye Zhang and Byron Wallace. 2015. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *arXiv preprint arXiv:1510.03820*.

Text analysis for hate speech detection in Italian messages on Twitter and Facebook

Giulio Bianchini
University of Perugia
Italy
giulio.bianchini@studenti.unipg.it

Lorenzo Ferri
University of Perugia
Italy
lorenzo.ferri@studenti.unipg.it

Tommaso Giorni
University of Perugia
Italy
tommaso.giorni@studenti.unipg.it

Abstract

English. In this paper, we present a system able to classify hate speeches in Italian messages from Facebook and Twitter platforms. The system combines several typical techniques from Natural Language Processing with a classifier based on Artificial Neural Networks. It has been trained and tested on a corpus of 3000 messages from the Twitter platform and 3000 messages from the Facebook platform. The system has been submitted to the HaSpeeDe task within the EVALITA 2018 competition and the experimental results obtained in the evaluation phase of the competition are presented and discussed.

Italiano. *In questo documento presentiamo un sistema in grado di classificare messaggi di incitamento all'odio in lingua italiana presi dalle piattaforme Facebook e Twitter. Il sistema combina diverse tecniche tipiche del Natural Language Processing con un classificatore basato su una Rete Neurale Artificiale. Quest'ultimo stato allenato e testato con un corpus di 3000 messaggi presi dalla piattaforma Twitter e 3000 messaggi presi dalla piattaforma Facebook. Il sistema stato sottomesso al task HaSpeeDe relativo alla competizione EVALITA 2018, e sono presentati e discussi i risultati sperimentali ottenuti nella fase di valutazione della competizione.*

1 Introduction

In the last years, social networks have revolutionized in a radical way the world of communication and the publication of contents. However, if on one hand social networks represent an instrument of freedom of expression and connection, on the other hand they are used for propagation and incitement to hatred. For this reason, recently, many softwares and technologies have been developed to reduce this phenomenon (Zhang and Luo, 2018) (Waseem and Hovy, 2016) (Del Vigna et al., 2017) (Davidson et al., 2017) (Badjatiya et al., 2017) (Gitari et al., 2015).

Specifically, approaches based on machine learning and deep learning are used by large companies to stem and stop this widespread fact. Despite the efforts spent to produce systems for the English language, there are very few resources for Italian (Del Vigna et al., 2017). In order to bridge this gap, a specific task (Bosco et al., 2018) for the detection of hateful contents has been proposed within the context of EVALITA 2018, the 6th evaluation campaign of Natural Language Processing and Speech tools for Italian. The EVALITA team provided the participants with the initial starting data sets, each consisting of 3000 classified comments taken respectively from Facebook and Twitter pages. The objective of the competition is to produce systems able to automatically annotate messages with boolean values (1 for message containing Hate Speech, 0 otherwise).

In this paper we describe the system submitted by the Vulpecula team. The system works in four phases: preprocessing of the initial dataset; encoding of the preprocessed dataset; training of the Machine Learning model; testing of the trained model. In the first phase, the comments were cleaned by applying text analysis techniques and some features have been extrapolated from these; then in the second phase, using a trained Word2Vec model (Mikolov et al., 2013), the comments were coded in a vector of 256 real num-

bers. In the third phase, an artificial neural network model was trained using the encoded comments as input along with their respective extra features. In order to have better and reliable results, to train and evaluate the model a cross-validation was used. In addition, together with accuracy and evaluation of the error, the F-measure were used to evaluate the quality of the model. Finally, in the fourth and last phase, the test set comments provided by EVALITA were classified. The rest of the paper is organized as follows. A system overview is provided in Section 2, while some details about the system components and the external tools used are provided in Section 3. Experimental results are shown and discussed in Section 7, while some conclusions and ideas for future works are depicted in Section 8.

2 System overview

The system has a structure similar to (Castellini et al., 2017); it has been organized into four main phases: preprocessing, encondig, training, testing, as also shown in Figure 1.

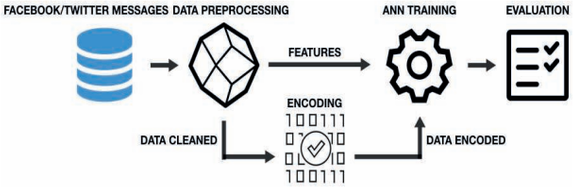


Figure 1: System architecture.

- In the first phase the corpus of 3000 Facebook comments, and the corpus of 3000 Twitter comments are cleaned and prepared to be encoded. In parallel within the cleaning we have extrapolated some interesting features for each comment. The entire phase is explained in details in section 4 and 5.

- In the second phase we trained a Word2Vec model, starting from 200k comments we download from some Facebook pages known to contain hate messages. Each of the initial data set comment has been encoded in a vector of real values by submitting it to the Word2Vec model. This phase is explained in details in the section 5.

- In the third phase we trained a multi-layer feed-forward neural network using the 3000 encoded comments and the respective features we extracted in the first phase. The description of the ANN is in the section 6.

- In the last phase the test set comments provided by EVALITA were classified and we joined the competition. This phase is explained in details in the section 7.

The source code of the project is provided online[1].

3 Tools Used

The entire project was developed using the Python programming language, for which several libraries are available and usable for the purpose of the project. Specifically, the following libraries were used for the preprocessing phase of the dataset:

- nltk: toolkit for natural language processing;
- unicode_emoji: library for the recognition and translation of emoticons;
- treetaggerwrapper: library for lemming and word tagging;
- textblob: another library for natural language processing;
- gensim: library that contains word2vec;
- sequence matcher: library for calculating the spelling distance between words;

For the training phase of the ML model the following libraries were used:

- keras (Chollet and others, 2015): High-level neural network API;
- sklearn (Pedregosa et al., 2011): Simple and efficient tools for data mining and data analysis ;

Finally, some corpora have been used:

- SentiWordNet (Baccianella et al., 2010);
- dataset of badwords, provided by Prof. Spina and research group of the University for Foreigners of Perugia;
- dataset of italian words;

[1] https://github.com/VulpeculaTeam/Hate-Speech-Detection

- dataset of 220k comments downloaded from Facebook pages (*Italia agli italiani stop ai clandestini, matteo renzi official, matteo salvini official, noiconsalvini, politici corrotti*)

4 Preprocessing

In the Knowledge Discovery in Databases (KDD) one of the crucial phases is data preparation. In this project this phase was tackled and the comments given to us were processed and prepared. Specific text analysis techniques have been applied in order to prepare the data in the best possible way in order to extract the most important information from them. All the operations performed for the data cleaning and for the extra-feature extraction are listed below. Each operation is iterated for all the 3000 comments of the data set.

- Extraction of the first feature: length of the comment.

- Extraction of the second feature: percentage of words written in CAPS-LOCK inside the comment. Calculated by the number of words written in CAPS-LOCK divided by the number of words in the comment.

- Replace the characters '&', '@' respectively in the letters 'e', 'a'.

- Conversion of disguised bad words. An interesting function added to the preprocessing is the recognition of censored bad-words, i.e. bad-words where some of their middle letters are replaced by special character (symbol, punctuation...) to make it recognizable by an human but not by a computer. At this scope we don't use a large vocabulary but it's better a simple list of most common bad-words censored (because only a small group of bad words is commonly censored). At this python function we pass an entire sentence creating a list splitting this by space. We scan the list of sentence words and we control if the first and last characters are letters and not number or symbols. Then we take this word without first and last letters and control if this middle sub-word is formed by special symbols/punctuation or by letter x (because "x" is often used for hiding bad-words). If yes, this middle sub-word is deleted from the censored bad-word, taking the top and end part of this formed by letters. At the end we scan the list of bad-words and we control if this top and end part matching with one of this scanned bad-words. If yes, this is replaced by the real word.

- Hashtag splitting. One of the most difficult cleaning phases is the Hashtag Splitting. For this we used a large dictionary of italian words in .csv format. First, we scan every word in this file and we control if these word is in the hashtag and then (for convenience we avoid the words of lenght 2) saving it in a list. In this phase will be taken also useless words not contextualized to the hashtag, so we will need to filter them. For this, first we sort all found words in decreasing length and we scan the list. So, starting to the first word on, we delete it from the hashtag. In this way the useless words in the list contained in larger words are found, saved in another list, and deleted from the beginning list containing all the words (both useful and useless) in the hashtag. In the final phase for each word in the resulting list we find its position within the hashtag and with this we create the real sentence, separating every word with a space.

- Removal of all the links from the comment.

- Editing of each word in the comment by this way: removal of nearby equal vowels, removal of nearby equal consonants if they are more than 2. Examples: from *"caaaaane"* to *"cane"*, from *"gallllina"* to *"gallina"*.

- Extraction of the third feature: number of sentences inside the comment. By sentence we mean a list of words that ends with *'.'* or *'?'* or *'!'*.

- Extraction of the fourth feature: number of *'?'* or *'!'* inside the comment.

- Extraction of the fifth feature: number of *'.'* or *','* inside the comment.

- Punctuation removal.

- Translation of emoticons (for Twitter messages). Given the large presence of emoticons in Twitter messages, it was decided to

translate the emoticons with the respective English translations. To do this, each sentence is scanned and if there are emoticons, these are translated into their corresponding meaning in English. Using the library unicode_emoji,

- Emoticon removal.
- Replacement of the abbreviations with the respective words, using a list of abbreviations created by ourselves.
- Removal of articles, pronouns, prepositions, conjunctions and numbers.
- Removal of the laughs.
- Replacement of accented characters with their unaccented characters.
- Lemmatization of each comment with the treetaggerwrapper library.
- Extraction of the sixth feature : polarity of the message. This feature is compute using the SentiWordNet corpora and his APIs. Since SentiWordNet was created to find the polarity of sentences in English, each message is translated using TextBlob in English and the polarity is then calculated.
- Extraction of the seventh feature : Percentage of spelling errors in the comment. To calculate a spelling error a word is compared with all the words of the Italian Vocabulary corpora; if the word is not present in the corpora there is a spelling error. Calculated by the number of spelling error divided by the number of words in the comment.
- Replacement of spelling error: In parallel with the previous step every spelling error is replaced with the most similar word in the Italian Vocabulary corpora. The similarity between the wrong word and all the other is calculated using a function of SequenceMatcher library. The wrong word is replaced with the most similar word in Italian Vocabulary corpora.
- Extraction of the eighth feature: number of bad words in the comment. Every word in the comment is compared with all the word in the Bad Words corpora; if the word is in the corpora it's a bad word.
- Extraction of the ninth feature: percentage of bad words. Calculated by the number of bad words divided by the number of words in the comment.
- Extraction of the tenth feature : Polarity TextBlob. This value is compute using a TextBlob function that allows to calculate the polarity. Also in this case the message is translated into English.
- Extraction of the final feature : Subjectivity TextBlob. Another value computed with a function in TextBlob.

5 Word Embeddings with Word2Vec

Very briefly, Word Embedding turns text into numbers. This transformation is necessary because many Machine Learning algorithms don't work with plain text but they require vectors of continuous values. Word Embedding has fundamental advantages in particular, it is a more efficient representation (dimensionality reduction) and also it is a more expressive representation (contextual similarity). So we have created a Word2Vec model for word embedding. For the training of the model, 200k messages were downloaded from several Facebook pages. These messages were preprocessed as explained in the previous section 4 and (in addiction with the messages provided by EVALITA's team) were used to train the Word2Vec model. The trained model encode each word in a vector of 128 real numbers. Each sentence is instead encoded with a vector of 256 real numbers divided into two components of 128 elements: the first component is the vector sum of the coding of each word in the sentence, while the second component is the arithmetic mean. At this point each of the 3000 comments of the starting training set is a vector of 265 reals: 256 for the coding of the sentence and 9 for the previously calculated features.

6 Model training

The vectors obtained by the process described in section 5 were used as input for training an Artificial Neural Network - ANN. (Russell and Norvig, 2016) The Articial Neural Network mathematical model composed of artificial "neurons", vaguely inspired by the simplification of a biological neural network. There are different types of ANN, the one used in this research is a feed-forward: this

means that the connections between nodes do not form cycles as opposed to recurrent neural networks. In this neural network, the information moves only in one direction, ahead, with respect to input nodes, through hidden nodes (if existing) up to the exit nodes. In the class of feed-forward networks there is the multilayer perceptron one. The network we have built is made up of two hidden layers in which the first layer consists of 128 nodes and the second one is 56. The last layer is the output one and is formed by 2 nodes. The activation functions for the respective levels are sigmoid, relu and softmax and the chosen optimizer is Adagrad, each layer has a dropout of 0.45. The reason why these parameters have been chosen is because after having tried countless configurations, the best results during the training phase have been obtained with these parameters. In particular, have been tried all the possible combinations of these parameters:

- **Number of nodes** of the hidden layers: 56, 128, 256, 512;
- **Activation function** of the hidden layers: sigmoid, relu, tanh, softplus;
- **Optimizer**: Adagrad, RMSProp, Adam.

Furthermore, the dropout was essential to prevent over-fitting. In fact, dropout consists to not consider neurons during the training phase of certain set of neurons which is chosen randomly. The dropout rate is set to 45%, meaning that the 45% of the inputs will be randomly excluded from each update cycle. As methods of estimation, cross-validation was used, partitioning the data into 10 disjoint subsets. As metrics for performance evaluation, the goodness of the model was analyzed by calculating True Positive, True Negative, False Positive and False Negative. From these the cost-sensitive measures precision, recall and f-score were calculated. These are the best results achieved with the training dataset of Facebook comments obtained during the cross validation:

- **Accuracy**: 83.73%;
- **Standard deviation**: 1.09;
- **True Positive**: 1455;
- **True Negative**: 1057;
- **False Positive**: 163;
- **False Negative**: 325;
- **Precision**: 0.899%;
- **Recall**: 0.817%;
- **F1-Score**: 0.856%;
- **F1-Score_Macro**: 0.856%;

7 Experimental Results

After the release of the unlabelled test set, the new 2000 messages (1000 of them from Facebook and 1000 of them from Twitter) were cleaned as explained in section 4 and the respective features were extrapolated. Then, these new comments were added to the comment's pool used to create the Word2Vec model, and a new Word2Vec model was created with the new pool. Finally, the 2000 comments were encoded as previously explained in section 5 in the 265 component vectors and these were the input of the neural network that classified them. From the training phase, two neural network models were built: one trained with the dataset of 3000 Facebook messages and the other trained with the dataset of 3000 Twitter messages. We call the first model **VTfb** and the second one **VTtw**. EVALITA's task consisted in four sub-tasks that were:

- **HaSpeeDe-FB**: test VTfb with the 1000 messages taken from Facebook;
- **HaSpeeDe-TW**: test VTtw with the 1000 messages taken from Twitter;
- **Cross-HaSpeeDe-FB**: test VTfb with the 1000 messages taken from Facebook;
- **Cross-HaSpeeDe-TW**: test VTtw with the 1000 messages taken from Twitter;

Sub-task	Model	F1	Distance
HaSpeeDe-FB	VTfb	0.7554	0.0734
HaSpeeDe-TW	VTtw	0.7783	0.021
Cross-HaSpeeDe-FB	VTfb	0.6189	0.0089
Cross-HaSpeeDe-TW	VTtw	0.6547	0.0438

Table 1: Team results in the HaSpeeDe sub-tasks.

In Table 1 we report the Macro-Average F1 score for each sub-task together with the differences

with the best result obtained in the competition (column "Distance" in the table). Compared with the results we had in the training phases (section 6), we would have expected better results in the HaSpeede-FB task. However, our system appears to be more general and not specifically targeted to a platform, in fact the differences in the other tasks are minimal.

8 Conclusion and Future Work

In this paper we presented a system based on neural networks for the hate speech detection in social media messages in Italian language. Recognizing negative comments is not easy, as the concept of negativity is often subjective. However, good results have been achieved that are not so far from the results obtained by the best within the competition. The proposed system can certainly be improved, an idea can be to use clustering techniques to categorize the messages (cleaned and with the related features) in two subgroups (positive and negative) and then, for each comment, calculate how much this is more similar to negative comments or positive comments and add it as a feature.

Acknowledgments

The authors would like to thank prof. Valentina Poggioni who has helped and supported us in the development of the whole project. A special thanks to Manuela Sanguinetti, our shepherd in EVALITA competition for all the support she has given to us.

References

Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. 2010. Sentiwordnet 3.0: an enhanced lexical resource for sentiment analysis and opinion mining. In *Proceedings of Lrec 2010*, pages 2200–2204.

Pinkesh Badjatiya, Shashank Gupta, Manish Gupta, and Vasudeva Varma. 2017. Deep learning for hate speech detection in tweets. In *Proceedings of the 26th International Conference on World Wide Web Companion*, pages 759–760. International World Wide Web Conferences Steering Committee.

Cristina Bosco, Felice Dell'Orletta, Fabio Poletto, Manuela Sanguinetti, and Maurizio Tesconi. 2018. Overview of the EVALITA 2018 Hate Speech Detection Task. In Tommaso Caselli, Nicole Novielli, Viviana Patti, and Paolo Rosso, editors, *Proceedings of the 6th Evaluation Campaign of Natural Language Processing and Speech Tools for Italian (EVALITA 2018)*, Turin, Italy. CEUR.org.

Jacopo Castellini, Valentina Poggioni, and Giulia Sorbi. 2017. Fake Twitter Followers Detection by Denoising Autoencoder. In *Proceedings of the International Conference on Web Intelligence*, WI '17, pages 195–202.

François Chollet et al. 2015. Keras. `https://keras.io`.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. *arXiv preprint arXiv:1703.04009*.

Fabio Del Vigna, Andrea Cimino, Felice Dell'Orletta, Marinella Petrocchi, and Maurizio Tesconi. 2017. Hate Me, Hate Me Not: Hate Speech Detection on Facebook. In *ITASEC*, volume 1816 of *CEUR Workshop Proceedings*, pages 86–95. CEUR-WS.org.

Njagi Dennis Gitari, Zhang Zuping, Hanyurwimfura Damien, and Jun Long. 2015. A lexicon-based approach for hate speech detection. *International Journal of Multimedia and Ubiquitous Engineering*, 10(4):215–230.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *arXiv preprint arXiv:1301.3781*.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Stuart J. Russell and Peter Norvig. 2016. *Artificial Intelligence: A Modern Approach*. Malaysia; Pearson Education Limited.

Zeerak Waseem and Dirk Hovy. 2016. Hateful symbols or hateful people? predictive features for hate speech detection on twitter. In *Proceedings of the NAACL student research workshop*, pages 88–93.

Ziqi Zhang and Lei Luo. 2018. Hate speech detection: A solved problem? the challenging case of long tail on twitter. *arXiv preprint arXiv:1803.03662*.

Exploiting Multiword Expressions to solve "La Ghigliottina"

Federico Sangati
University L'Orientale
Naples, Italy
fsangati@unior.it

Antonio Pascucci
University L'Orientale
Naples, Italy
apascucci@unior.it

Johanna Monti
University L'Orientale
Naples, Italy
jmonti@unior.it

Abstract

English. The paper describes UNIOR4NLP a system developed to solve "La Ghigliottina" game which took part in the NLP4FUN task of the Evalita 2018 evaluation campaign. The system is the best performing one in the competition and achieves better results than human players.

Italiano. *Il contributo descrive il sistema* UNIOR4NLP*, sviluppato per risolvere il gioco "La Ghigliottina", che ha partecipato alla sfida NLP4FUN della campagna di valutazione Evalita 2018. Il sistema risulta il migliore della competizione e ha prestazioni più elevate rispetto agli umani.*

1 Introduction

In this paper we describe UNIOR4NLP, a system which took part in the NLP4FUN task of the Evalita 2018 evaluation campaign (Basile et al., 2018). The goal of this task is to design a solver for "La Ghigliottina", the final game of the popular Italian TV quiz show "L'Eredità". The game involves a single player, who is given a set of five words (clues), each one linked with an unknown sixth word that represents the solution to the game. For example, given the set of clues [*fighting*, *gun*, *roof*, *eater*, *set*] the solution is *fire*, because: *the roof is on fire* is a title of a famous song, while *fire fighting*, *fire a gun*, *fire-eater*, and *set something on fire* are fixed word constructions.

UNIOR4NLP relies on the assumption that Multiword Expressions (MWEs) play an important role in solving the game: given a set of clues, the system outputs the solution word which forms the strongest connections with all of the clues.

The paper is organized as follows: in Section 2 we present related work. In Section 3 we describe the different steps we took in order to prepare and tune the UNIOR4NLP system. In Section 4 we describe our system and its functioning, while results are presented in Section 5, where we also focus on error analysis concerning both the data-set of the NLP4FUN task and our system. Finally, conclusions and future work are presented in Section 6.

2 Related work

From the very beginning of Artificial Intelligence (AI) games represented an interesting playground to test the results of research in this field (Yannakakis and Togelius, 2018). NLP plays an essential role in solving language related games and recent examples, such as the IBM Watson system in Jeopardy!^{TM} (Ferrucci et al., 2013), have proven that its use can result in groundbreaking technology. An interesting test-bed for this type of approach is represented by language games, such as the *Wheel of Fortune*, *Who Wants to be a Millionaire?* and "*La Ghigliottina*".

The game "La Ghigliottina" is particularly challenging because its solution is based on modelling how words are connected to each other. A first artificial player of the game, OTTHO (Semeraro et al., 2009; Basile et al., 2016) exploits i) resources from the web such as Wikipedia to build a lexicon and a knowledge repository and ii) a knowledge base modeling represented by an association matrix which stores the degree of correlation between any two terms in the lexicon. Word correlations are detected by connecting i) lemmas to the terms in its dictionary definition, pair of words occurring in a proverb, movie or song title, and iii) pair of similar words by exploiting Vector Space Models (Salton et al., 1975).

In our approach, we make use of similar resources but we only rely on a very limited set of syntactic constructions (patterns) to correlate words and build our association matrix.

3 Solving the Ghigliottina game

Building an automatic solver for the Ghigliottina game requires a number of preliminary steps: i) the analysis of real game instances, ii) the analysis of patterns that could help the system in solving the game, iii) the collection of the linguistic resources necessary to tune the system for the task.

3.1 Analysis of real game instances

We have analyzed a sample of 100 game instances that we personally collected from the last five editions of the TV show. We found out that in most cases each clue word is connected to the solution because they form a Multiword Expression (MWE). We have used this key observation in designing our system. We started working on our system before the announcement of the NLP4FUN task. Since our system is not supervised, the extra data-set is not adding any advantage to our system. After the official data-set was released, we found out that a good number of game instances was confirming our initial finding. However we also observed a number of unusual cases which will discuss in more depth in Section 5.2.

A MWE can be defined as a sequence of words that presents some characteristic behaviour (at the lexical, syntactic, semantic, pragmatic or statistical level) and whose interpretation crosses the boundaries between words (Sag et al., 2002). MWEs have to be considered as lexical items which convey a single meaning different from the meanings of the constituents of the MWE, such as in the idiomatic expression *kick the bucket* where the simple addition of the meanings of *kick* and *bucket* does not convey the meaning of *to die*.

We have different classes of MWEs, such as idioms (*break a leg*), verb particle constructions (*to call off*), light verbs constructions (*to provoke a reaction*). For a detailed overview of MWEs in NLP applications we refer the reader to Constant et al. (2017). For the purpose of the current task we considered only those MWEs characterized by fixed syntactic patterns described in the following section.

3.2 Pattern Analysis

A first analysis of the tuples from the sample mentioned above revealed that words in the clues are typically nouns, verbs, or adjectives, while the ones in the solutions are typically nouns or adjectives (never verbs). A more detailed investigation resulted in the definition of six patterns that identify valid MWEs connecting clue and solution pairs. We list them below with some examples from our data-set (solution words are underlined):

A B: *diario segreto* (‘diary secret’ → secret diary), *brutta caduta* (‘ugly fall’ → bad fall), *permesso premio* (‘permit price’ → good behaviour license), *dare gas* (‘give gas’ → accelerate).

A det B: *dare il permesso* (‘give the permit’ → authorize).

A prep B: *colpo di coda* (‘flick of tail’ → last ditch effort).

A conj B: *stima e affetto* (esteem and affection).

A prepart B or **A prep det B**: *virtù dei forti*, part of the famous Italian proverb *La calma è la virtù dei forti* (patience is the virtue of the strong).

A+B: compounds such as *radio* + *attività* = *radioattività* (radio + activity = radioactivity).

3.3 Linguistic Resources

On the basis of the linguistic analysis described above, we collected the linguistic resources which we deemed necessary for the task. To this end we used the following freely available corpora:

Paisà: 225 M words corpus automatically annotated (Lyding et al., 2014).

itWaC: 1.5 B words corpus automatically annotated (Baroni et al., 2009)

Wiki-IT-Titles: Wikipedia-IT titles downloaded via WikiExtractor (Attardi, 2016).

Proverbs: 1955 proverbs from Wikiquote (2016) and 371 from an online collection (Dige, 2016).

In addition, we have constructed the following lexical resources:

DeMauro-Ext: words extracted from “Il Nuovo vocabolario di base della lingua italiana”(De Mauro, 2016b), extended with morphological variations obtained by changing last vowel of the word and checking if the resulting word has frequency ≥ 1000 in Paisà.

DeMauro-MWEs: MWEs extracted from the “De Mauro online dictionary” (De Mauro, 2016a) composed of 30,633 entries.

4 System description

In order to build our system, we started processing the selected corpora via standard tokenization (only single word tokens) and removal of punctuation marks and non-word patterns. We next constructed two lexical sets: C_{LEX} to cover the *clue* words, and S_{LEX} to cover the *solution* words. S_{LEX} (composed of 7,942 nouns and adjectives in `DeMauro-Ext`) is smaller than C_{LEX} (composed of 19,414 words from the full `DeMauro-Ext` and `DeMauro-MWEs`) because solution words are almost always nouns or adjectives as described in Section 3.2.

Secondly, we built a co-occurrence matrix M_c which stores the counts c_i for every pair of words $w_i \in S_{LEX}$ and $w_j \in C_{LEX}$ such that w_i co-occurs with w_j in the resources according to patterns described in Section 3.2. Co-occurrence patterns were extracted from `Paisà` and `itWaC` with weight $w = 1$, from `DeMauro-MWE` with $w = 200$, from `Proverbs` with $w = 100$, and from `Wiki-IT-Titles` with $w = 50$. The weight were chosen manually taking into account the likelihood that a pattern in a given corpus represented a valid MWE. Compound patterns (A+B) were extracted from C_{LEX}: for every word w in C_{LEX} if $w = ab$, a and b are both in C_{LEX}, and a and b have at least 4 characters, the count for the pair (a, b) is incremented by 1 in the co-occurrence matrix.

Thirdly, for every pair of words w_i and w_j in M_c, we populate the association-score matrix M_{pmi} via the Pointwise Mutual Information measure:

$$M_{pmi}(w_i, w_j) = \log \frac{p(w_i, w_j)}{p(w_i) \cdot p(w_j)} \quad (1)$$

where

$$p(w_i) = \sum_{w_j \in C_{LEX}} M_c(w_i, w_j) \quad (2)$$

$$p(w_j) = \sum_{w_i \in S_{LEX}} M_c(w_i, w_j) \quad (3)$$

$$p(w_i, w_j) = \frac{M_c(w_i, w_j)}{\sum_{\substack{x \in S_{LEX} \\ y \in C_{LEX}}} M_c(x, y)} \quad (4)$$

Finally, for a given game instance with the 5 clue words $G = (w_{c1}, w_{c2}, w_{c3}, w_{c4}, w_{c5})$, we choose the solution word $\widehat{w_s} \in S_{LEX}$ such that:

$$\widehat{w_s} = \max_{w_s \in S_{LEX}} \sum_{w_c \in G} M_{pmi}(w_s, w_c) \quad (5)$$

that is, we choose the word in S_{LEX} which maximizes the score obtained by summing the pmi between each clue word and the candidate word. If two words are never seen co-occurring together in a pattern in the training corpora, we assign to them the lowest pmi value in M_{pmi}.

The system has been implemented in Python and the code is open source.[1] After the matrix has been loaded into memory the response time on an average laptop is around 1-2 seconds.

5 Results

According to Basile et al. (2018), UNIOR4NLP is the best performing system in the Evalita NLP4FUN task. Table 1 provides the detailed results, including split results on TV and Board Game (BG) subsets. The system achieved a very high performance: in more than half of the games (64/105) it is able to guess the correct word.

In the attempt to compare the performance of our AI system with that of a top player we analyzed the games played by Andrea Saccone, who has been the biggest champion of the Ghigliottina game so far: he was champion for 13 days (3-15 March 2018), and he managed to find the correct solution three times.[2] In comparison, UNIOR4NLP was able to win the same game instances 9 times.

SET	SIZE	MRR	R@100
TEST ALL	105	**0,64**	**0.82**
TEST TV	66	0.67	0.88
TEST BG	39	0.60	0.72
DEV ALL	315	0.56	0.80
DEV TV	204	0.61	0.85
DEV BG	111	0.48	0.71

Table 1: Results on the TEST and DEV set. Evaluations are the MRR (Mean Reciprocal Rank) and R@100 (recall at 100).

[1] https://gitlab.com/kercos/ghigliottina

[2] The players who reach the "Ghigliottina" game (the champion) continue to participate in the subsequent episodes even if they do not guess the solution word.

The plot in Figure 1 shows the distributions of the scores for the correct and missed solutions of our system on the full set of games in the development and test set (420 in total). This allows us to set a number of confidence values for our system: if the system returns a solution of a game with a score $S \geq 10$ we can be reasonably certain ($68/69 =$) that the system has guessed the correct solution, if $5 \leq S < 10$ we are above chance level ($50/70 =$), if $0 \leq S < 5$ we are at chance level ($56/112 =$) and if S is below 0, we are below chance level ($48/169 =$).

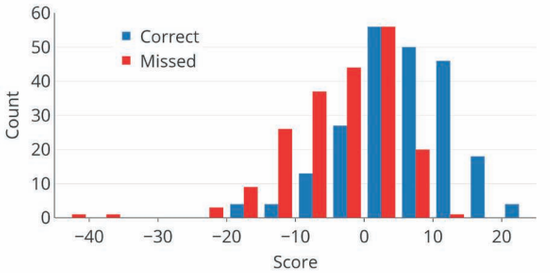


Figure 1: Distribution of the correct and missed solutions with respect to their score.

5.1 Data-set analysis

In the development data-set we found several cases which fall outside the patterns we observed in out data-set. For instance, we noticed the presence of digits in some clues or solution words (1973, 33), game instances with a clue being also the solution ('sostanza', 'fuori'), and words being spelled in different ways ('tenère', 'tenere').

Moreover, we also observed a number of 'clue - solution' pairs which are very difficult to relate. We list below some examples with some possible explanation:

- g - orecchio: (g - ear) the letter 'g' has the shape of a ear.
- classe 1973 - 33: (class 1973 - 33) this game instance was from 2006, and that year people born in 1973 were 33 years old.
- ...—... - titanic: the clue being the S.O.S. beacon in morse code.

One possible reason for these inconsistent cases is that Board Game edition use slightly different criteria to correlate words,[3] and that those from the TV set date back to the very first editions of the TV game (when correlation criteria where probably not yet well defined).

[3]This is supported by results in Table 1, where Board Game results are lower than those from the TV set.

5.2 System error analysis

In this section we analyze some types of errors that our system makes, and we provide some suggestion for possible improvement.

Word similarity Although quite rare, few of the clue-solution links can be explained by the similarity relation. An example is the clue-solution pair *sincero-franco* (sincere-frank). Those are not easily captured by patterns of the types described in Section 3.2, but could be included by means of automatic detection of word similarity via Vector Space Models (Salton et al., 1975) as done in Basile et al. (2016).

Missing words As explained in Section 3.2, we restricted the set of words in the solution set. This choice, while helping the system to restrict the search space, leads to some coverage issues. For instance, *pennello* (brush) is one of the solutions of the games in the test data not present in our solution set. In the future we would like to experiment increasing the size of the solution set while avoiding performance and memory problems.

Wrong PoS Our system analyzes words in their surface form, so it cannot distinguish cases where the same word-form can have multiple Part of Speech (PoS) (with different meaning). To avoid this problem we could envision a system which takes PoS and word-sense disambiguation into consideration.

Multiword clues Although the great majority of the clues are constituted by a single word, there are a few exceptions (typically names of saints). The current system considers only single-word tokens, so if a game has a 2-word clue, it is regarded as two separate clues (their contribution is then average to obtain the final score). The system could be optimized by using a tokenizer which keeps specific types of bigrams connected.

Association metrics As described in Section 4, we compute the association score between any pair of words in the matrix via the Pointwise Mutual Information measure (pmi). There is still a big number of alternative measures (Pecina, 2010) that might lead to higher performance.

6 Conclusions and future work

In this paper we described UNIOR4NLP, an artificial player of "La Ghigliottina", a challenging game which requires linguistic knowledge to be solved. We described the preliminary steps that we made before developing our system (identifying linguistic patterns that are relevant in the game) as well as the algorithms and the methodology we adopted. The system achieved a high performance but we believe that with further tuning it can still be improved.

Future work will focus on adopting the same methodology to automatically create novel game instances: using the same association-matrix we can choose a random word (the solution) and present the list of 5 clues with high score.

In order to make our system easily testable by the scientific community and general public, we have built an interactive version which can be accessed via a Telegram bot[4] and on Twitter[5] (see Figure 2).

Figure 2: Screenshot of @UNIOR4NLP twitter response.

[4]https://t.me/Unior4NLPbot
[5]https://twitter.com/UNIOR4NLP

Acknowledgments

This research has been partly supported by the PON Ricerca e Innovazione 2014/20 fund. Authorship contribution is as follows: Johanna Monti is author of Sections 1, 2, 3.3 and 6; Federico Sangati of Section 4 and 5, and Antonio Pascucci of Sections 3.1. and 3.2.

References

Giuseppe Attardi. 2016. Wikiextractor. http://attardi.github.io/wikiextractor. Last accessed on the 1st October 2018.

Marco Baroni, Silvia Bernardini, Adriano Ferraresi, and Eros Zanchetta. 2009. The wacky wide web: a collection of very large linguistically processed web-crawled corpora. *Language Resources and Evaluation*, 43(3):209–226.

Pierpaolo Basile, Marco de Gemmis, Pasquale Lops, and Giovanni Semeraro. 2016. Solving a complex language game by using knowledge-based word associations discovery. *IEEE Transactions on Computational Intelligence and AI in Games*, 8(1):13–26.

Pierpaolo Basile, Marco de Gemmis, Lucia Siciliani, and Giovanni Semeraro. 2018. Overview of the evalita 2018 solving language games (nlp4fun) task. In Tommaso Caselli, Nicole Novielli, Viviana Patti, and Paolo Rosso, editors, *Proceedings of the 6th evaluation campaign of Natural Language Processing and Speech tools for Italian (EVALITA'18)*. CEUR.org, Turin, Italy.

Mathieu Constant, Gülşen Eryiğit, Johanna Monti, Lonneke Van Der Plas, Carlos Ramisch, Michael Rosner, and Amalia Todirascu. 2017. Multiword expression processing: A survey. *Computational Linguistics*, 43(4):837–892.

Tullio De Mauro. 2016a. Il Nuovo De Mauro (Online). https://dizionario.internazionale.it. Last accessed on the 1st October 2018.

Tullio De Mauro. 2016b. Il Nuovo vocabolario di base della lingua italiana (pdf version). https://www.internazionale.it/opinione/tullio-de-mauro/2016/12/23/il-nuovo-vocabolario-di-base-della-lingua-italiana. Last accessed on the 1st October 2018.

Antonio Dige. 2016. Raccolta di proverbi e detti

italiani. http://web.tiscali.it/proverbiitaliani. Downloaded on the 24th April 2018.

David A. Ferrucci, Anthony Levas, Sugato Bagchi, David Gondek, and Erik T. Mueller. 2013. Watson: Beyond jeopardy! *Artif. Intell.*, 199:93–105.

Verena Lyding, Egon Stemle, Claudia Borghetti, Marco Brunello, Sara Castagnoli, Felice Dell'Orletta, Henrik Dittmann, Alessandro Lenci, and Vito Pirrelli. 2014. The PAISÀ corpus of italian web texts. In *Proceedings of the 9th Web as Corpus Workshop (WaC-9)*, pages 36–43. Association for Computational Linguistics, Gothenburg, Sweden.

Pavel Pecina. 2010. Lexical association measures and collocation extraction. *Language Resources and Evaluation*, 44(1-2):137–158.

Ivan A Sag, Timothy Baldwin, Francis Bond, Ann Copestake, and Dan Flickinger. 2002. Multiword expressions: A pain in the neck for nlp. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 1–15. Springer.

G. Salton, A. Wong, and C. S. Yang. 1975. A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620.

Giovanni Semeraro, Pasquale Lops, Pierpaolo Basile, and Marco De Gemmis. 2009. On the tip of my thought: Playing the guillotine game. In *Proceedings of the 21st International Jont Conference on Artifical Intelligence*, IJCAI'09, pages 1543–1548. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Wikiquote. 2016. Proverbi italiani. https://it.wikiquote.org/wiki/Proverbi_italiani. Downloaded on the 24th April 2018.

Georgios N Yannakakis and Julian Togelius. 2018. *Artificial Intelligence and Games*. Springer.

Computer challenges guillotine: how an artificial player can solve a complex language TV game with web data analysis

Luca Squadrone
University TorVergata
Rome, Italy
luca.squadrone@yahoo.it

Abstract

English. This paper describes my attempt to build an artificial player for a very popular language game, called "The Guillotine", within the Evalita Challenge (Basile et al., 2018). I have built this artificial player to investigate how far we can go by using resources available on the web and a simple matching algorithm. The resources used are Morph-it (Zanchetta and Baroni, 2005) and other online resources. The resolution algorithm is based on two steps: in the first step, it interrogates the knowledge base Morph-it with the five data clues, download the results and perform various intersection operations between the five data sets; in the second step, it refines the results through the other sources such as the Italian proverbs database and the IMDb. My artificial player identified the solution among the first 100 solutions proposed in 25% of cases. This is still far from systems like OTTHO (Semeraro et al., 2012) that obtained the solution in 68% of the cases. However, their result was obtained larger resources and not only with a simple web analysis.

Italiano. *Il contributo descrive il tentativo di costruire un giocatore artificiale per un gioco linguistico molto popolare, chiamato "La Ghigliottina", nell'ambito dell'Evalita Challenge (Basile et al., 2018). Ho costruito questo giocatore artificiale per verificare il limite raggiungibile utilizzando unicamente le risorse disponibili sul web e un semplice algoritmo di matching. Le risorse utilizzate sono Morph-it (Zanchetta and Baroni, 2005) e altre risorse online. L'algoritmo di risoluzione si basa su due fasi: nella prima fase, interroga la base di conoscenza Morph-it con i cinque indizi, scarica i risultati ed esegue varie operazioni di intersezione tra i cinque set di dati; nella seconda fase, affina i risultati attraverso altre fonti come il database dei proverbi italiani e l'IMDb. Il mio giocatore artificiale ha identificato la soluzione tra le prime 100 soluzioni proposte nel 25% dei casi. Il risultato ottenuto è ancora lontano da sistemi come OTTHO (Semeraro et al., 2012) che ha ottenuto la soluzione nel 68% dei casi. Tuttavia, il loro risultato è stato ottenuto con risorse più ampie e non solo con una semplice analisi web.*

1 System description

I have used Morph-it (Zanchetta and Baroni, 2005) as a basis for knowledge, instead of building one, as it is free, easy to interrogate and above all suitable for our purpose. Furthermore, it should not be underestimated that building a knowledge base involves an enormous amount of work in terms of time.

After querying the knowledge base with the five data clues, the results are downloaded and various intersection operations are performed between the five data sets. This procedure allows us to find all possible solutions and is the basis for choosing the solution.

Then to find the final solution is verified the existence of proverbs, Aphorisms, movies or books (etc.) that contain both the clue and the possible solution.

2 The memory of the system: Morph-it!

Morph-it! is a free morphological resource for the Italian language, a lexicon of inflected forms with their lemma and morphological features. It was

designed by Marco Baroni and Eros Zanchetta. The lexicon currently contains 505,074 entries and 35,056 lemmas. Morph-it! can be used as a data source for a lemmatizer/morphological analyzer/morphological generator.

The main source of linguistic data was the "Repubblica" corpus (approximately 380 million tokens), from which was extracted lemmas and inferred morphological information not present in the original corpus (i.e. gender) using distributional as well as morphological cues. With that information then was generated inflected forms for all extracted lemmas.

Morph-it includes several corpora. A corpus (plural corpora) or text corpus is a large and structured set of texts. In order to make the corpora more useful for doing linguistic research, they are often subjected to a process known as annotation. An example of annotating a corpus is part-of-speech tagging, or POS-tagging, in which information about each word's part of speech (verb, noun, adjective, etc.)

Since only some of these corpora are publicly available, I have chosen:

- La Repubblica, a corpus of Italian newspaper texts published between 1985 and 2000 (approximately 380M tokens);
- ItWac Complete, a corpus of web pages in Italian crawled from Italian university websites

These two corpora form our knowledge base and are united to provide the widest and most comprehensive knowledge base possible.

Then, these resources will be used to match the results found.

- Proverbs and Aphorisms on Italian version of wikiquotes and on the database of Italian proverbs by "Accademia della Crusca"
- Books database crawled from ibs web site
- Film titles, crawled from the Internet Movie Database
- Word definitions, from Dictionary of the Italian language HOEPLI

3 The algorithm

The basic idea of the algorithm[1] is to derive a set of words from MORPH-IT!, using only the 5 clues given in input. This will be the set of possible solutions.

Then the probability of each of these words being the actual solution will be evaluated. This is done through a second phase consisting of a verification of the existence of proverbs, Aphorisms, movies or books (etc.) that contain both the clue and the possible solution. The words with the most associations found are the solutions.

With the term "possible solutions" I will indicate a selection of words where the solution is contained, while with the term "final solutions" I will indicate the 100 words chosen among all the possible solutions.

I would also like to point out that the speed of execution of the algorithm depends on the speed of the internet connection, since I use an online knowledge base and different data must be downloaded each time.

Here is the pseudo code of the first part of the algorithm that finds the possible solutions. It is illustrated in Figure 1:

1. The algorithm takes the 5 clues as input.
2. For each clue it executes two queries, one respectively in each corpus, Repubblica and itWac Complete.
3. After downloading, the results from queries are concatenated as text.

   At the end of this procedure, for each clue, the algorithm will generate a single text. In this text there will be all the concepts with a relation to the clue.
4. Each of the five texts is transformed into a set of single words. So I get 5 sets of words, one for each clue.
5. The intersection between these sets (which I will call "Final Set") is made. The solution, semantically linked to all five clues, in most cases will be contained in the final set. This hypothesis will be verified later, in the next section.

[1] Source code https://gitlab.com/osiast/computer-challenges-guillotine

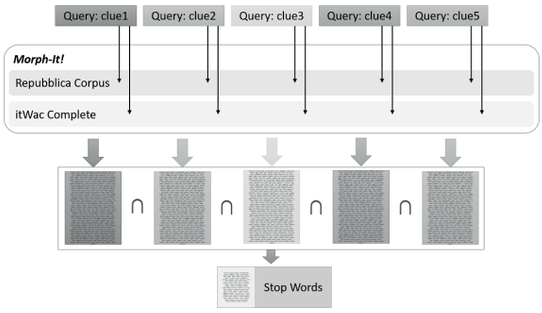


Figure 1: Process for finding possible solutions for a run of the game

6. Among the possible solutions there are many insignificant words such as articles, conjunctions and prepositions. To delete them, I subtracted the set of Stop Words from the final set.

   Stop words are words which are filtered out before or after processing of natural language data. Though ”stop words” usually refers to the most common words in a language, there is no single universal list of stop words used by all natural language processing tools, and indeed not all tools even use such a list. So, I built a special list of Stop Words specifically for this purpose, based on two online resources:

   - A collection of stopwords on github [2]
   - A collection of stopwords from the Snowball site [3]

   After the stopwords have been removed from the final set, I finally have the set of possible solutions.

The next step is the verification of the existence of proverbs, Aphorisms, movies or books (etc.) that contain both one clue and the possible solution. As already mentioned, the words with the most associations found are the final solution.

So, along with each possible solution, I search for clues within the following repositories:

- Proverbs and Aphorisms from two different resources: Italian version of wikiquotes and on the database of Italian proverbs by ”Accademia della Crusca”
- Books database crawled from ibs web site
- Film titles, crawled from the Internet Movie Database
- Word definitions, from Dictionary of the Italian language HOEPLI

Whenever the clue and the solution are found together, for example in the same proverb or title of a film, an additional weight of 0.2 is assigned to that solution. The weight can vary from 0 to 1. It indicates the probability that this is the solution of the game.

Here is the pseudo code of this second part of the algorithm.

1. For each of the 5 clues download proverbs, film and book titles, vocabulary definitions and aphorisms containing that clue and put them together in one text.

   At this point we have 5 texts.

2. To all the possible solutions I assign the value 0 as weight.

3. Whenever one of the possible solutions is found in one of these texts, its weight increases by 0.2.

4. Finally the first 100 are taken which have the largest weight in descending order.

4 Test and Results

Testing the algorithm is a key step in the validation process of the proposed solution.

In the first test below, I will run the algorithm on 315 instances of the game in order to evaluate its efficiency and study the results obtained. As already mentioned, each game is composed of five clues and a solution.

The second test will be on the ”knowledge base”. I’m going to measure how many clues contains on average. This will be useful to indicate an upper-bound of efficiency that the algorithm can not overstep.

4.1 Test the algorithm

To evaluate the efficiency of the algorithm, I tried it for 315 different games.[4]

[2] github.com/stopwords-iso/stopwords-it/blob/master/stopwords-it.txt

[3] snowball.tartarus.org/algorithms/italian/stop.txt

[4] The games used can be downloaded from this link https://goo.gl/6FpK3p

Results	Number	Percentage
Successful	242	76,83%
Fail	73	23,17%
Total games	315	

Table 1: Results on 315 games

	MRR	Returned game	Test game	Solved
This	0,0134	400	105	27
Other	0,6428	405	105	86

Table 2: comparison systems

The query is limited to 8,000 lines per request, as statistically I have noticed that it is a good compromise between the resolution speed and the efficiency of the algorithm.

I downloaded the set of games and coded them into a list. Then I ran the algorithm for each game in the list and I memorized the results.

The data has been processed to create table 1.

In over 76% of cases the exact solution is found in the "Possible Solution". This shows that "The solution, semantically linked to all five clues, in most cases will be contained in the final set". [5]

Regarding the final solutions, in 24,76% of cases the solution is among the first 100. I got a similar result with the test set, where I reached 25,7%.

Table 2 shows the distributions of the scores for the correct and missed solutions of our system on the full set of games in the test set in comparision with other system.[6]

4.2 Test the knowledge base: does it always contain the solution?

To verify that the knowledge base is suitable for our purpose and that it always contains the solution, I have performed tests on 1575 clues, that is, all the ones I had.

In particular I wanted to know if the knowledge base contained a relationship between the solution and each of them. The basic idea was to look for the clue inside the corpora and then filter the results. For this task I used the NoSketch Engine, an open-source tool to perform corpus searches.

As already mentioned, I did tests with 1575 clues of the game looking for them (one at a time) inside the corpora "Repubblica" and "ItWac Complete". The results were very positive. In fact, out of 1575 clues, 1519 of them were always connected with one or more correspondences to the solution.

We can see that out of 315 games:

- 18% (56 of them) can not be resolved due to the absence of 1 or more clues in the chosen knowledge base;
- only 5% (17 of them) can not be resolved due to the limit set on the algorithm query;

This means that without limiting the queries, I will find the solution at most 82% of the time.

So this result shows that the limit of 8000 lines per query penalized the efficiency of the algorithm by only 5% percent.

This confirms that limiting the query to 8000 rows is a good compromise.

5 Analysis of the results and future work

The algorithm's execution, both with the training set and with the test set, produced similar results. From the data obtained we can notice that the percentage of the solutions found in the first phase of the algorithm decreases in the second phase.

Furthermore the value of the MRR is very low despite the solution being found 27 times out of 105.

The reason for these results is that the number of resources in the matching phase are limited. So the solution, despite being found, is often not among first in the output of the 100 proposed solutions.

As future developments, we could improve the algorithm to find the solution from the possible solutions by increasing resources to provide more accurate results.

References

Pierpaolo Basile, Marco de Gemmis, Lucia Siciliani, and Giovanni Semeraro. 2018. Overview of the evalita 2018 solving language games (nlp4fun) task. In Tommaso Caselli, Nicole Novielli, Viviana Patti, and Paolo Rosso, editors, *Proceedings of the 6th evaluation campaign of Natural Language Processing and Speech tools for Italian (EVALITA'18)*, Turin, Italy. CEUR.org.

Giovanni Semeraro, Pasquale Lops, Marco de Gemmis, and Pierpaolo Basile. 2012. OTTHO: an artificial player for a complex language game. In *Popularize Artificial Intelligence, Proceedings of the*

[5] The full results can be viewed at this link https://goo.gl/BdCee9.

[6] MRR (Mean Reciprocal Rank)

*AI*IA Workshop and Prize for Celebrating 100th Anniversary of Alan Turing's Birth, Rome, Italy, June 15, 2012*, pages 47–53.

Eros Zanchetta and Marco Baroni. 2005. Morph-it! a free corpus-based morphological resource for the italian language. *Corpus Linguistics 2005*, 1(1).

The Perfect Recipe: Add SUGAR, Add Data

Simone Magnolini[1,2], **Vevake Balaraman**[1,3], **Marco Guerini**[1], **Bernardo Magnini**[1]
[1] Fondazione Bruno Kessler, Via Sommarive 18, Povo, Trento — Italy
[2] AdeptMind Scholar [3] University of Trento, Italy.
{magnolini, balaraman, guerini, magnini}@fbk.eu

Abstract

English. We present the FBK participation at the EVALITA 2018 Shared Task "SUGAR – Spoken Utterances Guiding Chef's Assistant Robots". There are two peculiar, and challenging, characteristics of the task: first, the amount of available training data is very limited; second, training consists of pairs `[audio-utterance, system-action]`, without any intermediate representation. Given the characteristics of the task, we experimented two different approaches: (i) design and implement a neural architecture that can use as less training data as possible, and (ii) use a state of art tagging system, and then augment the initial training set with synthetically generated data. In the paper we present the two approaches, and show the results obtained by their respective runs.

Italiano. *Presentiamo la partecipazione di FBK allo shared task "SUGAR – Spoken Utterances Guiding Chef's Assistant Robots" a EVALITA 2018. Ci sono due caratteristiche peculiari del task: primo, la quantitá di dati di training é molto limitata; secondo, il training consiste di coppie* `[enunciato-audio, azione-sistema]`*, senza alcuna rappresentazione intermedia. Date le caratteristiche del task, abbiamo sperimentato due approcci diversi: (i) la progettazione e implementazione di una architettura neurale che riesca ad usare la minor quantitá di traning possibile; (ii) l'uso di un sistama di tagging allo stato dell'arte, aumentato con dati generati in modo sintetico. Nel contributo presentiamo i due approcci, e mostriamo i risultati ottenuti nei loro rispettivi run.*

1 Introduction

In the last few years, voice controlled systems have been arising a great interest, both in research and industrial projects, resulting in many applications such as Virtual Assistants and Conversational Agents. The use of voice controlled systems allows to develop solutions for contexts where the user is busy and can not operate with traditional graphical interfaces, such as, for instance, while driving a car or while cooking, as suggested by the SUGAR task.

The traditional approach to Spoken Language Understanding (SLU) is based on a pipeline that combines several components:

- An automatic speech recognizer (ASR), which is in charge of converting the spoken user utterance into a text.
- A Natural Language Understanding (NLU) component, which takes as input the ASR output and produces a set of instructions to be used to operate on the system backend (e.g. a knowledge base).
- A Dialogue Manager (DM), which selects the appropriate state of the dialogue, based on the context of previous interactions.
- A domain Knowledge Base (KB), which is accessed in order to retrieve relevant information for the user request.
- An utterance generation component, which produces a text in natural language by taking the dialogue state and the KB response.
- Finally, a text-to-speech (TTS) component is responsible for generating a spoken response

to the user, on the base of the text produced by the utterance generation component.

While the pipeline approach has proven to be very effective in a large range of task-oriented applications, in the last years several deep learning architectures have been experimented, resulting in a strong push toward so called end-to-end approaches (Graves and Jaitly, 2014; Zeghidour et al., 2018). One of the main advantages of end-to-end approaches is avoiding the independent training of the various components of the SLU pipeline, this way reducing the need of human annotations and the risk of error propagation among components. However, despite the encouraging results of end-to-end approaches, they still need significant amount of training data, which are often not available for the task at hand. This situation is also true in the SUGAR task, where, as training data are rather limited, end-to-end approaches are not directly applicable.

Our contribution at the SUGAR task mainly focuses on the NLU component, since we make use of an 'off the shelf' ASR component. In particular, we experimented two approaches: (i) the implementation a neural NLU architecture that can use as less training data as possible (described in Section 4), and (ii) the use of a state of art neural tagging system, where the initial training data have been augmented with synthetically generated data (described in Section 5 and 6).

2 Task and Data description

In the SUGAR task (Maro et al., 2018) the system's goal is to understand a set of `command` in the context of a voice-controlled robotic agent that acts as a cooking assistant. In this scenario the user can not interact using a "classical" interface because he/she is supposed to be cooking. The training data set is a corpus of annotated utterances; spoken sentences are annotated only with the appropriate `command` for the robot. Transcription from speech to text are not available.

The corpus is collected in a 3D virtual environment, designed as a real kitchen, where users give commands to the robot assistant to accomplish some recipes. During data collection users are inspired by silent cooking videos, which should ensures a more natural spoken production. Videos are segmented into short portions (frames), that contain a single `action`, and sequentially showed to users, who have to utter a single sentence after each frame. The user's goal is to guide the robot to accomplish the same `action` seen in the frame. The resulting dataset is a list of utterances describing the actions needed to prepare three different recipes. While utterances are totally free, the `command`s are selected from a finite set of possible actions, which may refer either to to ingredients or tools. Audio files are recorded in a real acoustic environment, with a microphone posed at about 1 mt of distance from the different speakers. The final corpus contains audio files for the three recipes, grouped for each speaker, and segmented into sentences representing isolated `command`s (although few audio files may contain multiple actions (e.g. "add while mixing")).

3 Data Pre-processing

The SUGAR dataset is constituted by a collection of audio files, that needs to be pre-processed in several ways. The first step is ASR, i.e., transcription from audio to text. For this step we made use of an external ASR, selected among the ones easily available with a Python implementation. We used the Google API, based on a comparative study of the different ASR (Këpuska and Bohouta, 2017); we conducted some sample tests to be sure that the ASR ranking is reasonable also for Italian, and we confirmed our choice.

After this step, we split the dataset into training set, development set and test set; in fact the SUGAR corpus is a unique collection and there is no train-dev-test split. Although the train-dev-test split is quite standard, with two round of 80-20 split of the dataset (80% of the dataset is the training and development set, which we split 80-20 again, and 20% is the test set), in the SUGAR task we split the dataset in a more complex way. In fact, the dataset is composed by only three different recipes (i.e. a small amount of ingredients and similar sequence of operations), and with a classical 80-20 split the training, the development and the test sets would have been too different from the final set (the one used to evaluate the system). This is due to the fact this new set is composed by new recipes, with new ingredients and new a sequence of operations. To deal with this peculiar characteristic, we decided to use the first recipe as test set and the other two as train-dev sets. The final split of the data resulted in 1142 utterance and `command` pairs for training, a set of 291 pairs for

development and a set of 286 pairs for test.

Finally we substituted all the prepositions in the corpus with an apostrophe (e.g. "d'" "l'", "un'") with their corresponding form without apostrophe (e.g. "di", "lo", "una"). This substitution helps the classifiers to correctly tokenize the utterances.

In order to take advantage of the structure of the dialogue in the dataset, in every line of the corpus we added up to three previous interactions. Such previous interactions are supposed to be useful to correctly label a sample, because it is possible that either an ingredient or a verb can appear in a previous utterance, while being implied in the current utterance. The implication is formalized in the dataset, in fact the implied entity (`action` or `argument`) are surrounded by $*$. The decision of having a "conversation history" of a maximum of three utterances is due to a first formalization of the task, in which the maximum history for every utterance was set to three previous interactions. Even if this constraint has been relaxed in the final version of the task, we kept it in our system. In addition, a sample test on the data confirms the intuition that usually a history of three utterances is enough to understand a new utterance. For sake of clarity, we report below a line of the pre-processed dataset:

un filo di olio nella padella # e poi verso lo uovo nella padella # gira la frittata # togli la frittata dal fuoco

where the first three utterances are the history in reverse order, and the final is the current utterance.

4 System 1: Memory + Pointer Networks

The first system presented by FBK is based on a neural model similar to the architecture proposed by (Madotto et al., 2018), which implements a encoder-decoder approach. The encoder consists of a Gated Recurrent Unit (GRU) (Cho et al., 2014) that encodes the user sentence into a latent representation. The decoder consists of a combination of i) a MemNN that generate tokens from the output vocabulary, and ii) a Pointer network (Vinyals et al., 2015) that chooses which token from the input is to be copied to the output.

4.1 Encoder

Each word in the input sentence x from the user is represented in high-dimension by using an embedding matrix A. These representations are encoded by a Gated Recurrent Unit. The GRU takes in the current word at time t and the previous hidden state of the encoder to yield the representation at time t. Formally,

$$h_t = GRU(h_{t-1}, x_t)$$

where x_t is the current word at time t and h_{t-1} is the previous hidden state of the network. The final hidden state of the network is then passed on to the decoder.

4.2 Decoder

The input sentences, denoted by $x_1, x_2, ..x_n$, are represented as memories $r_1, r_2, ..r_n$ by using an embedding matrix R. A query h_t at time t is generated using a Gated Recurrent Unit (GRU) (Cho et al., 2014), that takes as input the previously generated output word $\hat{y}_{t-1}$ and the previous query h_{t-1}. Formally:

$$h_t = GRU(\hat{y}_{t-1}, h_{t-1})$$

The initial query h_0 is the final output vector o output by the encoder. The query h is then used as the reading head over the memories. At each time-step t, the model generates two probabilities, namely P_{vocab} and P_{ptr}. P_{vocab} denotes the probability over all the words in the vocabulary and it is defined as follows:

$$P_{vocab}(\hat{y}_t) = Softmax(Wh_t)$$

where W is the parameter learned during training. The probability over the input words is denoted by P_{ptr} and is calculated using the attention weights of the MemNN network. Formally:

$$P_{ptr}(\hat{y}_t) = a_t$$
$$a_{t,i} = Softmax(h_t^T r_i)$$

By generating two probabilities, P_{vocab} and P_{ptr}, the model learns both how to generate words from the output vocabulary and also how to copy words from the input sequence. Though it is possible to learn a gating function to combine the distributions, as used in (Merity et al., 2016), this model uses a hard gate to combine the distributions. A sentinel token \$ is added to the input sequence while training and the pointer network is trained to maximize the P_{ptr} probability for tokens that should be generated from output vocabulary. If the sentinel token is chosen by P_{ptr}, then the model

switches to P_{vocab} to generate a token, else the input token specified by P_{ptr} is chosen as output token. Though the MemNN can be modelled with $n\ hops$, the nature of the SUGAR task and several experiments that we carried on, showed that adding more hops is not useful. As a consequence the model is implemented as a single hop as explained above.

We use the pre-trained embeddings from (Bojanowski et al., 2016) to train the model.

5 System 2: Fairseq

The second system experimented by FBK is based on the work in (Gehring et al., 2017). In particular, we make use of the Python implementation of the toolkit known as Fairseq(-py)[1]. The toolkit is implemented using PyTorch, and provides reference implementations of various sequence-to-sequence models. There are configurations for several tasks, including translation, language model and stories generation. In our experiment we use the toolkit as a black-box since our goal is to obtain a dataset that could be used with this system; hence, we use the generic model (not designed for any specific task) without fine tuning. Moreover, we do not add any specific feature or tuning for the implicit arguments (the ones surrounded by $*$), but we let the system learn the rule by itself.

A common approach in sequence learning is to encode the input sequence with a series of bi-directional recurrent neural networks (RNN); this can be done with Long Short-Term Memory (LSTM) networks, Gated Recurrent Unit (GRU) networks or other types of network, and generate a variable length output with another set of decoder RNNs, not necessarily of the same type, both of which interface via an attention mechanism (Bahdanau et al., 2014; Luong et al., 2015).

On the other hand convolutional networks create representations for fixed size contexts, that can be seen as a disadvantage compared to the RNNs. However, the context size of the convolutional network can be expanded by adding new layers on top of each other. This allows to control the maximum length of dependencies to be modeled. Furthermore, convolutional networks allow parallelization over elements in the sequence, because they do not need the computations of the previous time step. This contrasts with RNNs, which maintain a hidden state of the entire past that prevents parallel computation within a sequence. This can increase dramatically the training time of the system without reducing the performance, as shown in (Gehring et al., 2017).

The weak point of the system is that it needs a consistent amount of training data to create reasonable models. In fact, Fairseq(-py) trained with only the SUGAR dataset can not converge and gets stuck after some epochs, producing pseudo-random sequences. Due to the small size of the SUGAR training set, combined with its low variability (training data are composed by possible variations of only two recipes), for the system is impossible to learn the correct structure of the commands (e.g. balancing the parenthesis) or to learn how to generalize `arguments`. In order to use effectively this system we have expanded the SUGAR dataset with data augmentation techniques, presented in Section 6.

6 Data augmentation

Overfitting is still an open issue in neural models, especially in situations of data sparsity. In the realm of NLP, regularization methods are typically applied to the network (Srivastava et al., 2014; Le et al., 2015), rather than to the training data.

However, in some application fields, data augmentation has proven to be fundamental in improving the performance of neural models when facing insufficient data. The first fields exploring data augmentation techniques were computer vision and speech recognition. In these fields there now exist well-established techniques for synthesizing data. In the former we can cite techniques such as rescaling or affine distortions (LeCun et al., 1998; Krizhevsky et al., 2012). In the latter, adding background noise or applying small time shifts (Deng et al., 2000; Hannun et al., 2014).

In the realm of NLP tasks, data augmentation has received little attention so far, some notable exceptions being feature noising (Wang et al., 2013) or Kneser-Ney smoothing (Xie et al., 2017). Additionally, negative examples generation has been used in (Guerini et al., 2018).

In this paper we build upon the idea of the aforementioned papers by moving a step forward and taking advantage of the structured nature of the SUGAR task and of some domain/linguistic knowledge. In particular, we used the following methods to expand the vocabulary and the size of the training data, but applying some substitution

[1] `https://github.com/pytorch/fairseq`.

strategies to the original data:

- **most-similar token substitution**: based on a similarity mechanisms (i.e. embeddings).
- **synonym token substitution**: synonymy relations taken from an online dictionary and applied to specific tokens.
- **entity substitution**: replace entities in the examples with random entities of the same type taken from available gazetteers.

The first approach implies substituting a token from a training example with one of the five most similar tokens (chosen at random) found through cosine similarity in the embedding space described in (Pennington et al., 2014). We use the top five candidates in order to add variability, since many tokens appeared multiple times in the training data. If the token appeared also as an `argument` in the `command`, it was substituted as well, while if it appeared as `action` it was left unchanged. This approach was applied with a probability of 30% on each token of the utterances in the training data.

The second approach has been used over verbs recognized in training utterances using the TextPro PoS tagger (Pianta et al., 2008). Such verbs have been substituted with one possible synonym taken from an electronic dictionary[2]. Also in this case, the `action` in the `command` was kept the same (in fact the verbs present in the utterance are usually paired with the `action` in the `command`). The third approach has been used to substitute ingredients in the text with other random ingredients from a list of foods (Magnini et al., 2018). In this case the ingredient has been modified accordingly also in the annotation of the sentence.

These methodologies allow to generate several variants starting from a single sentence. While the first approach has been used in isolation, the second and the third one have been used together to generate additional artificial training data. Doing so, we obtained two different data sets: the first is composed by 45680 pairs of utterances and `commands` (most-similar token applied forty times per example, $1142 * 40$); the second dataset contains 500916 pairs (each original sentence got at least each verb replaced 3 times, and for each of these variants, ingredients were randomly substituted twice), the high number of variants is due to the inclusion of the history of three previous utterances in the process.

[2] `http://www.sinonimi-contrari.it/`.

7 Results

	Actions	Arguments
Memory + Pointer Networks		
- Data Augmentation	65.091	30.856
+ Data Augmentation	65.396	35.786
Fine Tuning	66.158	36.102
Fairseq		
+ Data Augmentation	66,361	46,221

Table 1: Accuracy of the two experimented approaches in recognizing actions and their arguments.

Results of the two approaches are reported in Table 1. Both approaches obtain a higher accuracy in recognizing `actions`, than in recognizing `arguments`. Fairseq trained with augmented data is the top performer of the task, outperforming more than 10% of accuracy on `arguments` compared to the others approach. The ablation test on Memory + Pointer Networks also show the importance of data augmentation for tasks with low resources, in particular fine tuning the classifier with the new data.

8 Conclusion and Future Work

We presented the FBK participation at the EVALITA 2018 Shared Task "SUGAR – Spoken Utterances Guiding Chef's Assistant Robots". Given the characteristics of the task, we experimented two different approaches: (i) a neural architecture based on memory and pointer network, that can use as less training data as possible, and (ii) a state of the art tagging system, Fairseq, trained with several augmentation techniques to expand the initial training set with synthetically generated data. This second approach seems promising and in the future we want to deeper investigate the effect of the different techniques of data augmentation on the performances.

Acknowledgments

This work has been partially supported by the AdeptMind scholarship.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly

learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734. Association for Computational Linguistics.

Li Deng, Alex Acero, Mike Plumpe, and Xuedong Huang. 2000. Large-vocabulary speech recognition under adverse acoustic environments. In *Sixth International Conference on Spoken Language Processing*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. *arXiv preprint arXiv:1705.03122*.

Alex Graves and Navdeep Jaitly. 2014. Towards end-to-end speech recognition with recurrent neural networks. In *International Conference on Machine Learning*, pages 1764–1772.

Marco Guerini, Simone Magnolini, Vevake Balaraman, and Bernardo Magnini. 2018. Toward zero-shot entity recognition in task-oriented conversational agents. In *Proceedings of the 19th Annual SIGdial Meeting on Discourse and Dialogue*, pages 317–326, Melbourne, Australia, July.

Awni Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, et al. 2014. Deep speech: Scaling up end-to-end speech recognition. *arXiv preprint arXiv:1412.5567*.

Veton Këpuska and Gamal Bohouta. 2017. Comparing speech recognition systems (microsoft api, google api and cmu sphinx). *Journal of Engineering Research and Application*, 7(3):20–24.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Quoc V Le, Navdeep Jaitly, and Geoffrey E Hinton. 2015. A simple way to initialize recurrent networks of rectified linear units. *arXiv preprint arXiv:1504.00941*.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Andrea Madotto, Chien-Sheng Wu, and Pascale Fung. 2018. Mem2seq: Effectively incorporating knowledge bases into end-to-end task-oriented dialog systems. *arXiv preprint arXiv:1804.08217*.

Bernardo Magnini, Vevake Balaraman, Mauro Dragoni, Marco Guerini, Simone Magnolini, and Valerio Piccioni. 2018. Ch1: A conversational system to calculate carbohydrates in a meal. In *Proceedings of the 17th International Conference of the Italian Association for Artificial Intelligence (AI*IA 2018)*.

Maria Di Maro, Antonio Origlia, and Francesco Cutugno. 2018. Overview of the EVALITA 2018 Spoken Utterances Guiding Chef's Assistant Robots (SUGAR) Task. In Tommaso Caselli, Nicole Novielli, Viviana Patti, and Paolo Rosso, editors, *Proceedings of the 6th evaluation campaign of Natural Language Processing and Speech tools for Italian (EVALITA'18)*, Turin, Italy. CEUR.org.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Emanuele Pianta, Christian Girardi, and Roberto Zanoli. 2008. The textpro tool suite. In Bente Maegaard Joseph Mariani Jan Odijk Stelios Piperidis Daniel Tapias Nicoletta Calzolari (Conference Chair), Khalid Choukri, editor, *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco, may. European Language Resources Association (ELRA). http://www.lrec-conf.org/proceedings/lrec2008/.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2692–2700. Curran Associates, Inc.

Sida Wang, Mengqiu Wang, Stefan Wager, Percy Liang, and Christopher D Manning. 2013. Feature noising for log-linear structured prediction. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1170–1179.

Ziang Xie, Sida I Wang, Jiwei Li, Daniel Lévy, Aiming Nie, Dan Jurafsky, and Andrew Y Ng. 2017. Data noising as smoothing in neural network language models. *arXiv preprint arXiv:1703.02573*.

Neil Zeghidour, Nicolas Usunier, Gabriel Synnaeve, Ronan Collobert, and Emmanuel Dupoux. 2018. End-to-end speech recognition from the raw waveform. *Interspeech 2018*, Sep.