INTERSPEECH 2019
September 15–19, 2019, Graz, Austria

Neural Transition Systems for Modeling Hierarchical Semantic Representations

Riyaz Bhat, John Chen, Rashmi Prasad, Srinivas Bangalore

Interactions LLC, USA

rbhat@interactions.com, jchen@interactions.com, rprasad@interactions.com, sbangalore@interactions.com

Abstract

While virtual agents are becoming ubiquitous in our daily life, their functionality is limited to simple commands which involve a single intent and an unstructured set of entities. Typically, in such systems, the natural language understanding (NLU) component uses a sequence tagging model to extract a flat meaning representation. However, in order to support complex user requests with multiple intents with their associated entities, such as those in a product ordering domain, a structured semantic representation is necessary. In this paper, we present hierarchical semantic representations for product ordering in the food services domain and two NLU models that produce such representations efficiently using deep neural networks. The models are based on transition-based algorithms which have been proven to be effective and scalable for multiple NLP tasks such as syntactic parsing and slot filling. The first model uses a multi-tasking architecture containing multiple transition systems with tree constraints to model the hierarchical annotations, while the second model treats the task as a constituency parsing problem by mapping the target domain annotations to a constituency tree. We demonstrate that both multi-task and constituency-based transition systems achieve competitive results and even show improvements over sequential models, showing their effectiveness in modeling hierarchical structure.

1. Introduction

Virtual agents attempt to carry out the user's spoken requests. Natural language understanding (NLU) is a key part of these systems. The NLU output that is used by many virtual agents consists of representing a single user utterance as a set of slot fillers and a single intent. In this work, we explore NLU models that extract a more structured representation from each utterance.

Much previous work involves detecting slot-filler type representations from user utterances, with each utterance typically consisting of a single intent. This kind of work often involves experimentation on the ATIS corpus [1]. For example, Xu and Sarikaya [2] implement a CNN-CRF that performs joint intent detection and slot filling over this corpus. Kurata et al. [3] employ an encoder-labeler LSTM to extract intents and slot-fillers over a corpus consisting of not only ATIS but also the MIT Movie corpus. Alternatively, Hakkani-Tür et al. [4] experiment with bi-directional RNNs for extraction of intent, slot-fillers, and domain given a user utterance over ATIS and a proprietary virtual assistant corpus. In contrast, Asri et al. [5] implement a DNN to perform intent and slot filler extraction over user utterances in the Maluuba corpus. In this corpus, a user utterance may not infrequently consist of more than one intent, but besides that the NLU representation consists of flat slot fillers. Their modeling is accomplished through a DNN structured as two bi-GRUs, one for intents and the other for slot fillers.

Chen et al. [6] has shown that for the product ordering task, a more complex NLU representation of the user utterance is often required in order for the system to carry out the user's wishes. Furthermore, they designed an appropriate NLU representation, and performed preliminary experiments with maximum entropy Markov models (MEMM) for this task.

State of the art NLU models involve DNNs ([7, 8]). In this paper, we experiment with various DNN models that predict the more structured NLU representation that is needed to recognize user intent in tasks such as product ordering. We show that DNN models offer improved performance over MEMM models. Furthermore, our more structured DNN models are competitive with state of the art BiLSTM-CRF models.

2. Semantic Representation

Our semantic representation is designed to support and evaluate the NLU component in our system architecture for order-taking. For each caller utterance, the representation captures (a) the caller's *intents* in the utterance, (b) *entities*, which correspond to the basic orderable elements in the catalog or menu, (c) *attributes* (or customizations) of the entities such as ingredients (in the food domain), size, quantity, etc., and (d) the phrasal grouping of entities and attributes as *items* and *attribute phrases*. We developed annotation guidelines to annotate (a)-(d) in transcribed product ordering dialogs.[1]. Figure 1 shows caller utterances from our data, annotated for intents, entities, attributes and items.

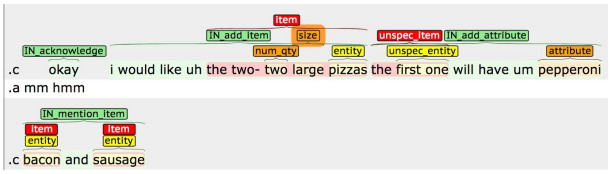

Figure 1: *Ordering dialog tagged with intents (green), entities (yellow), attributes (orange) and items (red)*

The annotation guidelines were created in light of the following objectives:

- First, because of the lack of annotated data for the product ordering domain, the semantic characterization abstracts away from specific elements of applications. In this way, while hypothesizing that the broad language used by customers across applications would be largely similar, we aim for models trained on data collected from a few applications, and generalizable to other applications where annotated data is not available. This semantic abstraction is reflected in the example in the Figure,

[1] We used the brat tool for annotation (http://brat.nlplab.org)

http://dx.doi.org/10.21437/Interspeech.2019-3075

where the application specific elements such as "pizza", "pepperoni", "bacon" and "sausage" are tagged as either *entity* or *attribute*, depending on their syntactic context.[2]

- Second, since products are often expressed as noun phrases (NP), with the orderable entity as the NP head and possible attributes as its modifiers, this entity-attribute relation is captured via a higher-order level of representation, called *item*, which extends over the NP and groups these related elements, as seen in the Figure. All entities project an item, regardless of any modifying attributes, while attributes that lie outside the scope of the NP are not similarly grouped with their head.
- Third, the grouping of entities and attributes as items reflects a hierarchical relationship of the tag types, which is utilized by one of our methods which creates separate models for each level, with the higher level model using tags assigned at the lower levels. To this extent, the scheme incorporates the notion of tag type inheritance for maximizing prediction accuracy at higher levels. For example, as mentioned above, all entities (along with their NP modifying attributes) project items. In addition, the intent tag is partly determined by the distribution of the top level tags within the intent.
- Fourth, in our system architecture, the NLU component is designed to process each caller utterance independent of the prior dialog context, extracting an under-specified semantic representation to pass on to the dialog manager, which is responsible for creating the fully specified representation for the order. Accordingly, the annotation scheme uses under-specified tags when interpretation requires knowledge of context beyond the utterance. For example, intents are assigned a *mention* type when the phrase or clause corresponding to the intent is ambiguous and requires context for disambiguation. We also do not resolve coreference relations, although under-specified coreferring expressions are annotated as *unspec_entity* (and correspondingly, *unspec_item*).[3]

We have evaluated the reliability of the annotation as accuracy in terms of a strict match of spans and labels between two annotators, where the second expert annotator reviewed and corrected annotations of the first trained annotator. For 521 customer utterances containing 1352 tags, we achieve 86% agreement between the annotators.

3. Models

3.1. Transition-Based Slot Filling

We propose two architectures that segment and label an input sequence and produce nested annotations using transition-based algorithms which are commonly used for constituency and dependency parsing. The first model uses a separate transition system for each level in the hierarchy and enforces nesting constraints to produce non-overlapping structures. The transition system used in this model allows us to construct representations of multi-word segments which have been shown to be essential for multi-word annotations such as NER and chunking ([7], [9], [10] and [11]). We will refer to this model as the Multi-task Shift-Reduce (MT-SR for short) system. The second model treats the multi-level annotations as a phrase structure tree and uses a transition-based system to predict the full structure in a single pass. We will refer to this model as the Phrase Structure Shift-Reduce (PS-SR for short) system.

3.1.1. Multi-task Shift-Reduce Model

Given a sequence of words $x_1, x_2, \ldots, x_n$, the goal of this model is to output a set of multi-level labelled slots. The model employs a transition system similar to arc-standard parser of Nivre [12]. The transition system uses a stack for building segment representations and a buffer that contains unprocessed input. The state of these data structures defines the state of the system which is changed at each step by applying an appropriate transition. For each level, we use the following transitions: The `SHIFT` transition moves a word from the buffer to the stack, the `NO-REDUCE` transition removes a word from the stack while the `REDUCE(y)` transition pops all items from the stack, creating a labeled slot. In order to make sure that the final structure is not discontinuous and does not have crossing substructures, we add some constraints on the predicted transitions. First, we allow the model to build the hierarchy bottom-up and then the higher levels are forced to contain lower level predictions as a whole, disallowing crossing structures. In case of a violation, the state of the system is updated by applying the next best transition. The algorithm takes a maximum of $2n$ actions and completes when both stack and buffer are empty. The sequence of actions taken by the system for an entity level annotation are depicted on the left in Figure 3.1.2.

3.1.2. Phrase Structure Shift-Reduce system

The MT-SR model has a major drawback. The number of output layers grows linearly with the number of levels in the hierarchy. The PS-SR model provides an efficient alternative. Given a sequence of words $x_1, x_2, \ldots, x_n$, the model predicts a nested structure in a single pass. The PS-SR model also employs a different transition system which is similar to the shift-reduce parser for constituency parsing [13, 14]. The PS-SR model also uses similar data structures to process the input. However, its stack stores (partially) processed binarized tree(s), unlike linear input in the case of MT-SR.

This system also consists of three but slightly different types of transitions: the `SHIFT` transition pushes the top word from the buffer to the stack, the `REDUCE-X` pops the top two items s_0 and s_1 from the stack and combines them as a new tree element $X \rightarrow (s_0\ s_1)$ which is then pushed back onto the stack, while the `UNARY-X` transition pops the top item s_0 from the stack and constructs a new tree element $X \rightarrow s_0$, which is also pushed back onto the stack. The system assumes a constituency-style tree structure as input, containing unary and binary branching. Wang et al. [15] used binary left-branching to binarize the trees in each forest in a nested NER. We, however, show that right-branching is more appropriate for nested structures that contain noun phrases. Since noun phrases in English are head final, right-branching would allow us to build the structure around the head noun, which may be beneficial for learning. In our case, three out of four levels of annotation revolve around noun phrases. Moreover, since PS-SR uses a bottom-up strategy to build the structure, right-branching has an additional benefit. It allows the model to sustain the benefits of bottom-up parsing even after binarization, which is not true for left-branching. The difference between the left-branching

[2] In the Figure, the syntactic modification context is the determinant of the tagging of "pepperoni" as an attribute, whereas the lack thereof is the basis for tagging "bacon" and "sausage" as entities. The latters' treatment as an attribute of the item is taken up by the dialog manager.

[3] In future work, we plan to annotate intra-utterance coreference relations and model these with NLU methods.

and right-branching is depicted in Figure 2. The sequence of actions taken by the system are shown on the right in Figure 3.1.2.

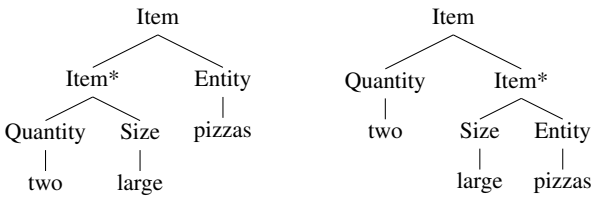

Figure 2: *Left and right branching showing the binarized tree structure for "two large pizzas". In right branching top level (Item) is build around "large pizzas", "pizzas" being the linguistic head of the Item, while in Left branching Item is build around sub phrase "two large", neither of the words project the item.*

Unlike phrase structure trees, our annotations do not necessarily form a rooted tree. There are many instances in our data sets where words are left out from the annotation. Therefore, following Wang et al. [15] we add an auxiliary symbol $ to each sentence to conveniently determine the termination of the transition process, rather than adding a dummy root which increases the time complexity of the system. Our transition system takes a maximum of kn actions[4] to build the nested structure due to the fact that we also consider unary rules of the form $X_1 \rightarrow X_2$. Wang et al. [15] on the other hand ignore such rules.

Transition	Stack	Buffer
	[]	[the two- two large pizzas]
SHIFT	[the]	[two- two large pizzas]
NO-REDUCE	[]	[two- two large pizzas]
SHIFT	[two-]	[two large pizzas]
NO-REDUCE	[]	[two large pizzas]
SHIFT	[two]	[large pizzas]
REDUCE(quantity)	[]	[large pizzas]
SHIFT	[large]	[pizzas]
REDUCE(size)	[]	[pizzas]
SHIFT	[pizzas]	[]
REDUCE(entity)	[]	[]

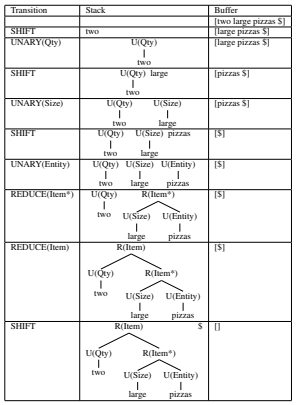

Figure 3: *Sequence of transitions taken by MT-SR and PS-SR for entity and full representation (see Figure 1 for the annotations for 'two large pizzas'). U and R in second column of the right figure stand for Unary and Reduce actions, respectively.* stands for intermediate non-terminals introduced after binarization.*

3.2. Neural Transition-based Model

We use neural networks to learn the representations underlying the transitions of our shift-reduce models.

3.2.1. Representation of Words

Given an utterance with word sequence $w_1, \ldots, w_n$ and menu tag sequence $g_1, \ldots, g_n$ (represented in the form of Begin, Inside, Outside format), we associate each word and menu tag with an embedding vector $e(w_i)$ and $e(g_i)$, and create a sequence of input vectors $x_{1:n}$ in which each x_i is obtained through concatenation of the corresponding word, menu, and character representations.

$$x_i = e(w_i) \oplus c(w_i) \oplus e(g_i) \tag{1}$$

Character representation $c(w_i)$ is obtained by learning a character-level model using bidirectional LSTM [16]. Formally, given a character sequence $c_1, c_2, \ldots, c_m$ for the i-th word, we use the last hidden states of forward and backward LSTM as the character-based representation of the word (w_i) as shown below:

$$BiLSTM_c(w_{1:m}) = \overrightarrow{LSTM}_c(w_{1:i}) \oplus \overleftarrow{LSTM}_c(w_{m:i}) \tag{2}$$

$$c(w_i) = BiLSTM_c(w_{1:m}, i) \tag{3}$$

We then use a token-level LSTM to encode the contextual information spread across the utterance. Formally, given a sequence of input vectors $x_1, x_2, \ldots, x_n$, we compute token representation t_i for the i-th input vector by concatenating the matching forward and backward hidden states of forward and backward LSTM in case of MT-SR model, while we use the hidden state of backward LSTM for PS-SR model, as shown below:

$$BiLSTM_t(x_{1:n}) = \overrightarrow{LSTM}_t(x_{1:i}) \oplus \overleftarrow{LSTM}_t(x_{n:i}) \tag{4}$$

$$t_i^{mt} = BiLSTM_t(x_{1:n}, i) \tag{5}$$

$$t_i^{ps} = \overleftarrow{LSTM}_t(x_{n:i}, i) \tag{6}$$

3.2.2. Representation of Parser States

Generally, in shift-reduce models, relevant features are extracted from a given parser state which are then used for predicting the next best transition to update the parser state. Both of our SR models use different strategies to extract features from parser configurations or states, which we will discuss below:

MT-SR Model In case of the MT-SR model, each SR system relies on features from the stack by average pooling its contents. Given the contents of the stack $s_1, s_2, \ldots, s_n$, average pooling is defined as follows:

$$s_c = ave(BiLSTM_t(s_{1:n})) \tag{7}$$

The resulting average pooled vector is then fed to a single-layer neural network for multiclass classification.

$$c = softmax(W \cdot s_c) \tag{8}$$

where cardinality of c is equal to the number of labelled transitions. Finally, the training objective in our multi-tasking architecture is to find parameters that maximize the data log-likelihood jointly for all tasks (layers in our annotation).

PS-SR Model Following Dyer et al. [17], we use Stack-LSTM to encode the contents (partial tree(s)) of the stack of our PS-SR system. For encoding tree representation in an evolving stack, Stack-LSTM provides an efficient implementation to keep track of the top element of the stack by using a stack-pointer. Given the representation of partial trees in the stack $h_{t_n}, \ldots, h_{t_1}$, the state of the stack s_k at time step k is computed as follows:

$$s_k = \text{Stack-LSTM}[h_{t_n}, ..., h_{t_1}] \tag{9}$$

We use composition functions similar to the ones used in the Recursive Neural Network Socher et al. [18] to build the representation of partial trees h_{t_i} bottom up, as:

$$h_{parent,l} = W_u \cdot h_{child} + b_u \tag{10}$$

$$h_{parent,l} = W_b \cdot [h_{lchild}, h_{rchild}] + b_b \tag{11}$$

[4] k is the number of levels in the hierarchy. In our case, the value of k is 4.

where W/b_u and W/b_b are the trainable parameters for unary(u) and binary(b) compositions for a non-terminal node(l).

We also compute the LSTM representation of action history on the fly. Formally, given an action history sequence $a_1, a_2, \ldots, a_{k-1}$, we can compute action history representation a_k at time step k as follows:

$$a_k = \overrightarrow{LSTM}_a(a_1 : a_{k-1}) \tag{12}$$

Finally, the parser configuration c_k is represented by the concatenation of the top states from input (buffer), stack and action, which is then fed to a single-layer neural network for multiclass classification.

$$c_k = [t_k^{ps}, s_k, a_k] \tag{13}$$

$$p = softmax(W \cdot c_k) \tag{14}$$

3.3. Baseline models

Our baselines are either MEMMs [19] or BiLSTM-CRFs [7]. For each type, there is one model trained for each annotation level. Hence, for each model type, each input sentence is run through four different models. For MEMMs, the output of one level is input to the next level. For BiLSTM-CRFs, the input to each level is always the input word sequence.

4. Experiments

We evaluate our models on two food services datasets which are extracted from customer call logs from restaurants, one selling pizzas and the other selling burgers. Only the utterances that are relevant for food ordering have been annotated. Each dataset was split into training, development, and test sets as follows. A fixed number of about 750 utterances was initially set aside as the test set. Subsequently, 87.5% of the remaining utterances became the training set with the rest (12.5%) being used as the development set. Table 1 shows the data statistics for each dataset.

Table 1: *Number of utterances in different datasets.*

Dataset	Train	Dev	Test	Total	Nesting
Burger	6830	975	763	8568	57%
Pizza	2727	184	782	3693	37%

4.1. Experimental Setup

The distributed representation of words from food domains are learned from two sources. We first train a generic embedding model on English Wikipedia data set. To address missing food words from generic space, another model is trained on data crawled from food blogs, restaurant web pages and description of the search results for food related queries. We use two multivariate linear regression models to create a distributed representation for words that are present in one embedding space, but missing from the other. Finally, the embeddings that we use in our models are average pooled from both spaces. The regression models are trained on the top 10K most frequently occurring words from both embedding spaces. The Wiki data used for training the embedding model contains around 300M sentences, while the food data contains around 655K sentences. The word representations are learned using Skip-gram model with negative sampling. Word embeddings have 64 dimensions. The embeddings of characters of a word, menu and boundary tags (Begin, Inside) of a text segment are initialized randomly with 32, 32 and 3 dimensions respectively. In all our models, we use 2-layered LSTM for word and/or tree representation and single-layered LSTM for character and/or action representation with 256 and 64 hidden dimensions respectively. All the models are trained using AMSGrad and the gradients of SR-PS model are clipped to a value of 5.0 to avoid exploding gradients [20]. We use dropout [21] to regularize our models. For all LSTMs, we use recurrent dropout [22] with a drop rate of 0.5 between hidden states and an output dropout with a drop rate of 0.5 for output states. We also use embedding dropout with a rate of 0.3 on word and character embeddings. We use early stopping based on the performance of development sets.

We evaluate our models using recall R (percent of brackets found by model that are correct), precision P (percent of true brackets that are found by the model), and F-measure F (harmonic mean of recall and precision), where $F = \frac{2RP}{R+P}$. A bracket is a consecutive word sequence in the input utterance. This metric is commonly used in related tasks, such as named entity recognition [23] and phrasal chunking [24].

5. Results

All the results are reported in Table 2. Among the sequence tagging models, BILSTM-CRF is more accurate and results in about a 5% increase in F score over the use of MEMM. In comparison to the tagging models, both of our transition-based models achieve competitive results on the task. Our neural MT-SR model performs slightly lower than the BiLSTM-CRF tagger. The reason for the drop in performance could be the fact that MT-SR model uses greedy decoding for inference, while the BiLSTM-CRF globally normalizes each layer in the hierarchical structure for improved performance. However, our PS-SR models perform better than BiLSTM-CRF on both datasets by considerable margin[5]. Since the PS-SR models are designed to model hierarchical structures, the performance gain is obvious. The lower performance of the CRF model can be attributed to its lack of ability to model hierarchical structure between segments directly [15]. As we speculated earlier in the models section, right-branching has provided better results on both datasets.

Table 2: *SR models achieve competitive results against BiLSTM-CRF. LB = Left Branching, RB = Right Branching.*

Model	Dataset	Precision	Recall	F
MEMM	Burger	78.91	77.72	78.31
	Pizza	69.85	60.15	64.64
BiLSTM-CRF	Burger	82.46	82.94	82.70
	Pizza	68.82	69.43	69.12
MT-SR	Burger	83.05	82.28	82.66
	Pizza	68.20	68.84	68.51
PS-SR(LB)	Burger	83.14	82.51	82.82
	Pizza	69.61	69.65	69.63
PS-SR(RB)	Burger	83.42	82.81	83.11
	Pizza	69.45	70.14	69.79

6. Conclusion

In this paper, we have presented two Shift-Reduce models for modeling hierarchical semantic representations for product ordering domain. Our evaluations on two data sets from food services show that our models achieve competitive results against the state-of-the-art models for sequence tagging.

[5]Note that we perform the exact match evaluation over the segments not the individual tokens.

7 References

[1] C. T. Hemphill, J. J. Godfrey, and G. R. Doddington, "The ATIS spoken language systems pilot corpus," in *Speech and Natural Language: Workshop Proceedings*, Hidden Valley, Pennsylvania, June 1990.

[2] P. Xu and R. Sarikaya, "Convolutional neural network based triangular crf for joint intent detection and slot filling," in *Automatic Speech Recognition and Understanding (ASRU)*, December 2013, pp. 78–83.

[3] G. Kurata, B. Xiang, B. Zhou, and M. Yu, "Leveraging sentence-level information with encoder lstm for semantic slot filling," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2016, pp. 2077–2083.

[4] D. Hakkani-Tür, G. Tür, A. Celikyilmaz, Y.-N. V. Chen, J. Gao, L. Dung, and Y. yi Wang, "Multi-domain joint semantic frame parsing using bi-directional rnn-lstm," in *Proceedings of The 17th Annual Meeting of the International Speech Communication Association (INTERSPEECH 2016)*, June 2016.

[5] L. E. Asri, H. Schulz, S. Sharma, J. Zumer, J. Harris, E. Fine, R. Mehrotra, and K. Suleman, "Frames: A corpus for adding memory to goal-oriented dialogue systems," in *Proceedings of the SIGDIAL 2017 Conference*, Saarbrücken, Germany, August 2017, pp. 207–219.

[6] J. Chen, R. Prasad, S. Stoyanchev, E. Selfridge, S. Bangalore, and M. Johnston, "Corpus and annotation towards nlu for customer ordering dialogs," in *Proceedings of the 2018 IEEE Spoken Language Technology Workshop (SLT)*, 2018, pp. 707–713.

[7] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer, "Neural architectures for named entity recognition," in *Proceedings of NAACL-HLT 2016*, 2016, pp. 260–270.

[8] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," *arXiv preprint arXiv:1802.05365*, 2018.

[9] S. Sarawagi and W. W. Cohen, "Semi-markov conditional random fields for information extraction," in *Advances in neural information processing systems*, 2005, pp. 1185–1192.

[10] Y. Lou, Y. Zhang, T. Qian, F. Li, S. Xiong, and D. Ji, "A transition-based joint model for disease named entity recognition and normalization," *Bioinformatics*, vol. 33, no. 15, pp. 2363–2371, 2017.

[11] F. Zhai, S. Potdar, B. Xiang, and B. Zhou, "Neural models for sequence chunking," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[12] J. Nivre, "Incrementality in deterministic dependency parsing," in *Proceedings of the Workshop on Incremental Parsing: Bringing Engineering and Cognition Together*. Association for Computational Linguistics, 2004, pp. 50–57.

[13] K. Sagae and A. Lavie, "A classifier-based parser with linear runtime complexity," in *Proceedings of the Ninth International Workshop on Parsing Technology*. Association for Computational Linguistics, 2005, pp. 125–132.

[14] T. Watanabe and E. Sumita, "Transition-based neural constituent parsing," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, vol. 1, 2015, pp. 1169–1179.

[15] B. Wang, W. Lu, Y. Wang, and H. Jin, "A neural transition-based model for nested mention recognition," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018, pp. 1011–1017.

[16] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[17] C. Dyer, M. Ballesteros, W. Ling, A. Matthews, and N. A. Smith, "Transition-based depe ndency parsing with stack long short-term memory," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*. Association for Computational Linguistics, 2015, pp. 334–343.

[18] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Ng, and C. Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," in *Proceedings of the 2013 conference on empirical methods in natural language processing*, 2013, pp. 1631–1642.

[19] A. McCallum, D. Freitag, and F. C. Pereira, "Maximum entropy markov models for information extraction and segmentation." in *Icml*, vol. 17, no. 2000, 2000, pp. 591–598.

[20] R. Pascanu, C. Gulcehre, K. Cho, and Y. Bengio, "How to construct deep recurrent neural networks," in *Proceedings of the Second International Conference on Learning Representations (ICLR 2014)*, 2014.

[21] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[22] Y. Gal and Z. Ghahramani, "A theoretically grounded application of dropout in recurrent neural networks," in *Advances in neural information processing systems*, 2016, pp. 1019–1027.

[23] E. F. T. K. Sang and F. D. Meulder, "Introduction to the conll-2003 shared task: Language-independent named entity recognition," in *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*, ser. CONLL '03. Stroudsburg, PA, USA: Association for Computational Linguistics, 2003, pp. 142–147. [Online]. Available: https://doi.org/10.3115/1119176.1119195

[24] F. Sha and F. Pereira, "Shallow parsing with conditional random fields," in *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, ser. NAACL '03. Stroudsburg, PA, USA: Association for Computational Linguistics, 2003, pp. 134–141. [Online]. Available: https://doi.org/10.3115/1073445.1073473