Multi-Task Learning for Improved Discriminative Training in SMT

Patrick Simianer and **Stefan Riezler**
Department of Computational Linguistics
Heidelberg University
69120 Heidelberg, Germany
{simianer,riezler}@cl.uni-heidelberg.de

Abstract

Multi-task learning has been shown to be effective in various applications, including discriminative SMT. We present an experimental evaluation of the question whether multi-task learning depends on a "natural" division of data into tasks that balance shared and individual knowledge, or whether its inherent regularization makes multi-task learning a broadly applicable remedy against overfitting. To investigate this question, we compare "natural" tasks defined as sections of the International Patent Classification versus "random" tasks defined as random shards in the context of patent SMT. We find that both versions of multi-task learning improve equally well over independent and pooled baselines, and gain nearly 2 BLEU points over standard MERT tuning.

1 Introduction

Multi-task learning is motivated by situations where a number of statistical models need to be estimated from data belonging to different tasks. It is assumed that the data are not completely independent of one another as they share some commonalities, yet they differ enough to counter a simple pooling of data. The goal of multi-task learning is to take advantage of commonalities among tasks by learning a shared model without neglecting individual knowledge. For example, Obozinski et al. (2010) present an optical character recognition scenario where data consist of samples of handwritten characters from several writers. While the styles of different writers vary, it is expected that there are also commonalities on a pixel- or stroke-level that are shared across writers. Chapelle et al. (2011) present a scenario where data from search engine query logs are available for different countries. While the rankings for some queries will have to be country-specific (they cite "football" as a query requiring different rankings in the US and the UK), a large fraction of queries will be country-insensitive. Wäschle and Riezler (2012b) present multi-task learning for statistical machine translation (SMT) of patents from different classes (so-called sections) according to the International Patent Classification (IPC)[1]. While the vocabulary may differ between the different IPC sections, specific legal jargon and a typical textual structure will be shared across IPC sections. As shown in the cited works, treating data from different writers, countries, or IPC classes as data from different tasks, and applying generic multi-task learning to the specific scenario, improves learning results over learning independent or pooled models.

The research question we ask in this paper is as follows: Is multi-task learning dependent on a "natural" task structure in the data, where shared and individual knowledge is properly balanced? Or can multi-task learning be seen as a general regularization technique that prevents overfitting irrespective of the task structure in the data?

We investigate this research question on the example of discriminative training for patent translation, using the algorithm for multi-task learning with ℓ_1/ℓ_2 regularization presented by Simianer et al. (2012). We compare multi-task learning on "natural" tasks given by IPC sections to multi-task learning on "random" tasks given by random shards and to baseline models trained on independent tasks and pooled tasks. We find that both versions of multi-task learning improve over independent or pooled training. However, differences between multi-task learning on IPC tasks and random tasks are small. This points to a more general regularization effect of multi-task learning and indicates a broad applicability of multi-task learning techniques. Another advantage of the ℓ_1/ℓ_2 reg-

[1] http://wipo.int/classifications/ipc/en/

ularization technique of Simianer et al. (2012) is a considerable efficiency gain due to parallelization and iterative feature selection that makes the algorithm suitable for big data applications and for large-scale training with millions of sparse features. Last but not least, our best result for multi-task learning improves by nearly 2 BLEU points over the standard MERT baseline.

2 Related Work

Multi-task learning is an active area in machine learning, dating back at least to Caruana (1997). A regularization perspective was introduced by Evgeniou and Pontil (2004), who formalize the central idea of trading off optimality of parameter vectors for each task-specific model and closeness of these model parameters to the average parameter vector across models in an SVM framework. Equivalent formalizations replace parameter regularization by Bayesian prior distributions on the parameters (Finkel and Manning, 2009) or by augmentation of the feature space with domain independent features (Daumé, 2007). Besides SVMs, several learning algorithms have been extended to the multi-task scenario in a parameter regularization setting, e.g., perceptron-type algorithms (Dredze et al., 2010) or boosting (Chapelle et al., 2011). Further variants include different formalizations of norms for parameter regularization, e.g., ℓ_1/ℓ_2 regularization (Obozinski et al., 2010) or ℓ_1/ℓ_∞ regularization (Quattoni et al., 2009), where only the features that are most important across all tasks are kept in the model.

Early research on multi-task learning for SMT has investigated pooling of IPC sections, with larger pools improving results (Utiyama and Isahara, 2007; Tinsley et al., 2010; Ceauşu et al., 2011). Wäschle and Riezler (2012b) apply multi-task learning to tasks defined as IPC sections and compare patent translation on independent tasks, pooled tasks, and multi-task learning, using same-sized training data. They show small but statistically significant improvements for multi-task learning over independent and pooled training. Duh et al. (2010) introduce random tasks as n-best lists of translations and showed significant improvements by applying various multi-task learning techniques to discriminative reranking. Song et al. (2011) define tasks as bootstrap samples from the development set and show significant improvements for a bagging-based system combination over individual MERT training.

In this paper we apply the multi-task learning technique of Simianer et al. (2012) to tasks defined as IPC sections and to random tasks. Their algorithm can be seen as a weight-based backward feature elimination variant of Obozinski et al. (2010)'s gradient-based forward feature selection algorithm for ℓ_1/ℓ_2 regularization. The latter approach is related to the general methodology of using block norms to select entire groups of features jointly. For example, such groups can be defined as non-overlapping subsets of features (Yuan and Lin, 2006), or as hierarchical groups of features (Zhao et al., 2009), or they can be grouped by the general structure of the prediction problem (Martins et al., 2011). However, these approaches are concerned with grouping features within a single prediction problem whereas multi-task learning adds an orthogonal layer of multiple task-specific prediction problems. By virtue of averaging selected weights after each epoch, the algorithm of Simianer et al. (2012) is related to McDonald et al. (2010)'s iterative mixing procedure. This algorithm is itself related to the bagging procedure of Breiman (1996), if random shards are considered from the perspective of random samples. In both cases averaging helps to reduce the variance of the per-sample classifiers.

3 Multi-task Learning for Discriminative Training in SMT

In multi-task learning, we have data points $\{(x_z^i, y_z^i), i = 1, \ldots, N_z, z = 1, \ldots, Z\}$, sampled from a distribution P_z on $X \times Y$. The subscript z indexes tasks and the superscript i indexes i.i.d. data for each task. For the application of discriminative ranking in SMT, the space X can be thought of as feature representations of n-best translations, and the space Y denotes corresponding sentence-level BLEU scores.[2] We assume that P_z is different for each task but that the P_z's are related as, for example, considered in Evgeniou and Pontil (2004). The standard approach is to fit an independent model involving a D-dimensional parameter vector $\mathbf{w}_z$ for each task z. In multi-task learning, we consider a Z-by-D matrix $\mathbf{W} = (\mathbf{w}_z^d)_{z,d}$ of stacked D-dimensional row vectors $\mathbf{w}_z$, and Z-dimensional column vectors $\mathbf{w}^d$ of weights associated with feature d across tasks. The central al-

[2] See Duh et al. (2010) for a similar formalization for the case of n-best reranking via multi-task learning.

gorithms in most multi-task learning techniques can be characterized as a form of regularization that enforces closeness of task-specific parameter vectors to shared parameter vectors, or promotes sparse models that only contain features that are shared across tasks. In this paper, we will follow the approach of Simianer et al. (2012), who formalize multi-task learning as a distributed feature selection algorithm using ℓ_1/ℓ_2 regularization. ℓ_1/ℓ_2 regularization can be described as penalizing weights $\mathbf{W}$ by the weighted ℓ_1/ℓ_2 norm, which is defined following Obozinski et al. (2010), as

$$\lambda\|\mathbf{W}\|_{1,2} = \lambda \sum_{d=1}^{D} \|\mathbf{w}^d\|_2.$$

Each ℓ_2 norm of a weight column $\mathbf{w}^d$ represents the relevance of the corresponding feature across tasks. The ℓ_1 sum of the ℓ_2 norms enforces a selection of features by encouraging several feature columns $\mathbf{w}^d$ to be $\mathbf{0}$ and others to have high weights across all tasks. This results in shrinking the matrix to the features that are useful across all tasks.

Simianer et al. (2012) achieve this behavior by the following weight-based iterative feature elimination algorithm that is wrapped around a stochastic gradient descent (SGD) algorithm for pairwise ranking (Shen and Joshi, 2005):

Algorithm 1 Multi-task SGD

Get data for Z tasks, each including S sentences; distribute to machines.
Initialize $\mathbf{v} \leftarrow \mathbf{0}$.
for epochs $t \leftarrow 0 \dots T-1$: **do**
  for all tasks $z \in \{1 \dots Z\}$: **parallel do**
    $\mathbf{w}_{z,t,0,0} \leftarrow \mathbf{v}$
    for all sentences $i \in \{0 \dots S-1\}$: **do**
      Decode i^{th} input with $\mathbf{w}_{z,t,i,0}$.
      for all pairs $j \in \{0 \dots P-1\}$: **do**
        $\mathbf{w}_{z,t,i,j+1} \leftarrow \mathbf{w}_{z,t,i,j} - \eta \nabla l_j(\mathbf{w}_{z,t,i,j})$
      end for
      $\mathbf{w}_{z,t,i+1,0} \leftarrow \mathbf{w}_{z,t,i,P}$
    end for
  end for
  Stack weights $\mathbf{W} \leftarrow [\mathbf{w}_{1,t,S,0}| \dots |\mathbf{w}_{Z,t,S,0}]^T$
  Select top K feature columns of $\mathbf{W}$ by ℓ_2 norm
  for $k \leftarrow 1 \dots K$ **do**
    $\mathbf{v}[k] = \frac{1}{Z} \sum_{z=1}^{Z} \mathbf{W}[z][k]$.
  end for
end for
return $\mathbf{v}$

The innermost loop of the algorithm computes an SGD update based on the subgradient ∇l_j of a pairwise loss function. ℓ_1/ℓ_2-based feature selection is done after each epoch of SGD training for each task in parallel. The ℓ_2 norm of the weights is computed for each feature column across tasks; features are sorted by this value; K top features are kept in the model; reduced weight vectors are mixed and the result is re-sent to each task-specific model to start another epoch of parallel training for each task.

We compare two different loss functions for pairwise ranking, one corresponding to the original perceptron algorithm (Rosenblatt, 1958), and an improved version called the margin perceptron (Collobert and Bengio, 2004). To create training data for a pairwise ranking setup, we generate preference pairs by ordering translations according to smoothed sentence-wise BLEU score (Nakov et al., 2012). Let each translation candidate in the n-best list be represented by a feature vector $\mathbf{x} \in \mathbb{R}^D$: For notational convenience, we denote by $\mathbf{x}_j$ a preference pair $\mathbf{x}_j = (\mathbf{x}_j^{(1)}, \mathbf{x}_j^{(2)})$ where $\mathbf{x}_j^{(1)}$ is ordered above $\mathbf{x}_j^{(2)}$ w.r.t. BLEU. Furthermore, we use the shorthand $\bar{\mathbf{x}}_j = \mathbf{x}_j^{(1)} - \mathbf{x}_j^{(2)}$ to denote a D-dimensional difference vector representing an input pattern. For completeness, a label $y = +1$ can be assigned to patterns $\bar{\mathbf{x}}_j$ where $\mathbf{x}_j^{(1)}$ is ordered above $\mathbf{x}_j^{(2)}$ ($y = -1$ otherwise), however, since the ordering relation is antisymmetric, we can consider an ordering in one direction and omit the label entirely.

The original perceptron algorithm is based on the following hinge loss-type objective function:

$$l_j(\mathbf{w}) = (-\langle \mathbf{w}, \bar{\mathbf{x}}_j \rangle)_+$$

where $(a)_+ = \max(0, a)$, $\mathbf{w} \in \mathbb{R}^D$ is a weight vector, and $\langle \cdot, \cdot \rangle$ denotes the standard vector dot product. Instantiating SGD to the stochastic subgradient

$$\nabla l_j(\mathbf{w}) = \begin{cases} -\bar{\mathbf{x}}_j & \text{if } \langle \mathbf{w}, \bar{\mathbf{x}}_j \rangle \leq 0, \\ 0 & \text{else.} \end{cases}$$

leads to the perceptron algorithm for pairwise ranking (Shen and Joshi, 2005).

Collobert and Bengio (2004) presented a version of perceptron learning that includes a margin term in order to control the capacity and thus the generalization performance. Their margin perceptron algorithm follows from applying SGD to the loss function

$$l_j(\mathbf{w}) = (1 - \langle \mathbf{w}, \bar{\mathbf{x}}_j \rangle)_+$$

with the following stochastic subgradient

$$\nabla l_j(\mathbf{w}) = \begin{cases} -\bar{\mathbf{x}}_j & \text{if } \langle \mathbf{w}, \bar{\mathbf{x}}_j \rangle < 1, \\ 0 & \text{else.} \end{cases}$$

Collobert and Bengio (2004) argue that the use of a margin term justifies not using an explicit regularization, thus making the margin perceptron an efficient and effective learning machine.

4 Experiments

4.1 Data & System Setup

For training, development and testing, we use data extracted from the PatTR[3] corpus for the experiments in Wäschle and Riezler (2012b). Training data consists of about 1.2 million German-English parallel sentences. We translate from German into English. German compound words were split using the technique of Koehn and Knight (2003). We use the SCFG decoder cdec (Dyer et al., 2010)[4] and build grammars using its implementation of the suffix array extraction method described in Lopez (2007). Word alignments are built from all parallel data using mgiza[5] and the Moses scripts[6]. SCFG models use the same settings as described in Chiang (2007). We built a modified Kneser-Ney smoothed 5-gram language model using the English side of the training data and performed querying with KenLM (Heafield, 2011)[7].

The International Patent Classification (IPC) categorizes patents hierarchically into eight sections, 120 classes, 600 subclasses, down to 70,000 subgroups at the leaf level. The eight top classes/sections are listed in Table 1.

Typically, a patent belongs to more than one section, with one section chosen as main classification. Our development and test sets for each of the classes, A to H, comprise 2,000 sentences each, originating from a patent with the respective class. These sets were built so that there is no overlap of development sets and test sets, and no overlap between sets of different classes. These eight test sets are referred to as *independent* test sets. Furthermore, we test on a combined set, called *pooled-cat*, that is constructed by concatenating the independent sets. Additionally we use two *pooled* sets for development and testing, each containing 2,000 sentences with all classes evenly represented.

A	Human Necessities
B	Performing Operations, Transporting
C	Chemistry, Metallurgy
D	Textiles, Paper
E	Fixed Constructions
F	Mechanical Engineering, Lighting, Heating, Weapons
G	Physics
H	Electricity

Table 1: IPC top level **sections**.

Our tuning baseline is an implementation of hypergraph MERT (Kumar et al., 2009), directly optimizing IBM BLEU4 (Papineni et al., 2002). Furthermore, we present a regularization baseline by applying ℓ_1 regularization with clipping (Carpenter, 2008; Tsuruoka et al., 2009) to the standard pairwise ranking perceptron. All pairwise ranking methods use a smoothed sentence-wise BLEU+1 score (Nakov et al., 2012) to create gold standard rankings. Our multi-task learning experiments are based on pairwise ranking perceptrons that differ in their objective, corresponding either to the original *perceptron* or to the *margin-perceptron*. Both versions of the perceptron are used for *single-task tuning* and *multi-task tuning*. In the multi-task setting, we compare three different methods for defining a task: "natural" tasks given by *IPC* sections where each *independent* data set is considered as task; "random" tasks, defined by *sharding* where data is shuffled and split once, tasks are kept fixed throughout, and by *resharding* where after an epoch data is shuffled and new random tasks are constructed. In all cases a task/shard is defined to contain 2,000 sentences[8], resulting in eight shards for each setting. The number of features selected after each epoch was set to $K = 100,000$.

For all perceptron runs, the following meta parameters were fixed: A cube pruning pop limit of 200 and non-terminal span limit of 15; 100-best lists with unique entries; constant learning rate; multipartite pair selection. Single-task perceptron runs on *independent* and *pooled* tasks were done

[3] http://www.cl.uni-heidelberg.de/statnlpgroup/pattr

[4] https://github.com/redpony/cdec

[5] http://www.kyloo.net/software/doku.php/mgiza:overview

[6] http://www.statmt.org/moses/?n=Moses.SupportTools

[7] http://kheafield.com/code/kenlm/estimation/

[8] This number is determined by the size of the original development sets; variations of this size did not change results.

	single-task tuning		
	indep. [0]	pooled [1]	pooled-cat [2]
pooled test	–	51.18	51.22
A	54.92	[02]55.27	[0]55.17
B	51.53	51.48	[01]51.69
C	[12]56.31	[2]55.90	55.74
D	49.94	[0]50.33	[0]50.26
E	[1]49.19	48.97	[1]49.13
F	[12]51.26	51.02	51.12
G	[1]49.61	49.44	49.55
H	49.38	49.50	[01]49.67
average test	**51.52**	**51.49**	**51.54**

Table 2: BLEU4 results of MERT baseline using dense features for three different tuning sets: *independent* (separate tuning sets for each IPC class), *pooled* and *pooled-cat* (concatenated *independent* sets). Significant superior performance over other systems in the same row is denoted by prefixed numbers. The first row shows, e.g., that the result of *pooled* [1] is significantly better than *independent* [0], and *pooled-cat* [2].

	single-task tuning		
	indep. [0]	pooled [1]	pooled-cat [2]
pooled test	–	50.75	[1] 52.08
A	[1] 55.11	54.32	[01] 55.94
B	[1] 52.61	50.84	[1] 52.57
C	56.18	56.11	[01] 56.75
D	[1] 50.68	49.48	[01] 51.22
E	[1] 50.27	48.69	[1] 50.01
F	[1] 51.68	50.71	[1] 51.95
G	[1] 49.90	49.06	[01] 50.51
H	[1] 50.48	49.16	[1] 50.53
average test	**52.11**	**51.05**	**52.44**
model size	430,092.5	457,428	1,574,259

Table 3: BLEU4 results for standard *perceptron with ℓ_1 regularization* baseline using sparse rule features, tuned on *independent*, *pooled* and *pooled-cat* sets. Prefixed superscripts denote a significant improvement over the result in the same row indicated by the superscript.

for 15 epochs; multi-task perceptron runs used 10 epochs. Single-task tuning on *pooled-cat* data increases computation time by a factor of 8 which makes this setup infeasible in practice. For the sake of comparison we performed 10 epochs in this setup.

MERT (with default parameters) is used to optimize the weights of 12 dense default features; eight translation model features, a word penalty, the passthrough weight, the language model (LM) score, and an LM out-of-vocabulary penalty. Perceptron training allows to add millions of sparse features which are directly derived from grammar rules: rule shape, rule identifier, bigrams in rule source and target. For a further explanation of these features see Simianer et al. (2012).

For testing we measured IBM BLEU4 on tokenized and lowercased data. Significance results were obtained by approximate randomization tests using the approach of Clark et al. (2011)[9] to account for optimizer instability. Tuning methods with a random component (MERT, randomized experiments) were repeated three times, scores reported in the tables are averaged over optimizer runs.

4.2 Experimental Results

In single-task tuning mode, systems are tuned on the eight *independent* data sets separately, the *pooled* data set, and the independent data sets concatenated (*pooled-cat*). Testing is done on each of the eight IPC sections separately, and on a *pooled test* set of 2,000 sentences where all sections are equally represented. Furthermore, we report **average test** results over runs for all independent data sets.

Results for the MERT baseline are shown in Table 2: Neither pooling nor concatenating independent sets leads to significant performance improvements on all sets with averaged scores being nearly identical.

Evaluation results obtained with the standard *perceptron* algorithm (Table 4) show improvements over MERT in *single-task tuning* mode. The gain on *pooled-cat* data shows that in contrast to MERT training on 12 dense features, discriminative training using large feature sets is able to benefit from large data sets. However, since the *pooled-cat* scenario increases computation time by a factor of 8, it is quite infeasible when used with large sets of sparse features. Single-task tuning on a small set of *pooled* data seems to show overfitting behavior.

Table 3 shows evaluation results for a regularization baseline that applies ℓ_1 *regularization with clipping* to the the single-task tuned standard perceptron in Table 4. We see gains in BLEU on independent and pooled-cat tuning data, but not on the small pooled data set.

Multi-task tuning for the standard perceptron is shown in the right half of Table 4. Because of parallelization, this scenario is as efficient as

[9] https://github.com/jhclark/multeval

	single-task tuning			multi-task tuning		
	indep. [0]	pooled [1]	pooled-cat [2]	IPC [3]	sharding [4]	resharding [5]
pooled test	–	51.33	[1] 51.77	[12] 52.56	[12] 52.54	[12] 52.60
A	54.79	54.76	[01] 55.31	[012] 56.35	[012] 56.22	[012] 56.21
B	[12] 52.45	51.30	[1] 52.19	[012] 52.78	[0123] 52.98	[012] 52.96
C	[2] 56.62	56.65	[1] 56.12	[01245] 57.76	[012] 57.30	[012] 57.44
D	[1] 50.75	49.88	[1] 50.63	[01245] 51.54	[012] 51.33	[012] 51.20
E	[1] 49.70	49.23	[01] 49.92	[012] 50.51	[012] 50.52	[012] 50.38
F	[1] 51.60	51.09	[1] 51.71	[012] 52.28	[012] 52.43	[012] 52.32
G	[1] 49.50	49.06	[01] 49.97	[012] 50.84	[012] 50.88	[012] 50.74
H	[1] 49.77	49.50	[01] 50.64	[012] 51.16	[012] 51.07	[012] 51.10
average test	**51.90**	**51.42**	**52.06**	**52.90**	**52.84**	**52.79**
model size	366,869.4	448,359	1,478,049	100,000	100,000	100,000

Table 4: BLEU4 results for standard *perceptron* algorithm using sparse rule features, tuned in single-task mode on *independent*, *pooled*, and *pooled-cat* sets, and in multi-task mode on eight tasks taken from *IPC* sections or by random *(re)sharding*. Prefixed superscripts denote a significant improvement over the result in the same row indicated by the superscript.

	single-task tuning			multi-task tuning		
	indep. [0]	pooled [1]	pooled-cat [2]	IPC [3]	sharding [4]	resharding [5]
pooled test	–	51.33	[1] 52.58	[12] 52.98	[12] 52.95	[12] 52.99
A	[1] 56.09	55.33	[1] 55.92	[01245]56.78	[012] 56.62	[012] 56.53
B	[1] 52.45	51.59	[1] 52.44	[012]53.31	[012] 53.35	[012] 53.21
C	[1] 57.20	56.85	[01] 57.54	[01]57.46	[1] 57.42	[1] 57.43
D	[1] 50.51	50.18	[01] 51.38	[01245]52.14	[0125] 51.82	[012] 51.66
E	[1] 50.27	49.36	[01] 50.72	[0124]51.13	[012] 50.89	[012] 51.02
F	[1] 52.06	51.20	[01] 52.61	[01245]53.07	[012] 52.80	[012] 52.87
G	[1] 50.00	49.58	[01] 50.90	[01245]51.36	[012] 51.19	[012] 51.11
H	[1] 50.57	49.80	[01] 51.32	[012]51.57	[012] 51.62	[01] 51.47
average test	**52.39**	**51.74**	**52.85**	**53.35**	**53.21**	**53.16**
model size	423,731.5	484,483	1,697,398	100,000	100,000	100,000

Table 5: BLEU4 results for *margin-perceptron* algorithm using sparse rule features, tuned in single-task mode on *independent* tasks, and in multi-task mode on eight tasks taken from *IPC* sections or by random *(re)sharding*. Prefixed superscripts denote a significant improvement over the result in the same row indicated by the superscript.

single-task tuning on small data. We see improvements in BLEU over single-task tuning on small and large tuning data sets. Concerning our initial research questions, we see that the performance difference between "natural" tasks (IPC) and "random" tasks is not conclusive. However, multi-task learning using ℓ_1/ℓ_2 regularization consistently outperforms the standard perceptron under ℓ_1 regularization as shown in Table 3 and MERT tuning as shown in Table 2.

Table 5 shows the evaluation results of the *margin-perceptron* algorithm. Evaluation results on *single-task tuning* show that this algorithm improves over the standard perceptron (Table 4), even in its ℓ_1-regularized version (Table 3), on all tuning sets. Results for *multi-task tuning* show improvements over the same scenario for the standard perceptron (Table 4). This means that the improvements due to the orthogonal regularization techniques in example space and feature space, namely large-margin learning and multi-task learning, add up. A comparison between *single-task* and *multi-task tuning* modes of the margin-perceptron shows a gain for the latter scenarios. Differences between multi-task learning on *IPC* classes versus random *sharding* or *resharding* are again small, with the best overall result obtained by multi-task learning of the margin-perceptron on IPC classes.

Overall, our best multi-task learing result is nearly 2 BLEU points better than MERT training. The algorithm to achieve this result is efficient due

to parallelization and due to iterative feature selection. As shown in the last rows of Tables 3-5 , the average size is around 400K features for independently tuned models and around 1.6M features for models tuned on pooled-cat data. In multi-task learning, models can be iteratively cut to 100K shared features whose weights are tuned in parallel.

5 Conclusion

We presented an experimental investigation of the question whether the power of multi-task learning depends on data structured along tasks that exhibit a proper balance of shared and individual knowledge, or whether its inherent feature selection and regularization makes multi-task learning a widely applicable remedy against overfitting. We compared multi-task patent SMT for "natural" tasks of IPC sections and "random" tasks of shards in distributed learning. Both versions of multi-task learning yield significant improvements over independent and pooled training, however, the difference between "natural" and "random" tasks is marginal. This is an indication for the usefulness of multi-task learning as a generic regularization tool. Considering also the efficiency gained by iterative feature selection, the ℓ_1/ℓ_2 regularization algorithm presented in Simianer et al. (2012) presents itself as an efficient and effective learning algorithm for general big data and sparse feature applications. Furthermore, the improvements by multi-task feature selection add up with improvements by large-margin learning, delivering overall improvements of nearly 2 BLEU points over the standard MERT baseline.

Our research question regarding the superiority of "natural" or "random" tasks was shown to be undetermined for the application of patent translation. The obvious question for future work is if and how a task division can be found that improves multi-task learning over our current results. Such an investigation will have to explore various similarity metrics and clustering techniques for IPC sub-classes (Wäschle and Riezler, 2012a), e.g., for the goal of optimizing clustering with respect to the ratio of between-cluster to within-cluster similarity for a given metric. However, the final criterion for the usefulness of a clustering is necessarily application specific (von Luxburg et al., 2012), in our case specific to patent translation performance. Nevertheless, we hope that the presented and future work will prove useful and generalizable for related multi-task learning scenarios.

Acknowledgments

The research presented in this paper was supported in part by DFG grant "Cross-language Learning-to-Rank for Patent Retrieval".

References

Leo Breiman. 1996. Bagging predictors. *Machine Learning*, 24:123–140.

Bob Carpenter. 2008. Lazy sparse stochastic gradient descent for regularized multinomial logistic regression. Technical report, Alias-i.

Rich Caruana. 1997. Multitask learning. *Journal of Machine Learning Research*, 28.

Alexandru Ceauşu, John Tinsley, Jian Zhang, and Andy Way. 2011. Experiments on domain adaptation for patent machine translation in the PLuTO project. In *Proceedings of the 15th Conference of the European Association for Machine Translation (EAMT 2011)*, Leuven, Belgium.

Olivier Chapelle, Pannagadatta Shivaswamy, Srinivas Vadrevu, Kilian Weinberger, Ya Zhang, and Belle Tseng. 2011. Boosted multi-task learning. *Machine Learning*.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33.

Jonathan Clark, Chris Dyer, Alon Lavie, and Noah Smith. 2011. Better hypothesis testing for statistical machine translation: Controlling for optimizer instability. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL'11)*, Portland, OR.

Ronan Collobert and Samy Bengio. 2004. Links between perceptrons, MLPs, and SVMs. In *Proceedings of the 21st International Conference on Machine Learning (ICML'04)*, Banff, Canada.

Hal Daumé. 2007. Frustratingly easy domain adaptation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL'07)*, Prague, Czech Republic.

Mark Dredze, Alex Kulesza, and Koby Crammer. 2010. Multi-domain learning by confidence-weighted parameter combination. *Machine Learning*, 79:123–149.

Kevin Duh, Katsuhito Sudoh, Hajime Tsukada, Hideki Isozaki, and Masaaki Nagata. 2010. N-best reranking by multitask learning. In *Proceedings of the 5th Joint Workshop on Statistical Machine Translation and MetricsMATR*, Uppsala, Sweden.

Chris Dyer, Adam Lopez, Juri Ganitkevitch, Johnathan Weese, Ferhan Ture, Phil Blunsom, Hendra Setiawan, Vladimir Eidelman, and Philip Resnik. 2010. cdec: A decoder, alignment, and learning framework for finite-state and context-free translation models. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL'10)*.

Theodoros Evgeniou and Massimiliano Pontil. 2004. Regularized multi-task learning. In *Proceedings of the 10th ACM SIGKDD conference on knowledge discovery and data mining (KDD'04)*, Seattle, WA.

Jenny Rose Finkel and Christopher D. Manning. 2009. Hierarchical Bayesian domain adaptation. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics - Human Language Technologies (NAACL-HLT'09)*, Boulder, CO.

Kenneth Heafield. 2011. KenLM: faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation (WMT'11)*, Edinburgh, UK.

Philipp Koehn and Kevin Knight. 2003. Empirical methods for compound splitting. In *Proceedings of the 10th conference on European chapter of the Association for Computational Linguistics (EACL'03)*, Budapest, Hungary.

Shankar Kumar, Wolfgang Macherey, Chris Dyer, and Franz Och. 2009. Efficient minimum error rate training and minimum Bayes-risk decoding for translation hypergraphs and lattices. In *Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th IJCNLP of the AFNLP (ACL-IJCNLP'09*, Suntec, Singapore.

Adam Lopez. 2007. Hierarchical phrase-based translation with suffix arrays. In *Proceedings of EMNLP-CoNLL*, Prague, Czech Republic.

André F. T. Martins, Noah A. Smith, Pedro M. Q. Aguiar, and Mário A. T. Figueiredo. 2011. Structured sparsity in structured prediction. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, Edinburgh, Scotland.

Ryan McDonald, Keith Hall, and Gideon Mann. 2010. Distributed training strategies for the structured perceptron. In *Proceedings of Human Language Technologies: The 11th Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT'10)*, Los Angeles, CA.

Preslav Nakov, Francisco Guzmán, and Stephan Vogel. 2012. Optimizing for sentence-level bleu+1 yields short translations. In *Proceedings of the 24th International Conference on Computational Linguistics (COLING 2012)*, Bombay, India.

Guillaume Obozinski, Ben Taskar, and Michael I. Jordan. 2010. Joint covariate selection and joint subspace selection for multiple classification problems. *Statistics and Computing*, 20:231–252.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ariadna Quattoni, Xavier Carreras, Michael Collins, and Trevor Darrell. 2009. An efficient projection for $\ell_{1,\infty}$ regularization. In *Proceedings of the 26th International Conference on Machine Learning (ICML'09)*, Montreal, Canada.

Frank Rosenblatt. 1958. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6).

Libin Shen and Aravind K. Joshi. 2005. Ranking and reranking with perceptron. *Journal of Machine Learning Research*, 60(1-3):73–96.

Patrick Simianer, Stefan Riezler, and Chris Dyer. 2012. Joint feature selection in distributed stochastic learning for large-scale discriminative training in SMT. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL 2012)*, Jeju, Korea.

Linfeng Song, Haitao Mi, Yajuan Lü, and Qun Liu. 2011. Bagging-based system combination for domain adaptation. In *Proceedings of MT Summit XIII*, Xiamen, China.

John Tinsley, Andy Way, and Paraic Sheridan. 2010. PLuTO: MT for online patent translation. In *Proceedings of the 9th Conference of the Association for Machine Translation in the Americas (AMTA 2010)*, Denver, CO.

Yoshimasa Tsuruoka, Jun'ichi Tsujii, and Sophia Ananiadou. 2009. Stochastic gradient descent training for ℓ_1-regularized log-linear models with cumulative penalty. In *Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics (ACL-IJCNLP'09)*, Singapore.

Masao Utiyama and Hitoshi Isahara. 2007. A Japanese-English patent parallel corpus. In *Proceedings of MT Summit XI*, Copenhagen, Denmark.

Ulrike von Luxburg, Robert C. Williamson, and Isabelle Guyon. 2012. Clustering: Science or art? In *Proceedings of the ICML 2011 Workshop on Unsupervised and Transfer Learning*, Bellevue, WA.

Katharina Wäschle and Stefan Riezler. 2012a. Analyzing parallelism and domain similarities in the MAREC patent corpus. In *Proceedings of the 5th Information Retrieval Facility Conference (IRFC 2012)*, Vienna, Austria.

Katharina Wäschle and Stefan Riezler. 2012b. Structural and topical dimensions in multi-task patent translation. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, Avignon, France.

Ming Yuan and Yi Lin. 2006. Model selection and estimation in regression with grouped variables. *J.R.Statist.Soc.B*, 68(1):49–67.

Peng Zhao, Guilherme Rocha, and Bin Yu. 2009. The composite absolute penalties family for grouped and hierarchical variable selection. *The Annals of Statistics*, 37(6A):3468–3497.