Joint Feature Selection in Distributed Stochastic Learning for Large-Scale Discriminative Training in SMT

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

Chris Dyer
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA, 15213, USA
cdyer@cs.cmu.edu

Abstract

With a few exceptions, discriminative training in statistical machine translation (SMT) has been content with tuning weights for large feature sets on small development data. Evidence from machine learning indicates that increasing the training sample size results in better prediction. The goal of this paper is to show that this common wisdom can also be brought to bear upon SMT. We deploy local features for SCFG-based SMT that can be read off from rules at runtime, and present a learning algorithm that applies ℓ_1/ℓ_2 regularization for joint feature selection over distributed stochastic learning processes. We present experiments on learning on 1.5 million training sentences, and show significant improvements over tuning discriminative models on small development sets.

1 Introduction

The standard SMT training pipeline combines scores from large count-based translation models and language models with a few other features and tunes these using the well-understood line-search technique for error minimization of Och (2003). If only a handful of dense features need to be tuned, minimum error rate training can be done on small tuning sets and is hard to beat in terms of accuracy and efficiency. In contrast, the promise of large-scale discriminative training for SMT is to scale to arbitrary types and numbers of features and to provide sufficient statistical support by parameter estimation on large sample sizes. Features may be lexicalized and sparse, non-local and overlapping, or be designed to generalize beyond surface statistics by incorporating part-of-speech or syntactic labels. The modeler's goals might be to identify complex properties of translations, or to counter errors of pre-trained translation models and language models by explicitly down-weighting translations that exhibit certain undesired properties. Various approaches to feature engineering for discriminative models have been presented (see Section 2), however, with a few exceptions, discriminative learning in SMT has been confined to training on small tuning sets of a few thousand examples. This contradicts theoretical and practical evidence from machine learning that suggests that larger training samples should be beneficial to improve prediction also in SMT. Why is this?

One possible reason why discriminative SMT has mostly been content with small tuning sets lies in the particular design of the features themselves. For example, the features introduced by Chiang et al. (2008) and Chiang et al. (2009) for an SCFG model for Chinese/English translation are of two types: The first type explicitly counters overestimates of rule counts, or rules with bad overlap points, bad rewrites, or with undesired insertions of target-side terminals. These features are specified in hand-crafted lists based on a thorough analysis of a tuning set. Such finely hand-crafted features will find sufficient statistical support on a few thousand examples and thus do not benefit from larger training sets. The second type of features deploys external information such as syntactic parses or word alignments to penalize bad reorderings or undesired translations of phrases that cross syntactic constraints. At large scale, extraction of such features quickly becomes

Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics, pages 11–21,
Jeju, Republic of Korea, 8-14 July 2012.

(1) $X \rightarrow$ X_1 hat X_2 versprochen, X_1 promised X_2
(2) $X \rightarrow$ X_1 hat mir X_2 versprochen,
X_1 promised me X_2
(3) $X \rightarrow$ X_1 versprach X_2, X_1 promised X_2

Figure 1: SCFG rules for translation.

infeasible because of costly generation and storage of linguistic annotations. Another possible reason why large training data did not yet show the expected improvements in discriminative SMT is a special overfitting problem of current popular online learning techniques. This is due to stochastic learning on a per-example basis where a weight update on a misclassified example may apply only to a small fraction of data that have been seen before. Thus many features will not generalize well beyond the training examples on which they were introduced.

The goal of this paper is to investigate if and how it is possible to benefit from scaling discriminative training for SMT to large training sets. We deploy generic features for SCFG-based SMT that can efficiently be read off from rules at runtime. Such features include rule ids, rule-local n-grams, or types of rule shapes. Another crucial ingredient of our approach is a combination of parallelized stochastic learning with feature selection inspired by multi-task learning. The simple but effective idea is to randomly divide training data into evenly sized shards, use stochastic learning on each shard in parallel, while performing ℓ_1/ℓ_2 regularization for joint feature selection on the shards after each epoch, before starting a new epoch with a reduced feature vector averaged across shards. Iterative feature selection procedure is the key to both efficiency and improved prediction: Without interleaving parallelized stochastic learning with feature selection our largest experiments would not be feasible. Selecting features jointly across shards and averaging does counter the overfitting effect that is inherent to stochastic updating. Our resulting models are learned on large data sets, but they are small and outperform models that tune feature sets of various sizes on small development sets. Our software is freely available as a part of the `cdec`[1] framework.

[1] `https://github.com/redpony/cdec`

2 Related Work

The great promise of discriminative training for SMT is the possibility to design arbitrarily expressive, complex, or overlapping features in great numbers. The focus of many approaches thus has been on feature engineering and on adaptations of machine learning algorithms to the special case of SMT (where gold standard rankings have to be created automatically). Examples for adapted algorithms include Maximum-Entropy Models (Och and Ney, 2002; Blunsom et al., 2008), Pairwise Ranking Perceptrons (Shen et al., 2004; Watanabe et al., 2006; Hopkins and May, 2011), Structured Perceptrons (Liang et al., 2006a), Boosting (Duh and Kirchhoff, 2008; Wellington et al., 2009), Structured SVMs (Tillmann and Zhang, 2006; Hayashi et al., 2009), MIRA (Watanabe et al., 2007; Chiang et al., 2008; Chiang et al., 2009), and others. Adaptations of the loss functions underlying such algorithms to SMT have recently been described as particular forms of ramp loss optimization (McAllester and Keshet, 2011; Gimpel and Smith, 2012).

All approaches have been shown to scale to large feature sets and all include some kind of regularization method. However, most approaches have been confined to training on small tuning sets. Exceptions where discriminative SMT has been used on large training data are Liang et al. (2006a) who trained 1.5 million features on 67,000 sentences, Blunsom et al. (2008) who trained 7.8 million rules on 100,000 sentences, or Tillmann and Zhang (2006) who used 230,000 sentences for training.

Our approach is inspired by Duh et al. (2010) who applied multi-task learning for improved generalization in n-best reranking. In contrast to our work, Duh et al. (2010) did not incorporate multi-task learning into distributed learning, but defined tasks as n-best lists, nor did they develop new algorithms, but used off-the-shelf multi-task tools.

3 Local Features for Synchronous CFGs

The work described in this paper is based on the SMT framework of hierarchical phrase-based translation (Chiang, 2005; Chiang, 2007). Translation rules are extracted from word-aligned parallel sentences and can be seen as productions of a synchronous CFG. Examples are rules like (1)-(3)

shown in Figure 1. Local features are designed to be readable directly off the rule at decoding time. We use three rule templates in our work:

Rule identifiers: These features identify each rule by a unique identifier. Such features correspond to the relative frequencies of rewrites rules used in standard models.

Rule n-grams: These features identify n-grams of consecutive items in a rule. We use bigrams on source-sides of rules. Such features identify possible source side phrases and thus can give preference to rules including them.[2]

Rule shape: These features are indicators that abstract away from lexical items to templates that identify the location of sequences of terminal symbols in relation to non-terminal symbols, on both the source- and target-sides of each rule used. For example, both rules (1) and (2) map to the same indicator, namely that a rule is being used that consists of a (NT, term*, NT, term*) pattern on its source side, and an (NT, term*, NT) pattern on its target side. Rule (3) maps to a different template, that of (NT, term*, NT) on source and target sides.

4 Joint Feature Selection in Distributed Stochastic Learning

The following discussion of learning methods is based on pairwise ranking in a Stochastic Gradient Descent (SGD) framework. The resulting algorithms can be seen as variants of the perceptron algorithm. Let each translation candidate be represented by a feature vector $\mathbf{x} \in \mathbb{R}^D$ where preference pairs for training are prepared by sorting translations according to smoothed sentence-wise BLEU score (Liang et al., 2006a) against the reference. For a preference pair $\mathbf{x}_j = (\mathbf{x}_j^{(1)}, \mathbf{x}_j^{(2)})$ where $\mathbf{x}_j^{(1)}$ is preferred over $\mathbf{x}_j^{(2)}$, and $\bar{\mathbf{x}}_j = \mathbf{x}_j^{(1)} - \mathbf{x}_j^{(2)}$, we consider 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 following stochastic subgradient leads to the perceptron algorithm for pairwise ranking[3] (Shen and Joshi, 2005):

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

Our baseline algorithm 1 (SDG) scales pairwise ranking to large scale scenarios. The algorithm takes an average over the final weight updates of each epoch instead of keeping a record of all weight updates for final averaging (Collins, 2002) or for voting (Freund and Schapire, 1999).

Algorithm 1 SGD: int I, T, float η

```
Initialize w_{0,0,0} ← 0.
for epochs t ← 0...T−1: do
    for all i ∈ {0...I−1}: do
        Decode i^th input with w_{t,i,0}.
        for all pairs x_j, j ∈ {0...P−1}: do
            w_{t,i,j+1} ← w_{t,i,j} − η∇l_j(w_{t,i,j})
        end for
        w_{t,i+1,0} ← w_{t,i,P}
    end for
    w_{t+1,0,0} ← w_{t,I,0}
end for
return (1/T) Σ_{t=1}^{T} w_{t,0,0}
```

While stochastic learning exhibits a runtime behavior that is linear in sample size (Bottou, 2004), very large datasets can make sequential processing infeasible. Algorithm 2 (MixSGD) addresses this problem by parallelization in the framework of MapReduce (Dean and Ghemawat, 2004).

Algorithm 2 MixSGD: int I, T, Z, float η

```
Partition data into Z shards, each of size S ← I/Z;
distribute to machines.
for all shards z ∈ {1...Z}: parallel do
    Initialize w_{z,0,0,0} ← 0.
    for epochs t ← 0...T−1: do
        for all i ∈ {0...S−1}: do
            Decode i^th input with w_{z,t,i,0}.
            for all pairs x_j, j ∈ {0...P−1}: do
                w_{z,t,i,j+1} ← w_{z,t,i,j} − η∇l_j(w_{z,t,i,j})
            end for
            w_{z,t,i+1,0} ← w_{z,t,i,P}
        end for
        w_{z,t+1,0,0} ← w_{z,t,S,0}
    end for
end for
Collect final weights from each machine,
return (1/Z) Σ_{z=1}^{Z} ( (1/T) Σ_{t=1}^{T} w_{z,t,0,0} ).
```

[2] Similar "monolingual parse features" have been used in Dyer et al. (2011).

[3] Other loss functions lead to stochastic versions of SVMs (Collobert and Bengio, 2004; Shalev-Shwartz et al., 2007; Chapelle and Keerthi, 2010).

Algorithm 2 is a variant of the SimuParallelSGD algorithm of Zinkevich et al. (2010) or equivalently of the parameter mixing algorithm of McDonald et al. (2010). The key idea of algorithm 2 is to partition the data into disjoint shards, then train SGD on each shard in parallel, and after training mix the final parameters from each shard by averaging. The algorithm requires no communication between machines until the end.

McDonald et al. (2010) also present an iterative mixing algorithm where weights are mixed from each shard after training a single epoch of the perceptron in parallel on each shard. The mixed weight vector is re-sent to each shard to start another epoch of training in parallel on each shard. This algorithm corresponds to our algorithm 3 (IterMixSGD).

Algorithm 3 IterMixSGD: int I, T, Z, float η

Partition data into Z shards, each of size $S \leftarrow I/Z$; distribute to machines.
Initialize $\mathbf{v} \leftarrow \mathbf{0}$.
for epochs $t \leftarrow 0 \dots T-1$: **do**
  for all shards $z \in \{1 \dots Z\}$: **parallel do**
    $\mathbf{w}_{z,t,0,0} \leftarrow \mathbf{v}$
    for all $i \in \{0 \dots S-1\}$: **do**
      Decode i^{th} input with $\mathbf{w}_{z,t,i,0}$.
      for all pairs $x_j, j \in \{0 \dots P-1\}$: **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
  Collect weights $\mathbf{v} \leftarrow \frac{1}{Z} \sum_{z=1}^{Z} \mathbf{w}_{z,t,S,0}$.
end for
return v

Parameter mixing by averaging will help to ease the feature sparsity problem, however, keeping feature vectors on the scale of several million features in memory can be prohibitive. If network latency is a bottleneck, the increased amount of information sent across the network after each epoch may be a further problem.

Our algorithm 4 (IterSelSGD) introduces feature selection into distributed learning for increased efficiency and as a more radical measure against overfitting. The key idea is to view shards as tasks, and to apply methods for joint feature selection from multi-task learning to achieve small sets of features that are useful across all tasks or shards. Our algorithm represents weights in a Z-by-D matrix $\mathbf{W} = [\mathbf{w}_{z_1} | \dots | \mathbf{w}_{z_Z}]^T$ of stacked D-dimensional weight vectors across Z shards. We compute the ℓ_2 norm of the weights in each feature column, sort features by this value, and keep K features in the model. This feature selection procedure is done after each epoch. Reduced weight vectors are mixed and the result is re-sent to each shard to start another epoch of parallel training on each shard.

Algorithm 4 IterSelSGD: int I, T, Z, K, float η

Partition data into Z shards, each of size $S = I/Z$; distribute to machines.
Initialize $\mathbf{v} \leftarrow \mathbf{0}$.
for epochs $t \leftarrow 0 \dots T-1$: **do**
  for all shards $z \in \{1 \dots Z\}$: **parallel do**
    $\mathbf{w}_{z,t,0,0} \leftarrow \mathbf{v}$
    for all $i \in \{0 \dots S-1\}$: **do**
      Decode i^{th} input with $\mathbf{w}_{z,t,i,0}$.
      for all pairs $x_j, j \in \{0 \dots P-1\}$: **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
  Collect/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 and
  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 v

This algorithm can be seen as an instance of ℓ_1/ℓ_2 regularization as follows: Let w_d be the dth column vector of $\mathbf{W}$, representing the weights for the dth feature across tasks/shards. ℓ_1/ℓ_2 regularization penalizes weights $\mathbf{W}$ by the weighted ℓ_1/ℓ_2 norm

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

Each ℓ_2 norm of a weight column represents the relevance of the corresponding feature across tasks/shards. The ℓ_1 sum of the ℓ_2 norms enforces a selection among features based on these norms. Consider for example the two 5-feature, 3-task weight matrices in Figure 2. Assuming the same loss for both matrices, the right-hand side matrix is preferred because of a smaller ℓ_1/ℓ_2 norm (12 instead of 18). This matrix shares features across tasks which leads to larger ℓ_2 norms for some columns (here $\|w_1\|_2$ and $\|w_2\|_2$) and forces other columns to zero. This results in shrinking the matrix to those features that are useful across all tasks.

	w_1	w_2	w_3	w_4	w_5			w_1	w_2	w_3	w_4	w_5	
$\mathbf{w}_{z_1}$	[6	4	0	0	0	]	[	6	4	0	0	0	]
$\mathbf{w}_{z_2}$	[0	0	3	0	0	]	[	3	0	0	0	0	]
$\mathbf{w}_{z_3}$	[0	0	0	2	3	]	[	2	3	0	0	0	]
column ℓ_2 norm:	6	4	3	2	3			7	5	0	0	0	
ℓ_1 sum:					$\Rightarrow$	18						$\Rightarrow$	12

Figure 2: ℓ_1/ℓ_2 regularization enforcing feature selection.

Our algorithm is related to Obozinski et al. (2010)'s approach to ℓ_1/ℓ_2 regularization where feature columns are incrementally selected based on the ℓ_2 norms of the gradient vectors corresponding to feature columns. Their algorithm is itself an extension of gradient-based feature selection based on the ℓ_1 norm, e.g., Perkins et al. (2003).[4] In contrast to these approaches we approximate the gradient by using the weights given by the ranking algorithm itself. This relates our work to weight-based recursive feature elimination (RFE) (Lal et al., 2006). Furthermore, algorithm 4 performs feature selection based on a choice of meta-parameter of K features instead of by thresholding a regularization meta-parameter λ, however, these techniques are equivalent and can be transformed into each other.

5 Experiments

5.1 Data, Systems, Experiment Settings

The datasets used in our experiments are versions of the News Commentary (*nc*), News Crawl (*crawl*) and Europarl (*ep*) corpora described in Table 1. The translation direction is German-to-English.

The SMT framework used in our experiments is hierarchical phrase-based translation (Chiang, 2007). We use the `cdec` decoder[5] (Dyer et al., 2010) and induce SCFG grammars from two sets of symmetrized alignments using the method described by Chiang (2007). All data was tokenized and lowercased; German compounds were split (Dyer, 2009). For word alignment of the news-commentary data, we used GIZA++ (Och and Ney, 2000); for aligning the Europarl data, we used the Berkeley aligner (Liang et al., 2006b). Before training, we collect all the grammar rules necessary to translate each individual sentence into separate files (so-called per-sentence grammars) (Lopez, 2007). When decoding, `cdec` loads the appropriate file immediately prior to translation of the sentence. The computational overhead is minimal compared to the expense of decoding. Also, deploying disk space instead of memory fits perfectly into the MapReduce framework we are working in. Furthermore, the extraction of grammars for training is done in a leave-one-out fashion (Zollmann and Sima'an, 2005) where rules are extracted for a parallel sentence pair only if the same rules are found in other sentences of the corpus as well.

3-gram (news-commentary) and 5-gram (Europarl) language models are trained on the data described in Table 1, using the SRILM toolkit (Stolcke, 2002) and binarized for efficient querying using kenlm (Heafield, 2011). For the 5-gram language models, we replaced every word in the lm training data with `<unk>` that did not appear in the English part of the parallel training data to build an open vocabulary language model.

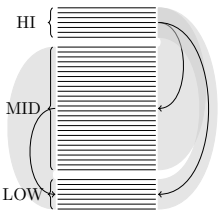

Figure 3: Multipartite pairwise ranking.

Training data for discriminative learning are prepared by comparing a 100-best list of translations against a single reference using smoothed per-sentence BLEU (Liang et al., 2006a). From the BLEU-reordered n-best list, translations were put into sets for the top 10% level (HI), the middle 80% level (MID), and the bottom 10% level (LOW). These level sets are used for multipartite ranking

[4] Note that by definition of $\|\mathbf{W}\|_{1,2}$, standard ℓ_1 regularization is a special case of ℓ_1/ℓ_2 regularization for a single task.

[5] `cdec` metaparameters were set to a non-terminal span limit of 15 and standard cube pruning with a pop limit of 200.

News Commentary(*nc*)					
	train-*nc*	**lm-train-*nc***	**dev-*nc***	**devtest-*nc***	**test-*nc***
Sentences	132,753	180,657	1057	1064	2007
Tokens *de*	3,530,907	–	27,782	28,415	53,989
Tokens *en*	3,293,363	4,394,428	26,098	26,219	50,443
Rule Count	14,350,552 (1G)	–	2,322,912	2,320,264	3,274,771
Europarl(*ep*)					
	train-*ep*	**lm-train-*ep***	**dev-*ep***	**devtest-*ep***	**test-*ep***
Sentences	1,655,238	2,015,440	2000	2000	2000
Tokens *de*	45,293,925	–	57,723	56,783	59,297
Tokens *en*	45,374,649	54,728,786	58,825	58,100	60,240
Rule Count	203,552,525 (31.5G)	–	17,738,763	17,682,176	18,273,078
News Crawl(*crawl*)					
			dev-*crawl*	**test-*crawl10***	**test-*crawl11***
Sentences			2051	2489	3003
Tokens *de*			49,848	64,301	76,193
Tokens *en*			49,767	61,925	74,753
Rule Count			9,404,339	11,307,304	12,561,636

Table 1: Overview of data used for train/dev/test. News Commentary (*nc*) and Europarl (*ep*) training data and also News Crawl (*crawl*) dev/test data were taken from the WMT11 translation task (`http://statmt.org/wmt11/translation-task.html`). The dev/test data of *nc* are the sets provided with the WMT07 shared task (`http://statmt.org/wmt07/shared-task.html`). *Ep* dev/test data is from WMT08 shared task (`http://statmt.org/wmt08/shared-task.html`). The numbers in brackets for the rule counts of *ep*/*nc* training data are total counts of rules in the per-sentence grammars.

where translation pairs are built between the elements in HI-MID, HI-LOW, and MID-LOW, but not between translations inside sets on the same level. This idea is depicted graphically in Figure 3. The intuition is to ensure that good translations are preferred over bad translations without teasing apart small differences.

For evaluation, we used the `mteval-v11b.pl` script to compute lowercased BLEU-4 scores (Papineni et al., 2001). Statistical significance was measured using an Approximate Randomization test (Noreen, 1989; Riezler and Maxwell, 2005).

All experiments for training on dev sets were carried out on a single computer. For grammar extraction and training of the full data set we used a 30 node hadoop Map/Reduce cluster that can handle 300 jobs at once. We split the data into 2290 shards for the *ep* runs and 141 shards for the *nc* runs, each shard holding about 1,000 sentences, which corresponds to the dev set size of the *nc* data set.

5.2 Experimental Results

The baseline learner in our experiments is a pairwise ranking perceptron that is used on various features and training data and plugged into various meta-

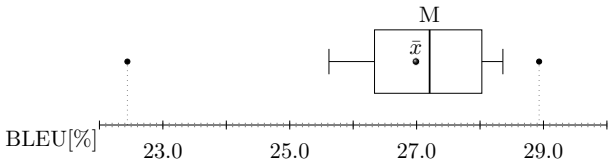

Figure 4: Boxplot of BLEU-4 results for 100 runs of MIRA on news commentary data, depicting median (M), mean ($\bar{x}$), interquartile range (box), standard deviation (whiskers), outliers (end points).

algorithms for distributed processing. The perceptron algorithm itself compares favorably to related learning techniques such as the MIRA adaptation of Chiang et al. (2008). Figure 4 gives a boxplot depicting BLEU-4 results for 100 runs of the MIRA implementation of the `cdec` package, tuned on dev-*nc*, and evaluated on the respective test set test-*nc*.[6] We see a high variance (whiskers denote standard deviations) around a median of 27.2 BLEU and a mean of 27.1 BLEU. The fluctuation of results is due to sampling training examples from the translation hy-

[6]MIRA was used with default meta parameters: 250 hypothesis list to search for oracles, regularization strength $C = 0.01$ and using 15 passes over the input. It optimized IBM BLEU-4. The initial weight vector was $\mathbf{0}$.

Algorithm	Tuning set	Features	#Features	devtest-*nc*	test-*nc*
MIRA	dev-*nc*	default	12	–	27.10
1	dev-*nc*	default	12	**25.88**	28.0
	dev-*nc*	+id	137k	25.53	$27.6^{\dagger 23}$
	dev-*nc*	+ng	29k	25.82	$27.42^{\dagger 234}$
	dev-*nc*	+shape	51	25.91	28.1
	dev-*nc*	+id,ng,shape	180k	25.71	$\mathbf{28.15}^{34}$
2	train-*nc*	default	12	25.73	27.86
	train-*nc*	+id	4.1M	25.13	$27.19^{\dagger 134}$
	train-*nc*	+ng	354k	**26.09**	$\mathbf{28.03}^{134}$
	train-*nc*	+shape	51	26.07	27.91^{3}
	train-*nc*	+id,ng,shape	4.7M	26.08	27.86^{34}
3	train-*nc*	default	12	26.09 @2	$27.94^{\dagger}$
	train-*nc*	+id	3.4M	26.1 @4	$27.97^{\dagger 12}$
	train-*nc*	+ng	330k	26.33 @4	28.34^{12}
	train-*nc*	+shape	51	26.39 @9	28.31^{2}
	train-*nc*	+id,ng,shape	4.7M	**26.42** @9	$\mathbf{28.55}^{124}$
4	train-*nc*	+id	100k	25.91 @7	$27.82^{\dagger 2}$
	train-*nc*	+ng	100k	26.42 @4	$28.37^{\dagger 12}$
	train-*nc*	+id,ng,shape	100k	**26.8** @8	$\mathbf{28.81}^{123}$

Table 2: BLEU-4 results for algorithms 1 (SGD), 2 (MixSGD), 3 (IterMixSDG), and 4 (IterSelSGD) on news-commentary (*nc*) data. Feature groups are 12 dense features (default), rule identifiers (id), rule n-gram (ng), and rule shape (shape). Statistical significance at p-level < 0.05 of a result difference on the test set to a different algorithm applied to the same feature group is indicated by raised algorithm number. † indicates statistically significant differences to best result across features groups for same algorithm, indicated in **bold face**. @ indicates the optimal number of epochs chosen on the devtest set.

pergraph as is done in the `cdec` implementation of MIRA. We found similar fluctuations for the `cdec` implementations of PRO (Hopkins and May, 2011) or hypergraph-MERT (Kumar et al., 2009) both of which depend on hypergraph sampling. In contrast, the perceptron is deterministic when started from a zero-vector of weights and achieves favorable 28.0 BLEU on the news-commentary test set. Since we are interested in relative improvements over a stable baseline, we restrict our attention in all following experiments to the perceptron.[7]

Table 2 shows the results of the experimental comparison of the 4 algorithms of Section 4. The default features include 12 dense models defined on SCFG rules;[8] The sparse features are the 3 templates described in Section 3. All feature weights were tuned together using algorithms 1-4. If not indicated otherwise, the perceptron was run for 10 epochs with learning rate $\eta = 0.0001$, started at zero weight vector, using deduplicated 100-best lists.

The results on the news-commentary (*nc*) data show that training on the development set does not benefit from adding large feature sets – BLEU result differences between tuning 12 default features

[7] Absolute improvements would be possible, e.g., by using larger language models or by adding news data to the *ep* training set when evaluating on *crawl* test sets (see, e.g., Dyer et al. (2011)), however, this is not the focus of this paper.

[8] negative log relative frequency $p(e|f)$; log count(f); log count(e, f); lexical translation probability $p(f|e)$ and $p(e|f)$ (Koehn et al., 2003); indicator variable on singleton phrase e; indicator variable on singleton phrase pair f, e; word penalty; language model weight; OOV count of language model; number of untranslated words; Hiero glue rules (Chiang, 2007).

Alg.	Tuning set	Features	#Feats	devtest-*ep*	test-*ep*	Tuning set	test-*crawl*10	test-*crawl*11
1	dev-*ep*	default	12	25.62	26.42†	dev-*crawl*	15.39†	14.43†
	dev-*ep*	+id,ng,shape	300k	**27.84**	**28.37**	dev-*crawl*	**17.8**^{4}	**16.83**^{4}
4	train-*ep*	+id,ng,shape	100k	**28.0** @9	**28.62**	train-*ep*	**19.12**^{1}	**17.33**^{1}

Table 3: BLEU-4 results for algorithms 1 (SGD) and 4 (IterSelSGD) on Europarl (*ep*) and news crawl (*crawl*) test data. Feature groups are 12 dense features (default), rule identifiers (id), rule n-gram (ng), and rule shape (shape). Statistical significance at p-level < 0.05 of a result difference on the test set to a different algorithm applied to the same feature group is indicated by raised algorithm number. † indicates statistically significant differences to best result across features groups for same algorithm, indicated in **bold face**. @ indicates the optimal number of epochs chosen on the devtest set.

and tuning the full set of 180,000 features are not significant. However, scaling all features to the full training set shows significant improvements for algorithm 3, and especially for algorithm 4, which gains 0.8 BLEU points over tuning 12 features on the development set. The number of features rises to 4.7 million without feature selection, which iteratively selects 100,000 features with best ℓ_2 norm values across shards. Feature templates such as rule n-grams and rule shapes only work if iterative mixing (algorithm 3) or feature selection (algorithm 4) are used. Adding rule id features works in combination with other sparse features.

Table 3 shows results for algorithms 1 and 4 on the Europarl data (*ep*) for different devtest and test sets. Europarl data were used in all runs for training and for setting the meta-parameter of number of epochs. Testing was done on the Europarl test set and news crawl test data from the years 2010 and 2011. Here tuning large feature sets on the respective dev sets yields significant improvements of around 2 BLEU points over tuning the 12 default features on the dev sets. Another 0.5 BLEU points (test-*crawl*11) or even 1.3 BLEU points (test-*crawl*10) are gained when scaling to the full training set using iterative features selection. Result differences on the Europarl test set were not significant for moving from dev to full train set. Algorithms 2 and 3 were infeasible to run on Europarl data beyond one epoch because features vectors grew too large to be kept in memory.

6 Discussion

We presented an approach to scaling discriminative learning for SMT not only to large feature sets but also to large sets of parallel training data. Since inference for SMT (unlike many other learning problems) is very expensive, especially on large training sets, good parallelization is key. Our approach is made feasible and effective by applying joint feature selection across distributed stochastic learning processes. Furthermore, our local features are efficiently computable at runtime. Our algorithms and features are generic and can easily be reimplemented and make our results relevant across datasets and language pairs.

In future work, we would like to investigate more sophisticated features, better learners, and in general improve the components of our system that have been neglected in the current investigation of relative improvements by scaling the size of data and feature sets. Ultimately, since our algorithms are inspired by multi-task learning, we would like to apply them to scenarios where a natural definition of tasks is given. For example, patent data can be characterized along the dimensions of patent classes and patent text fields (Wäschle and Riezler, 2012) and thus are well suited for multi-task translation.

Acknowledgments

Stefan Riezler and Patrick Simianer were supported in part by DFG grant "Cross-language Learning-to-Rank for Patent Retrieval". Chris Dyer was supported in part by a MURI grant "The linguistic-core approach to structured translation and analysis of low-resource languages" from the US Army Research Office and a grant "Unsupervised Induction of Multi-Nonterminal Grammars for SMT" from Google, Inc.

References

Phil Blunsom, Trevor Cohn, and Miles Osborne. 2008. A discriminative latent variable models for statistical machine translation. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT'08)*, Columbus, OH.

Léon Bottou. 2004. Stochastic learning. In Olivier Bousquet, Ulrike von Luxburg, and Gunnar Rätsch, editors, *Advanced Lectures on Machine Learning*, pages 146–168. Springer, Berlin.

Olivier Chapelle and S. Sathiya Keerthi. 2010. Efficient algorithms for ranking with SVMs. *Information Retrieval Journal*.

David Chiang, Yuval Marton, and Philip Resnik. 2008. Online large-margin training of syntactic and structural translation features. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing (EMNLP'08)*, Waikiki, Honolulu, Hawaii.

David Chiang, Kevin Knight, and Wei Wang. 2009. 11,001 new features for statistical machine translation. In *Proceedings of the 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT'09)*, Boulder, CO.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, Ann Arbor, MI.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2).

Michael Collins. 2002. Discriminative training methods for hidden markov models: theory and experiments with perceptron algorithms. In *Proceedings of the conference on Empirical Methods in Natural Language Processing (EMNLP'02)*, Philadelphia, PA.

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.

Jeffrey Dean and Sanjay Ghemawat. 2004. Mapreduce: Simplified data processing on large clusters. In *Proceedings of the 6th Symposium on Operating System Design and Implementation (OSDI'04)*, San Francisco, CA.

Kevin Duh and Katrin Kirchhoff. 2008. Beyond log-linear models: Boosted minimum error rate training for n-best ranking. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL'08), Short Paper Track*, Columbus, OH.

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, Jonathan 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 ACL 2010 System Demonstrations*, Uppsala, Sweden.

Chris Dyer, Kevin Gimpel, Jonathan H. Clark, and Noah A. Smith. 2011. The CMU-ARK german-english translation system. In *Proceedings of the 6th Workshop on Machine Translation (WMT11)*, Edinburgh, UK.

Chris Dyer. 2009. Using a maximum entropy model to build segmentation lattices for MT. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics - Human Language Technologies (NAACL-HLT'09)*, Boulder, CO.

Yoav Freund and Robert E. Schapire. 1999. Large margin classification using the perceptron algorithm. *Journal of Machine Learning Research*, 37:277–296.

Kevin Gimpel and Noah A. Smith. 2012. Structured ramp loss minimization for machine translation. In *Proceedings of 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2012)*, Montreal, Canada.

Katsuhiko Hayashi, Taro Watanabe, Hajime Tsukada, and Hideki Isozaki. 2009. Structural support vector machines for log-linear approach in statistical machine translation. In *Proceedings of IWSLT*, Tokyo, Japan.

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

Mark Hopkins and Jonathan May. 2011. Tuning as ranking. In *Proceedings of 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP'11)*, Edinburgh, Scotland.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the Human Language Technology Conference and the 3rd Meeting of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL'03)*, Edmonton, Cananda.

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.

Thomas Navin Lal, Olivier Chapelle, Jason Weston, and André Elisseeff. 2006. Embedded methods. In I.M. Guyon, S.R. Gunn, M. Nikravesh, and L. Zadeh, editors, *Feature Extraction: Foundations and Applications*. Springer.

Percy Liang, Alexandre Bouchard-Côté, Dan Klein, and Ben Taskar. 2006a. An end-to-end discriminative approach to machine translation. In *Proceedings of the joint conference of the International Committee on Computational Linguistics and the Association for Computational Linguistics (COLING-ACL'06)*, Sydney, Australia.

Percy Liang, Ben Taskar, and Dan Klein. 2006b. Alignment by agreement. In *Proceedings of the Human Language Technology Conference - North American Chapter of the Association for Computational Linguistics annual meeting (HLT-NAACL'06)*, New York, NY.

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

David McAllester and Joseph Keshet. 2011. Generalization bounds and consistency for latent structural probit and ramp loss. In *Proceedings of the 25th Annual Conference on Neural Information Processing Sytems (NIPS 2011)*, Granada, Spain.

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.

Eric W. Noreen. 1989. *Computer Intensive Methods for Testing Hypotheses. An Introduction*. Wiley, New York.

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.

Franz Josef Och and Hermann Ney. 2000. Improved statistical alignment models. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL'00)*, Hongkong, China.

Franz Josef Och and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL'02)*, Philadelphia, PA.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the Human Language Technology Conference and the 3rd Meeting of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL'03)*, Edmonton, Cananda.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2001. Bleu: a method for automatic evaluation of machine translation. Technical Report IBM Research Division Technical Report, RC22176 (W0190-022), Yorktown Heights, N.Y.

Simon Perkins, Kevin Lacker, and James Theiler. 2003. Grafting: Fast, incremental feature selection by gradient descent in function space. *Journal of Machine Learning Research*, 3:1333–1356.

Stefan Riezler and John Maxwell. 2005. On some pitfalls in automatic evaluation and significance testing for MT. In *Proceedings of the ACL-05 Workshop on Intrinsic and Extrinsic Evaluation Measures for MT and/or Summarization*, Ann Arbor, MI.

Shai Shalev-Shwartz, Yoram Singer, and Nathan Srebro. 2007. Pegasos: Primal Estimated sub-GrAdient SOlver for SVM. In *Proceedings of the 24th International Conference on Machine Learning (ICML'07)*, Corvallis, OR.

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

Libin Shen, Anoop Sarkar, and Franz Josef Och. 2004. Discriminative reranking for machine translation. In *Proceedings of the Human Language Technology conference / North American chapter of the Association for Computational Linguistics annual meeting (HLT/NAACL'04)*, Boston, MA.

Andreas Stolcke. 2002. SRILM - an extensible language modeling toolkit. In *Proceedings of the International Conference on Spoken Language Processing*, Denver, CO.

Christoph Tillmann and Tong Zhang. 2006. A discriminatie global training algorithm for statistical MT. In *Proceedings of the joint conference of the International Committee on Computational Linguistics and the Association for Computational Linguistics (COLING-ACL'06)*, Sydney, Australia.

Katharina Wäschle and Stefan Riezler. 2012. 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.

Taro Watanabe, Jun Suzuki, Hajime Tsukada, and Hideki Isozaki. 2006. NTT statistical machine translation for IWSLT 2006. In *Proceedings of the International Workshop on Spoken Language Translation (IWSLT)*, Kyoto, Japan.

Taro Watanabe, Jun Suzuki, Hajime Tsukada, and Hideki Isozaki. 2007. Online large-margin training for statistical machine translation. In *Proceedings of the 2007*

Joint Conference on Empirical Mehtods in Natural Language Processing and Computational Language Learning (EMNLP'07), Prague, Czech Republic.

Benjamin Wellington, Joseph Turian, and Dan Melamed. 2009. Toward purely discriminative training for tree-structured translation models. In Cyril Goutte, Nicola Cancedda, and Marc Dymetman, editors, *Learning Machine Translation*, pages 132–149, Cambridge, MA. The MIT Press.

Martin A. Zinkevich, Markus Weimer, Alex Smola, and Lihong Li. 2010. Parallelized stochastic gradient descent. In *Proceedings of the 24th Annual Conference on Neural Information Processing Sytems (NIPS'10)*, Vancouver, Canada.

Andreas Zollmann and Khalil Sima'an. 2005. A consistent and efficient estimator for data-oriented parsing. *Journal of Automata, Languages and Combinatorics*, 10(2/3):367–388.