Towards Efficient Large-Scale Feature-Rich Statistical Machine Translation

Vladimir Eidelman[1], Ke Wu[1], Ferhan Ture[1], Philip Resnik[2], Jimmy Lin[3]
[1] Dept. of Computer Science [2] Dept. of Linguistics [3] The iSchool
Institute for Advanced Computer Studies
University of Maryland
{vlad,wuke,fture,resnik,jimmylin}@umiacs.umd.edu

Abstract

We present the system we developed to provide efficient large-scale feature-rich discriminative training for machine translation. We describe how we integrate with MapReduce using Hadoop streaming to allow arbitrarily scaling the tuning set and utilizing a sparse feature set. We report our findings on German-English and Russian-English translation, and discuss benefits, as well as obstacles, to tuning on larger development sets drawn from the parallel training data.

1 Introduction

The adoption of discriminative learning methods for SMT that scale easily to handle sparse and lexicalized features has been increasing in the last several years (Chiang, 2012; Hopkins and May, 2011). However, relatively few systems take full advantage of the opportunity. With some exceptions (Simianer et al., 2012), most still rely on tuning a handful of common dense features, along with at most a few thousand others, on a relatively small development set (Cherry and Foster, 2012; Chiang et al., 2009). While *more* features tuned on *more* data usually results in better performance for other NLP tasks, this has not necessarily been the case for SMT.

Thus, our main focus in this paper is to improve understanding into the effective use of sparse features, and understand the benefits and shortcomings of large-scale discriminative training. To this end, we conducted experiments for the shared translation task of the 2013 Workshop on Statistical Machine Translation for the German-English and Russian-English language pairs.

2 Baseline system

We use a hierarchical phrase-based decoder implemented in the open source translation system `cdec`[1] (Dyer et al., 2010). For tuning, we use Mr. MIRA[2] (Eidelman et al., 2013), an open source decoder agnostic implementation of online large-margin learning in Hadoop MapReduce. Mr. MIRA separates learning from the decoder, allowing the flexibility to specify the desired inference procedure through a simple text communication protocol. The decoder receives input sentences and weight updates from the learner, while the learner receives k-best output with feature vectors from the decoder.

Hadoop MapReduce (Dean and Ghemawat, 2004) is a popular distributed processing framework that has gained widespread adoption, with the advantage of providing scalable parallelization in a manageable framework, taking care of data distribution, synchronization, fault tolerance, as well as other features. Thus, while we could otherwise achieve the same level of parallelization, it would be in a more ad-hoc manner.

The advantage of online methods lies in their ability to deal with large training sets and high-dimensional input representations while remaining simple and offering fast convergence. With Hadoop streaming, our system can take advantage of commodity clusters to handle parallel large-scale training while also being capable of running on a single machine or PBS-managed batch cluster.

System design To efficiently encode the information that the learner and decoder require (source sentence, reference translation, grammar rules) in a manner amenable to MapReduce, i.e. avoiding dependencies on "side data" and large transfers across the network, we append the reference and

[1] `http://cdec-decoder.org`
[2] `https://github.com/kho/mr-mira`

per-sentence grammar to each input source sentence. Although this file's size is substantial, it is not a problem since after the initial transfer, it resides on Hadoop distributed file system, and MapReduce optimizes for data locality when scheduling mappers.

A single iteration of training is performed as a Hadoop streaming job. Each begins with a *map* phase, with every parallel mapper loading the same initial weights and decoding and updating parameters on a shard of the data. This is followed by a *reduce* phase, with a single reducer collecting final weights from all mappers and computing a weighted average to distribute as initial weights for the next iteration.

Parameter Settings We tune our system toward approximate sentence-level BLEU (Papineni et al., 2002),[3] and the decoder is configured to use cube pruning (Huang and Chiang, 2007) with a limit of 200 candidates at each node. For optimization, we use a learning rate of η=1, regularization strength of C=0.01, and a 500-best list for hope and fear selection (Chiang, 2012) with a single passive-aggressive update for each sentence (Eidelman, 2012).

Baseline Features We used a set of 16 standard baseline features: rule translation relative frequency $P(e|f)$, lexical translation probabilities $P_{lex}(\overline{e}|\overline{f})$ and $P_{lex}(\overline{f}|\overline{e})$, target n-gram language model $P(e)$, penalties for source and target words, passing an untranslated source word to the target side, singleton rule and source side, as well as counts for arity-0,1, or 2 SCFG rules, the total number of rules used, and the number of times the glue rule is used.

2.1 Data preparation

For both languages, we used the provided Europarl and News Commentary parallel training data to create the translation grammar necessary for our model. For Russian, we additionally used the Common Crawl and Yandex data. The data were lowercased and tokenized, then filtered for length and aligned using the GIZA++ implementation of IBM Model 4 (Och and Ney, 2003) to obtain one-to-many alignments in both directions and symmetrized sing the grow-diag-final-and method (Koehn et al., 2003).

[3]We approximate corpus BLEU by scoring sentences using a pseudo-document of previous 1-best translations (Chiang et al., 2009).

We constructed a 5-gram language model using SRILM (Stolcke, 2002) from the provided English monolingual training data and parallel data with modified Kneser-Ney smoothing (Chen and Goodman, 1996), which was binarized using KenLM (Heafield, 2011). The sentence-specific translation grammars were extracted using a suffix array rule extractor (Lopez, 2007).

For German, we used the 3,003 sentences in newstest2011 as our Dev set, and report results on the 3,003 sentences of the newstest2012 Test set using BLEU and TER (Snover et al., 2006). For Russian, we took the first 2,000 sentences of newstest2012 for Dev, and report results on the remaining 1,003. For both languages, we selected 1,000 sentences from the bitext to be used as an additional testing set (Test2).

Compound segmentation lattices As German is a morphologically rich language with productive compounding, we use word segmentation lattices as input for the German translation task. These lattices encode alternative segmentations of compound words, allowing the decoder to automatically choose which segmentation is best. We use a maximum entropy model with recommended settings to create lattices for the dev and test sets, as well as for obtaining the 1-best segmentation of the training data (Dyer, 2009).

3 Evaluation

This section describes the experiments we conducted in moving towards a better understanding of the benefits and challenges posed by large-scale high-dimensional discriminative tuning.

3.1 Sparse Features

The ability to incorporate sparse features is the primary reason for the recent move away from Minimum Error Rate Training (Och, 2003), as well as for performing large-scale discriminative training. We include the following sparse Boolean feature templates in our system in addition to the aforementioned baseline features: rule identity (for every unique rule in the grammar), rule shape (mapping rules to sequences of terminals and nonterminals), target bigrams, lexical insertions and deletions (for the top 150 unaligned words from the training data), context-dependent word pairs (for the top 300 word pairs in the training data), and structural distortion (Chiang et al., 2008).

	Dev	Test	Test2	5k	10k	25k	50k
en	75k	74k	27k	132k	255k	634k	1258k
de	74k	73k	26k	133k	256k	639k	1272k

Table 1: Corpus statistics in tokens for German.

	Dev	Test	Test2	15k
ru	46k	24k	24k	350k
en	50k	27k	25k	371k

Table 2: Corpus statistics in tokens for Russian.

Set	# features	Tune	Test	
		↑BLEU	↑BLEU	↓TER
de-en	16	22.38	22.69	60.61
+sparse	108k	23.86	**23.01**	**59.89**
ru-en	16	30.18	29.89	49.05
+sparse	77k	32.40	**30.81**	**48.40**

Table 3: Results with the addition of sparse features for German and Russian.

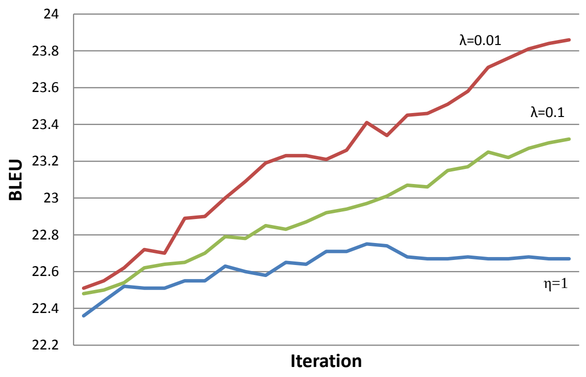

Figure 1: Learning curves for tuning when using a single step size (η) versus different per-feature learning rates.

All of these features are generated from the translation rules on the fly, and thus do not have to be stored as part of the grammar. To allow for memory efficiency while scaling the training data, we hash all the lexical features from their string representation into a 64-bit integer.

Altogether, these templates result in millions of potential features, thus how to select appropriate features, and how to properly learn their weights can have a large impact on the potential benefit.

3.2 Adaptive Learning Rate

The passive-aggressive update used in MIRA has a single learning rate η for all features, which along with α limits the amount each feature weight can change at each update. However, since the typical dense features (e.g., language model) are observed far more frequently than sparse features (e.g., rule identity), it has been shown to be advantageous to use an adaptive per-feature learning rate that allows larger steps for features that do not have much support (Green et al., 2013; Duchi et al., 2011). Essentially, instead of having a single parameter η,

$$\alpha \leftarrow \min\left(C, \frac{\text{cost}(y') - \mathbf{w}^\top(\mathbf{f}(y^+) - \mathbf{f}(y'))}{\|\mathbf{f}(y^+) - \mathbf{f}(y')\|^2}\right)$$
$$\mathbf{w} \leftarrow \mathbf{w} + \alpha\eta\left(\mathbf{f}(y^+) - \mathbf{f}(y')\right)$$

we instead have a vector Σ with one entry for each feature weight:

$$\Sigma^{-1} \leftarrow \Sigma^{-1} + \lambda \text{diag}\left(\mathbf{w}\mathbf{w}^\top\right)$$
$$\mathbf{w} \leftarrow \mathbf{w} + \alpha\Sigma^{1/2}\left(\mathbf{f}(y^+) - \mathbf{f}(y')\right)$$

In practice, this update is very similar to that of AROW (Crammer et al., 2009; Chiang, 2012).

Figure 1 shows learning curves for sparse models with a single learning rate, and adaptive learning with λ=0.01 and λ=0.1, with associated results on Test in Table 4.[4] As can be seen, using a single η produces almost no gain on Dev. However, while both settings using an adaptive rate fare better, the proper setting of λ is important. With λ=0.01 we observe 0.5 BLEU gain over λ=0.1 in tuning, which translates to a small gain on Test. Henceforth, we use an adaptive learning rate with λ=0.01 for all experiments.

Table 3 presents baseline results for both languages. With the addition of sparse features, tuning scores increase by 1.5 BLEU for German, leading to a 0.3 BLEU increase on Test, and 2.2 BLEU for Russian, with 1 BLEU increase on Test. The majority of active features for both languages are rule id (74%), followed by target bigrams (14%) and context-dependent word pairs (11%).

3.3 Feature Selection

As the tuning set size increases, so do the number of active features. This may cause practical problems, such as reduced speed of computation and memory issues. Furthermore, while some

[4]All sparse models are initialized with the same tuned baseline weights. Learning rates are local to each mapper.

Adaptive	# feat.	Tune	Test	
		↑BLEU	↑BLEU	↓TER
none	74k	22.75	22.87	60.19
λ=0.01	108k	23.86	**23.01**	**59.89**
λ=0.1	62k	23.32	22.92	60.09

Table 4: Results with different λ settings for using a per-feature learning rate with sparse features.

Set	# feat.	Tune	Test	
		↑BLEU	↑BLEU	↓TER
all	510k	32.99	22.36	59.26
top 200k	200k	32.96	22.35	59.29
all	373k	34.26	28.84	49.29
top 200k	200k	34.45	**28.98**	49.30

Table 5: Comparison of using all features versus top k selection.

sparse features will generalize well, others may not, thereby incurring practical costs with no performance benefit. Simianer et al. (2012) recently explored ℓ_1/ℓ_2 regularization for joint feature selection for SMT in order to improve efficiency and counter overfitting effects. When performing parallel learning, this allows for selecting a reduced set of the top k features at each iteration that are effective across all learners.

Table 5 compares selecting the top 200k features versus no selection for a larger German and Russian tuning set (§3.4). As can be seen, we achieve the same performance with the top 200k features as we do when using double that amount, while the latter becomes increasing cumbersome to manage. Therefore, we use a top 200k selection for the remainder of this work.

3.4 Large-Scale Training

In the previous section, we saw that learning sparse features on the small development set leads to substantial gains in performance. Next, we wanted to evaluate if we can obtain further gains by scaling the tuning data to learn parameters directly on a portion of the training bitext. Since the bitext is used to learn rules for translation, using the same parallel sentences for grammar extraction as well as for tuning feature weights can lead to severe overfitting (Flanigan et al., 2013). To avoid this issue, we used a jackknifing method to split the training data into $n = 10$ folds, and built a translation system on $n-1$ folds, while sampling sentences from the News Commentary portion of the held-out fold to obtain tuning sets from 5,000 to 50,000 sentences for German, and 15,000 sentences for Russian.

Results for large-scale training for German are presented in Table 6. Although we cannot compare the tuning scores across different size sets, we can see that tuning scores for all sets improve substantially with sparse features. Unfortunately, with increasing tuning set size, we see very little improvement in Test BLEU and TER with either feature set. Similar findings for Russian are presented in Table 7. Introducing sparse features improves performance on each set, respectively, but Dev always performs better on Test.

While tuning on Dev data results in better BLEU on Test than when tuning on the larger sets, it is important to note that although we are able to tune more features on the larger bitext tuning sets, they are not composed of the same genre as the Tune and Test sets, resulting in a domain mismatch.

This phenomenon is further evident in German when testing each model on Test2, which is selected from the bitext, and is thus closer matched to the larger tuning sets, but is separate from both the parallel data used to build the translation model and the tuning sets. Results on Test2 clearly show significant improvement using any of the larger tuning sets versus Dev for both the baseline and sparse features. The 50k sparse setting achieves almost 1 BLEU and 2 TER improvement, showing that there are significant differences between the Dev/Test sets and sets drawn from the bitext.

For Russian, we amplified the effects by selecting Test2 from the portion of the bitext that is separate from the tuning set, but is among the sentences used to create the translation model. The effects of overfitting are markedly more visible here, as there is almost a 7 BLEU difference between tuning on Dev and the 15k set with sparse features. Furthermore, it is interesting to note when looking at Dev that using sparse features has a significant negative impact, as the baseline tuned Dev performs

Tuning	Test	
	↑BLEU	↓TER
5k	22.81	59.90
10k	22.77	59.78
25k	22.88	59.77
50k	22.86	59.76

Table 8: Results for German with 2 iterations of tuning on Dev after tuning on larger set.

reasonably well, while the introduction of sparse features leads to overfitting the specificities of the Dev/Test genre, which are not present in the bitext.

We attempted two strategies to mitigate this problem: combining the Dev set with the larger bitext tuning set from the beginning, and tuning on a larger set to completion, and then running 2 additional iterations of tuning on the Dev set using the learned model. Results for tuning on Dev and a larger set together are presented in Table 7 for Russian and Table 6 for German. As can be seen, the resulting model improves somewhat on the other genre and strikes a middle ground, although it is worse on Test than Dev.

Table 8 presents results for tuning several additional iterations after learning a model on the larger sets. Although this leads to gains of around 0.5 BLEU on Test, none of the models outperform simply tuning on Dev. Thus, neither of these two strategies seem to help. In future work, we plan to forgo randomly sampling the tuning set from the bitext, and instead actively select the tuning set based on similarity to the test set.

4 Conclusion

We explored strategies for scaling learning for SMT to large tuning sets with sparse features. While incorporating an adaptive per-feature learning rate and feature selection, we were able to use Hadoop to efficiently take advantage of large amounts of data. Although discriminative training on larger sets still remains problematic, having the capability to do so remains highly desirable, and we plan to continue exploring methods by which to leverage the power of the bitext effectively.

Acknowledgments

This research was supported in part by the DARPA BOLT program, Contract No. HR0011-12-C-0015; NSF under awards IIS-0916043 and IIS-1144034. Vladimir Eidelman is supported by a NDSEG Fellowship.

References

S. Chen and J. Goodman. 1996. An empirical study of smoothing techniques for language modeling. In *ACL*.

Colin Cherry and George Foster. 2012. Batch tuning strategies for statistical machine translation. In *NAACL*.

David Chiang, Yuval Marton, and Philip Resnik. 2008. Online large-margin training of syntactic and structural translation features. In *EMNLP*.

D. Chiang, K. Knight, and W. Wang. 2009. 11,001 new features for statistical machine translation. In *NAACL-HLT*.

D. Chiang. 2012. Hope and fear for discriminative training of statistical translation models. *JMLR*, 13:1159–1187.

K. Crammer, A. Kulesza, and M. Dredze. 2009. Adaptive regularization of weight vectors. In *NIPS*.

J. Dean and S. Ghemawat. 2004. MapReduce: Simplified data processing on large clusters. In *OSDI*.

J. Duchi, E. Hazan, and Y. Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*, 12:2121–2159.

C. Dyer, A. Lopez, J. Ganitkevitch, J. Weese, F. Ture, P. Blunsom, H. Setiawan, V. Eidelman, and P. Resnik. 2010. cdec: A decoder, alignment, and learning framework for finite-state and context-free translation models. In *ACL System Demonstrations*.

Chris Dyer. 2009. Using a maximum entropy model to build segmentation lattices for mt. In *Proceedings of NAACL-HLT*.

Vladimir Eidelman, Ke Wu, Ferhan Ture, Philip Resnik, and Jimmy Lin. 2013. Mr. MIRA: Open-source large-margin structured learning on map-reduce. In *ACL System Demonstrations*.

Vladimir Eidelman. 2012. Optimization strategies for online large-margin learning in machine translation. In *WMT*.

Jeffrey Flanigan, Chris Dyer, and Jaime Carbonell. 2013. Large-scale discriminative training for statistical machine translation using held-out line search. In *NAACL*.

S. Green, S. Wang, D. Cer, and C. Manning. 2013. Fast and adaptive online training of feature-rich translation models. In *ACL*.

Kenneth Heafield. 2011. KenLM: faster and smaller language model queries. In *WMT*.

Mark Hopkins and Jonathan May. 2011. Tuning as ranking. In *EMNLP*.

Tuning	# mappers	# features	Tune	Test		Test2	
			↑BLEU	↑BLEU	↓TER	↑BLEU	↓TER
Dev	120	16	22.38	**22.69**	60.61	29.31	54.26
5k	120	16	32.60	22.14	59.60	29.69	52.96
10k	120	16	33.16	22.06	59.43	29.93	52.37
Dev+10k	120	16	19.40	22.32	59.37	30.17	52.45
25k	300	16	32.48	22.21	59.54	30.03	51.71
50k	600	16	32.21	22.21	**59.39**	29.94	52.55
Dev	120	108k	23.86	**23.01**	59.89	29.65	53.86
5k	120	159k	33.70	22.26	59.26	30.53	51.84
10k	120	200k	34.00	22.12	59.24	30.51	51.71
Dev+10k	120	200k	19.62	22.42	59.17	30.26	52.21
25k	300	200k	32.96	22.35	59.29	30.39	52.14
50k	600	200k	32.86	22.40	**59.15**	30.54	51.88

Table 6: German evaluation with large-scale tuning, showing numbers of mappers employed, number of active features for best model, and test scores on Test and bitext Test2 domains.

Tuning	# mappers	# features	Tune	Test		Test2	
			↑BLEU	↑BLEU	↓TER	↑BLEU	↓TER
Dev	120	16	30.18	29.89	49.05	57.14	32.56
15k	200	16	34.65	28.60	49.63	59.64	30.65
Dev+15k	200	16	33.97	28.88	49.37	58.24	31.81
Dev	120	77k	32.40	**30.81**	**48.40**	52.90	36.85
15k	200	200k	35.05	28.34	49.69	59.81	30.59
Dev+15k	200	200k	34.45	28.98	49.30	57.61	32.71

Table 7: Russian evaluation with large-scale tuning, showing numbers of mappers employed, number of active features for best model, and test scores on Test and bitext Test2 domains.

Liang Huang and David Chiang. 2007. Forest rescoring: Faster decoding with integrated language models. In *ACL*.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *NAACL*.

Adam Lopez. 2007. Hierarchical phrase-based translation with suffix arrays. In *EMNLP*.

Franz Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. In *Computational Linguistics*.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *ACL*.

K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *ACL*.

P. Simianer, S. Riezler, and C. Dyer. 2012. Joint feature selection in distributed stochastic learning for large-scale discriminative training in SMT. In *ACL*.

M. Snover, B. Dorr, R. Schwartz, L. Micciulla, and J. Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *AMTA*.

Andreas Stolcke. 2002. SRILM - an extensible language modeling toolkit. In *ICSLP*.