Tuning SMT with A Large Number of Features via Online Feature Grouping

Lemao Liu[1], **Tiejun Zhao**[1], **Taro Watanabe**[2], **Eiichiro Sumita**[2]
[1]School of Computer Science and Technology
Harbin Institute of Technology, Harbin, China
[2]National Institute of Information and Communications Technology
3-5 Hikari-dai, Seika-cho, Soraku-gun, Kyoto, Japan
{lmliu|tjzhao}@mtlab.hit.edu.cn
{taro.watanabe|eiichiro.sumita}@nict.go.jp

Abstract

In this paper, we consider the tuning of statistical machine translation (SMT) models employing a large number of features. We argue that existing tuning methods for these models suffer serious sparsity problems, in which features appearing in the tuning data may not appear in the testing data and thus those features may be over tuned in the tuning data. As a result, we face an over-fitting problem, which limits the generalization abilities of the learned models. Based on our analysis, we propose a novel method based on feature grouping via OSCAR to overcome these pitfalls. Our feature grouping is implemented within an online learning framework and thus it is efficient for a large scale (both for features and examples) of learning in our scenario. Experiment results on IWSLT translation tasks show that the proposed method significantly outperforms the state of the art tuning methods.

1 Introduction

Since the introduction of log-linear based SMT (Och and Ney, 2002), tuning has been a hot topic. Various methods have been explored: their objectives are either error rates (Och, 2003), hinge loss (Watanabe et al., 2007; Chiang et al., 2008) or ranking loss (Hopkins and May, 2011), and they are either batch training or online training methods. In this paper, we consider tuning translation models with a large number of features such as lexical, n-gram level and rule level features, where the number of features is largely greater than the number of bilingual sentences. Practically, existing tuning methods such as PRO and MIRA might be applied in our scenario, however, they will suffer from some pitfalls as well, which have been less investigated in previous works.

One of pitfalls is that these features are so sparse that many features which are potentially useful for a test set may not be included in a given tuning set, and many useless features for testing will be over tuned on the developement set meanwhile. As a result, the generalization abilities of features are limited due to the mismatch between the testing data and the tuning data, and over-fitting occurs. One practice is to tune translation models on a larger tuning set, such as the entire training data (Xiao et al., 2011; Simianer et al., 2012), in the hope that more features would be included during tuning. However, tuning robust weights for translation models has additional requirements to a tuning set. Firstly, multiple reference translations in the tuning data are helpful for better tuning, especially when testing data contains multiple reference translations. Secondly, the closeness between the tuning set and a test set is also important for better testing performance (Li et al., 2010). These requirements can explain why tuning on the training data leads to unsatisfactory performance on the IWSLT translation task, as will be shown in our experiments later. Therefore, enlarging a tuning set is not always a sufficient solution for robust tuning, since it would be impractical to create a large scale tuning set with these requirements.

We propose a novel tuning method by *grouping* a large number of features to leverage the above pitfalls. Instead of directly taking the large number of atomic features into translation model, we firstly learn their group structure on the training data to alleviate their serious sparsity. Then, we tune the translation model consisting of grouped features on a multi-reference development set to ensure robust tuning. Unlike unsupervised clustering methods such as k-means (MacQueen, 1967) for feature clustering, we group the features with

This joint work was done while the first author visited NICT.

the OSCAR (Octagonal Shrinkage and Clustering Algorithm for Regression) method (Bondell and Reich, 2008), which directly relates the objective of feature grouping to translation evaluation metrics such as BLEU (Papineni et al., 2002) and thus grouped features are optimized with respect to BLEU. Due to the large number of features and large number of training examples, efficient grouping is not simple. We apply the online gradient projection method under the FOBOS (forward-backward splitting) framework (Duchi and Singer, 2009) to accelerate feature grouping.

We employ a large number of features by treating each translation rule in a synchronous-CFG as a single feature. Experiments on IWSLT Chinese-to-English translation tasks show that, with the help of grouping these features, our method can overcome the above pitfalls and thus achieves significant improvements.

2 Tuning Method

We propose a novel tuning method for translation models with a large number of features, which incorporates feature grouping. Our assumption is that although a feature which is useful for a test set does not appear in the tuning set, another similar feature may exist. Therefore, grouping similar features can alleviate sparsity in this way. The proposed tuning method consists of two steps: first, it tries to learn a group structure for atomic features; second, it treats each feature group as a single feature and tunes the translation model on a given tuning set using off-the-shelf toolkits such as PRO. In the first step, we learn a group structure of atomic features in the large training data for better coverage. In the second step, we tune a translation model with the grouped features on a given development set with multiple references to ensure the robust tuning.

Before describing our tuning algorithm, we present notations for the rest of this paper. Suppose $\mathscr{H}$ is a feature set consisting of atomic features $\{h_1, h_2, \cdots, h_d\}$ or their index set $\{1, 2, \cdots, d\}$ for simplicity; $H = \langle h_1, h_2, \cdots, h_d \rangle$ is a d-dimensional feature vector function with respect to $\mathscr{H}$, and W is its weight vector with each component W_i and dimension d; $\mathscr{G} = \{g_1, g_2, ..g_M\}$ is a group of $\mathscr{H}$, where each element g_i is a power set of $\mathscr{H}$. Similarly, $G = \langle g_1, g_2, \cdots, g_M \rangle$ is an M-dimensional feature vector function with respect to $\mathscr{G}$ and $W^{\mathscr{G}}$ is its weight with with each component $W_i^{\mathscr{G}}$. In this paper, we consider the disjoint $\mathscr{G}$, i.e. $g_i \cap g_j = \emptyset$ if $i \neq j$. Further, suppose $\Delta(W)$ is a set of the index i such that $W_i \neq 0$, and $|\cdot|$ is either the number of elements in a set S or the absolute value of a real number x.

Algorithm 1 Tuning Algorithm

```
Input: training data, dev, $W^{ini}$, $T$
1: Initialize $W^1 = W^{ini}$
2: for all $i$ such that $1 \leq i \leq T$ do
3:     Decode on training data with $W^i$ to obtain a k-best-list and merge k-best-lists
4:     Update the group set $\mathscr{G}$ based on the merged k-best-list ▷ Call Algorithm 2
5:     Tune the translation model with $\mathscr{G}$ as the feature set on dev with PRO to update $W^{\mathscr{G}}$
6:     Unpack $W^{\mathscr{G}}$ to $W^{i+1}$
7: end for
8: $W = W^{T+1}$
Output: $W$
```

Algorithm 1 describes our two-step tuning procedure for a translation model with $\mathscr{H}$ as its feature set. It inputs a training data set, a development set, initial weight W^1 with respect to $\mathscr{H}$, and maximal iterations T; and outputs a weight W. It initializes with W^1 in line 1; from line 2 to line 7, it iteratively obtains a k-best-list by decoding with W^i, updates the group set $\mathscr{G}$, tunes the translation weight $W^{\mathscr{G}}$ based on $\mathscr{G}$, and unpacks the $W^{\mathscr{G}}$ to obtain W^{i+1}. At the end, it returns the final weight W. In particular, the k-best-list is obtained using H as a feature vector with its weight vector W derived from grouped weights $W^{\mathscr{G}}$ through unpacking: if $h_j \in g_k$, then W_j=$W_k^{\mathscr{G}}$. The grouping algorithm in line 3 will be introduced in the next section.

In this paper, we use a hierarchical phrase based translation model, which consists of 8 default features: translation probabilities, lexical translation probabilities, word penalty, glue rule penalty, synchronous rule penalty and language model. In addition, we also employ a large number rule identify (id) features: each rule itself is a feature, and if a translation contains a rule for x times, then the value of this rule id feature is x. In line 4 we group these id features and impose that each default feature itself is a group.

3 Online Feature grouping

Algorithm 2 Feature Grouping Algorithm

```
Input: $\lambda_1$,$\lambda_2$,k-best-list, $W^1$, $n$
1: Collect a set of tuples $\{\langle f, e', e^* \rangle\}$ from k-best-list
2: for all $i$ such that $1 \leq t \leq n$ do
3:     Randomly select $\langle f, e', e^* \rangle$ from the tuple set
4:     $W^{t+1/2} = W^t + \nabla_W \delta(f, e', e^*, W^t)/t$
5:     Minimize $Q(W; 2W^{t+1/2}, t+1, \lambda_1, \lambda_2)$ to obtain $(W^{t+1}, \mathscr{G})$
                                              ▷ Group optimization
6: end for
Output: $\mathscr{G}$
```

Suppose f is a sentence in a development set, C is a set of translations for f, and r is a set of reference translations for f. Following PRO, we define ranking loss function as follows:

$$L(W) = \frac{1}{N} \sum_f \sum_{e^*, e'} \delta(f, e', e^*, W), \qquad (1)$$

with

$$\delta(f, e', e^*, W) = \max_W \left\{ \left(H(f, e') - H(f, e^*) \right) \cdot W + 1, 0 \right\},$$

where $e', e^* \in C$ such that BLEU$(e^*, r) >$ BLEU(e', r), and N is the number of all tuples $\langle e^*, e', f \rangle$.

To achieve group structure and avoid the sparsity in H, we apply the OSCAR over the above loss function, and obtain the function:

$$L(W) + \lambda_1 \sum_{i=1}^{d} |W_i| + \lambda_2 \sum_{1 \leq i < j \leq d} \max\{|W_i|, |W_j|\}, \qquad (2)$$

where d is the dimension of feature vector H or its weight W, λ_1 and λ_2 are two hyperparameters for two regularizers taking positive value. Minimization of Eq.2 makes some components in W equal and thus achieves a feature grouping effect. In other words, $W_i = W_j$ means that h_i and h_j lie in the same group, i.e. $h_i, h_j \in g_k$ for some $g_k \in \mathscr{D}(W)$, where $\mathscr{D}(W)$ denotes the group derived from W as follows. Given W, we first sort its components W_i to obtain a permutation $\{i_k\}_{k=1}^d$ such that $W_{i_1} \leq W_{i_2} \cdots \leq W_{i_d}$ with $1 \leq i_k \leq d$; then we can easily obtain $\mathscr{D}(W)$ after traversing $\{W_{i_k}\}_{k=1}^d$. For example, $W = \langle 1, 3, 1, 3, 1 \rangle$, then $\mathscr{D}(W) = \{\{1, 3, 5\}, \{2, 4\}\}$. One advantage of OSCAR over unsupervised clustering methods (e.g. k-means) is that it relates the objective of grouping to an error metric, such as BLEU, and thus can achieve an optimal grouping towards BLEU.

Bondell and Reich (2008) firstly proposed two approaches for OSCAR. The first one casts the problem into a quadratic program (QP) consisting of $O(d^2)$ variables and $O(d^2)$ constraints. The second one tries to optimize a sequence of (potentially smaller) QP's with more constraints, which can be up to $O(d!)$ in the worst case. Zhong and Kwok (2011) explored a much faster approach which is based on the accelerated gradient and projection method. Its complexity is reduced to $O(d \log d)$. Since the dimension d of H is large enough in our scenario where d is up to hundred of thousands, these existing optimization methods are inefficient to minimize Eq.2. Here, based on (Zhong and Kwok, 2011), we employ an online gradient projection algorithm under the FOBOS framework for faster learning. The framework of FOBOS is a type of online learning, in which it is theoretically guaranteed to solve such a problem as in Equation 2: the objectives consisting of two additive terms, in which one is non-smooth but convex and the other is smooth and convex[1]. FOBOS contains two steps: it first performs a gradient descent operator, and then updates weight by a proximity (or projection) operator.

Algorithm 2 describes the online training of feature grouping. It requires some inputs: two regularizer parameters λ_1 and λ_2; a k-best-list translations; an initial weight W^1; and a maximum iterations n. It firstly collects a set of tuples encoded with translation pairs from k-best-list following the strategy implemented in the PRO toolkit in line 1. It repeatedly updates weight W^t and feature group $\mathscr{G}$ from line 2 to line 6: it randomly samples a tuple $\langle f, e', e^* \rangle$ from the collected tuple set in line 3, it performs a gradient descent operator in line 4 where $\nabla_W \delta(f, e', e^*, W^t)$ denotes the subgradient of $\delta(f, e', e^*, W^t)$ at current weight W^t, and it optimizes $(W^{t+1}, \mathscr{G})$ by a proximity operator for group optimization in line 5. At last it returns the group result $\mathscr{G}$.

In particular, the subgradient of $\delta(f, e', e^*, W^t)$

[1]For Eq.2, the non-smooth but convex term is the entire of Eq.2, and the smooth and convex term can be considered as 0.

in line 4 is defined via the following equation:

$$\nabla_W \delta(f, e', e^*, W) = \begin{cases} H(f,e') - H(f,e^*), & \text{if } \delta(f,e',e^*,W) > 0; \\ 0, & \text{else.} \end{cases}$$

The main technique is the proximity operator for group optimization in line 5, which tries to minimize the function $Q(\cdot; a, t, \lambda_1, \lambda_2)$ with $a = 2 \times \big(W^t + \nabla_W \delta(f, e', e^*, W^t)/t\big)$:

$$Q(W; a, t, \lambda_1, \lambda_2) = (W-a)^\top W + \frac{2}{t}\big(\lambda_1 \sum_{i=1}^{d} |W_i| + \lambda_2 \sum_{1 \le i < j \le d} \max\{|W_i|, |W_j|\}\big). \quad (3)$$

In the next Section, we will present the details of this proximity for group optimization.

4 Group Optimization

To derive an efficient algorithm with large d for group optimization, we present the following lemma with its proof attached in appendix.

Lemma 1. *In Eq.3, if a_k=0, then its minimal solution $\hat{W}$ suffices to $\hat{W}_k = 0$.*

Suppose W^t is sparse, i.e. d is largely greater than the number of its non-zero components ($|\Delta(W^t)|$), then $W^{t+1/2}$ in line 4 is also sparse since H is sparse. The above Lemma states that the optimal solution W^{t+1} in line 5 of Algorithm 2 is also a sparse vector. Therefore, it is desirable to optimize the W^{t+1} in a low complexity independent on d. If so, we can easily see that if we set W^1 as a sparse vector, W^t is sparse for all $t > 1$ by a mathematical induction. Based on these analysis, the efficiency of proximity operator for group optimization only requires an assumption that its proximity step can be efficiently solved in a low complexity independent on a large d.

Let $u = |\Delta(a)|$, and p be a one-to-one map[2] $p : \{1, \cdots, u\} \to \{1, \cdots, d\}$, s.t. $|a_{p(1)}| \ge |a_{p(2)}| \ge \cdots \ge |a_{p(u)}| > 0$. Followed by Lemma 1, minimizing Eq.2 is equivalent to minimizing the following equation if we ignore the zero components in the optimal solutions of both equations:

$$\bar{Q}(W; a, t, \lambda_1, \lambda_2) = \sum_{i'=1}^{u} W_{p(i')}^2 - \sum_{i'=1}^{u} a_{p(i')} W_{p(i')} + \sum_{i'=1}^{u} \frac{2\big(\lambda_1 + \lambda_2(d-u)\big)}{t} |W_{p(i')}| + \frac{2\lambda_2}{t} \sum_{1 \le i' < j' \le u} \max\{|W_{p(i')}|, |W_{p(j')}|\}.$$

The advantage of optimizing $\bar{Q}(W; a, t, \lambda_1, \lambda_2)$ instead of $Q(W; a, t, \lambda_1, \lambda_2)$ is that it explicitly reduces the size of active components in W into u rather than d, and thus it is more direct to expect a faster optimization algorithm. Further, Proposition 1 in (Zhong and Kwok, 2011) states that the minimal solution $\hat{W}$ of such an equation as $\bar{Q}(W; a, t, \lambda_1, \lambda_2)$ suffices to the constraint $|\hat{W}_{p(1)}| \ge \cdots \ge |\hat{W}_{p(u)}|$. Therefore, minimizing $Q(W; a, t, \lambda_1, \lambda_2)$ is also equivalent to optimizing the following constraint programming:

$$\begin{aligned} \underset{W}{\text{minimize}} \quad & \bar{Q}(W; a, t, \lambda_1, \lambda_2) \\ \text{subject to} \quad & |W_{p(1)}| \ge \cdots \ge |W_{p(u)}|, \end{aligned}$$

where $\bar{Q}(W; a, t, \lambda_1, \lambda_2)$ defined on the constraint is rewritten as

$$\bar{Q}(W; a, t, \lambda_1, \lambda_2) = \sum_{i'=1}^{u} W_{p(i')}^2 - \sum_{i'=1}^{u} a_{p(i')} W_{p(i')} + \sum_{i'=1}^{u} \frac{2\big(\lambda_1 + \lambda_2(d-i')\big)}{t} |W_{p(i')}|. \quad (4)$$

Now, we can implement line 5 in Algorithm 2 as summarized by Algorithm 3, after some modifications over the projection algorithm in (Zhong and Kwok, 2011). Algorithm 3 requires some variables λ_1, λ_2, a and t. Firstly, it sorts $|a_i|$ for the indice in $\Delta(a)$ to obtain the map p in line 1, and initializes $\mathscr{G}$ as $\big\{\{p(1)\}\big\}$ in line 2. From line 3 to line 10, it goes into a merging loop where it repeatedly merges two group members to pre-calculate $\mathscr{G}$: for each i', it iteratively merges the member g initialized as $\{p(i')\}$ and the top member in the stack, updates g with the merged member, and substitutes the top member in the stack with g, if the v value (will be defined later) of g is greater than that of the top member. Then, it begins to calculate W initialized as 0 and $\mathscr{G}$. For each index i in each member g of $\mathscr{G}$, it assigns W_i

[2]For easier understanding, $i'(j')$ denotes the index in $\{1, \cdots, u\}$, while $i(j)$ denotes the index in $\{1, \cdots, d\}$.

Algorithm 3 Group Optimization

```
Input: $\lambda_1$,$\lambda_2$,$a$,$t$
 1: Sort $\{|a_i| : i \in \Delta(a)\}$ to obtain $p$ ▷ See the definition of $\Delta$ in Section 2
 2: Initialize stack of group set $\mathscr{G} = \{\{p(1)\}\}$
 3: for all $i'$ such that $2 \leq i' \leq |\Delta(a)|$ do
 4:     $g = \{p(i')\}$
 5:     while $\mathscr{G} \neq \emptyset$ and $v(g) \geq v(top(\mathscr{G}))$ do
 6:         $g = g \cup top(\mathscr{G})$ ▷ Merge $g$
 7:         Pop $top(\mathscr{G})$
 8:     end while
 9:     Push $g$ onto $\mathscr{G}$
10: end for                          ▷ Pre-calculate $\mathscr{G}$
11: $W = 0$
12: for all $g \in \mathscr{G}$ do
13:     for all $i \in g$ do
14:         $W_i = sign(a_i)max(v(g), 0)$
15:     end for
16: end for                          ▷ Calculate $W$
17: $\mathscr{G}$=$\mathscr{D}(W)$   ▷ Calculate $\mathscr{G}$
Output: $W$,$\mathscr{G}$            ▷ $W$ minimizes Eq.3
```

according to the sign[3] of a_i and $v(g)$ in line 14. In line 17 it calculates $\mathscr{G} = \mathscr{D}(W)$ as discussed in Section 3. At last it returns the pair $\langle W, \mathscr{G} \rangle$.

In particular, the v value $v(g)$ in line 5 is defined as

$$v(g) = \frac{\sum_{i \in g}\Big(|a_i| - 2\big(\lambda_1 + \lambda_2(d - p^*(i))\big)/t\Big)}{2\,|g|},$$

where $p^*(i)$ denotes the inversion of p such that $p\big(p^*(i)\big) = i$. And $v(g)$ can be intuitively interpreted as the group averaged sub-gradient of $\big(\sum_{i'=1}^{u} W_{p(i')}^2 - \bar{Q}(W; a, t, \lambda_1, \lambda_2)\big)/2$. In addition, an intuitive explanation of merging loop is that the value of objective in Eq.4 will be decreased after each merging step in line 6.

In summary, if we use a sparse representation for vector a in Algorithm 3, then its complexity is $O\big(|\Delta(a)|\log(|\Delta(a)|)\big)$, which is independent of d. Therefore, the whole tuning algorithm (Algorithm 1) with feature grouping is efficient even with a large value of d.

5 Experiments

We conduct experiments on the IWSLT2008 Chinese-to-English translation tasks, whose training data consists of about 30K bilingual sentence pairs. Test sets 2003, 2004 and 2008 are used as the development set, development test (devtest) set and test set, respectively; and all of them contain 16 references. A 5-gram language model is trained on the training data with the SRILM toolkit, and word alignment is obtained with GIZA++. In our experiments, the translation performances are measured by the case-insensitive BLEU4 metric. The significance testing is performed by paired bootstrap re-sampling (Koehn, 2004).

We use an in-house developed hierarchical phrase-based translation (Chiang, 2005) as our baseline decoder, and we use the state of the art tuning methods MERT and PRO as our comparison methods[4]. Based on our in-house decoder, we implement three translation models with different feature sets: default features (default); default features plus rule id features (+id) ; and default features plus group features of rule id (+group). On the IWSLT training data, the number of rule id features is 500K, i.e. $d = 500$K, which is significantly greater than the number of bilingual sentences 30K. Our proposed tuning method is with the following setting by tuning on the dev-test set: $\lambda_1 = 1e - 10$, $\lambda_2 = 3e - 8$, and $T = 15$, $n = 20 \times N$, i.e. 20 passes over k-best-lists.

From Table 1, we can see that tuning the translation model on the development set is much better (improvements of 4.3 BLEU scores) than that on the training data under the default features setting. Its main reason, as presented in Section 1, may be that multiple references and closeness[5] of tuning sets are much helpful for translation tasks. Further, the id features do not achieve improvements and even decreases 0.9 BLEU scores when tuned on the development set, due to its serious sparsity. However, after grouping id features, the groups learned by our method can alleviate the feature sparsity and thus significantly obtain gains of 0.7 BLEU scores over default feature setting.

Further, we implement another tuning method[6] for comparison, i.e. L$_1$ regularization method (Tsuruoka et al., 2009) based on the ranking loss $L(W)$ defined in Eq.1. We tune the translation

[3]The reason is attributed to the Eq.5 in (Zhong and Kwok, 2011).

[4]Both of them are derived from the Moses toolkit: http://www.statmt.org/moses/.

[5]If the tuning set and test set are close enough or identically distributed, it is possible to get gains by sparse discriminative features without using feature grouping(Chiang et al., 2009).

[6]It is similar to dtrain implemented in the cdec toolkit: http://cdec-decoder.org/, except that it does not use the distributed learning framework.

Methods	Tuning set	Feature set	# Features		BLEU4		Runtimes
			Active	Reused	devtest	test	
MERT	dev	default	8	8	45.7	40.6	15
PRO	dev	default	8	8	46.3	41.1	34
PRO	train	default	8	8	42.8	36.8	834
PRO	dev	+id	11081	4534	45.5	40.2	47
L_1	train	+id	584	71	42.7	36.9	975
L_1	dev	+id	443	248	46.2	41.0	39
OSCAR	–	**+group**	**503**	**425**	**46.9**	**41.8**	1256

Table 1: BLEU scores on the test set and tuning runtimes (minutes) for the different tuning methods with different settings. Tuning sets dev and train denote the development and training data sets, respectively. ”Active” denotes the number of active features for all methods except OSCAR or active grouped features for OSCAR; and ”Reused” denotes the number of active (or grouped) features which also appear during 1000-best decoding on the test set. Boldface BLEU means our method OSCAR is significantly better than other methods with $p < 0.05$.

model with the +id feature setting on both the development set and training data set, respectively, and their hyperameters are tuned on the dev-test set. As depicted in Table 1, our method significantly outperforms the L_1 method.

In addition, Table 1 presents the number of both active and reused features for each method on different settings. We can see that the active features (503 grouped features) in OSCAR method are much less than those (11081 features) in PRO with +id setting, which means that OSCAR has lower model complexity. Further, most (84.5%) of active features tuned on dev set are be used during testing for OSCAR, which means that OSCAR is more efficient to address feature sparsity problem compared with both L_1 and PRO.

At last, Table 1 also shows the runtimes for each tuning method. Tuning on training data is much inefficient compared with tuning on dev set, since it requires repeatedly decoding on a much larger dataset. Furthermore, the efficiency of our OSCAR method is comparable to that of tuning on training data. Anyway, distributed training is a reasonable approach to improve the efficiency of OSCAR, as suggested by Simianer et al. (2012).

6 Conclusion and Future Work

This paper proposes a novel training method for a translation model with a large number of features, which is the main contribution of this paper. This method is based on automatic feature grouping, which is implemented within an online learning method and thus is efficient for large scale training in SMT. The other contribution is that we successfuly extend OSCAR to a large scale of learning setting. In future work, we will investigate distributed learning for OSCAR and then testify it on larger scale training data.

Acknowledgments

We would like to thank our colleagues in both HIT and NICT for insightful discussions, and three anonymous reviewers for many invaluable comments and suggestions to improve our paper. This work is supported by National Natural Science Foundation of China (61173073, 61100093, 61073130, 61272384), and the Key Project of the National High Technology Research and Development Program of China (2011AA01A207).

References

H. D. Bondell and B. J. Reich. 2008. Simultaneous regression shrinkage, variable selection, and supervised clustering of predictors with oscar. *Biometrics*, 64(1):115–123.

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

David Chiang, Kevin Knight, and Wei Wang. 2009. 11,001 new features for statistical machine translation. In *NAACL*, NAACL ’09, pages 218–226.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *ACL*, ACL ’05, pages 263–270.

John Duchi and Yoram Singer. 2009. Efficient online and batch learning using forward backward splitting. *J. Mach. Learn. Res.*, 10:2899–2934, December.

Mark Hopkins and Jonathan May. 2011. Tuning as ranking. In *EMNLP*, pages 1352–1362, July.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proc. of EMNLP*. ACL.

Mu Li, Yinggong Zhao, Dongdong Zhang, and Ming Zhou. 2010. Adaptive development data selection for log-linear model in statistical machine translation. In *COLING*, COLING '10, pages 662–670.

J. B. MacQueen. 1967. Some methods for classification and analysis of multivariate observations. In *Proc. of 5-th Berkeley Symposium on Mathematical Statistics and Probability*.

Franz Josef Och and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proc. of ACL*.

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

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proc. of ACL*.

Patrick Simianer, Stefan Riezler, and Chris Dyer. 2012. Joint feature selection in distributed stochastic learning for large-scale discriminative training in smt. In *ACL*, ACL '12, pages 11–21.

Yoshimasa Tsuruoka, Jun'ichi Tsujii, and Sophia Ananiadou. 2009. Stochastic gradient descent training for l1-regularized log-linear models with cumulative penalty. In *ACL-IJCNLP*, ACL '09, pages 477–485.

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

Xinyan Xiao, Yang Liu, Qun Liu, and Shouxun Lin. 2011. Fast generation of translation forest for large-scale smt discriminative training. In *EMNLP*, pages 880–888.

Wenliang Zhong and James Kwok. 2011. Efficient sparse modeling with automatic feature grouping. In *ICML*, ICML '11, pages 9–16.

Appendix

Proof. Suppose $\hat{W}_k \neq 0$, and thus $|\hat{W}_k| > 0$. Set $\hat{W}'$ as another weight such that $\hat{W}'_j = \hat{W}_j$ for all $j(j \neq k)$, and $\hat{W}'_k = 0$. Then, for each i, j the following equations hold:

$$|\hat{W}_i| \geq |\hat{W}'_i|,$$

and

$$\max\{|\hat{W}_i|, |\hat{W}_j|\} \geq \max\{|\hat{W}'_i|, |\hat{W}'_j|\}.$$

Thus, the following equations hold based on the above equations by simple algebraic operations:

$$Q(\hat{W}; a, t, \lambda_1, \lambda_2) - Q(\hat{W}'; a, t, \lambda_1, \lambda_2)$$
$$= \hat{W}_k \times \hat{W}_k + \frac{2\lambda_1}{t} \sum_{i=1}^{d} \big(|\hat{W}_i| - |\hat{W}'_i|\big) + \frac{2\lambda_2}{t} \times$$
$$\sum_{1 \leq i < j \leq d} \big(\max\{|\hat{W}_i|, |\hat{W}_j|\} - \max\{|\hat{W}'_i|, |\hat{W}'_j|\}\big)$$
$$\geq \hat{W}_k \times \hat{W}_k > 0.$$

Therefore, we conclude that $Q(\hat{W}; a, t, \lambda_1, \lambda_2) > Q(\hat{W}'; a, t, \lambda_1, \lambda_2)$. This contradicts the assumption that $\hat{W}$ is the minimal solution of Eq.3. $\square$