Locally Training the Log-Linear Model for SMT

Lemao Liu[1], **Hailong Cao**[1], **Taro Watanabe**[2], **Tiejun Zhao**[1], **Mo Yu**[1], **CongHui Zhu**[1]
[1]School of Computer Science and Technology
Harbin Institute of Technology, Harbin, China
[2]National Institute of Information and Communication Technology
3-5 Hikari-dai, Seika-cho, Soraku-gun, Kyoto, Japan
{lmliu,hailong,tjzhao,yumo,chzhu}@mtlab.hit.edu.cn
taro.watanabe@nict.go.jp

Abstract

In statistical machine translation, minimum error rate training (MERT) is a standard method for tuning a single weight with regard to a given development data. However, due to the diversity and uneven distribution of source sentences, there are two problems suffered by this method. First, its performance is highly dependent on the choice of a development set, which may lead to an unstable performance for testing. Second, translations become inconsistent at the sentence level since tuning is performed globally on a document level. In this paper, we propose a novel local training method to address these two problems. Unlike a global training method, such as MERT, in which a single weight is learned and used for all the input sentences, we perform training and testing in one step by learning a sentence-wise weight for each input sentence. We propose efficient incremental training methods to put the local training into practice. In NIST Chinese-to-English translation tasks, our local training method significantly outperforms MERT with the maximal improvements up to 2.0 BLEU points, meanwhile its efficiency is comparable to that of the global method.

1 Introduction

Och and Ney (2002) introduced the log-linear model for statistical machine translation (SMT), in which translation is considered as the following optimization problem:

$$\begin{aligned}\hat{e}(f;W) &= \arg\max_{e} \mathbf{P}(e|f;W) \\ &= \arg\max_{e} \frac{\exp\{W \cdot h(f,e)\}}{\sum_{e'} \exp\{W \cdot h(f,e')\}} \\ &= \arg\max_{e} \{W \cdot h(f,e)\}, \end{aligned} \quad (1)$$

where f and e (e') are source and target sentences, respectively. h is a feature vector which is scaled by a weight W. Parameter estimation is one of the most important components in SMT, and various training methods have been proposed to tune W. Some methods are based on likelihood (Och and Ney, 2002; Blunsom et al., 2008), error rate (Och, 2003; Zhao and Chen, 2009; Pauls et al., 2009; Galley and Quirk, 2011), margin (Watanabe et al., 2007; Chiang et al., 2008) and ranking (Hopkins and May, 2011), and among which minimum error rate training (MERT) (Och, 2003) is the most popular one.

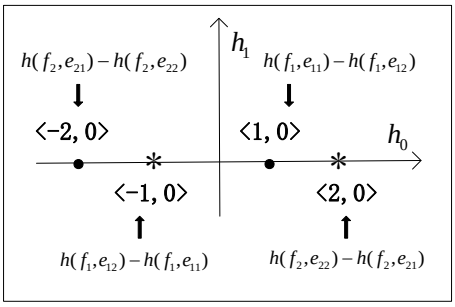
All these training methods follow the same pipeline: they train only a single weight on a given development set, and then use it to translate all the sentences in a test set. We call them a global training method. One of its advantages is that it allows us to train a single weight offline and thereby it is efficient. However, due to the diversity and uneven distribution of source sentences(Li et al., 2010), there are some shortcomings in this pipeline.

Firstly, on the document level, the performance of these methods is dependent on the choice of a development set, which may potentially lead to an unstable translation performance for testing. As referred in our experiment, the BLEU points on NIST08 are

Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pages 402–411, Jeju Island, Korea, 12–14 July 2012.

Source		Candidate Translation			
i	f_i	j	e_{ij}	h	$score$
1	我 是 学生 。	1	I am students .	<2, 1>	0.5
		2	I was students .	<1,1>	0.2
2	今天 星期 几 ？	1	week several today ?	<1,2>	0.3
		2	today several weeks .	<3,2>	0.1

(a)



(b)

Figure 1: (a). An Example candidate space of dimensionality two. $score$ is a evaluation metric of e. (b). The non-linearly separable classification problem transformed from (a) via tuning as ranking (Hopkins and May, 2011). Since score of e_{11} is greater than that of e_{12}, $\langle 1, 0\rangle$ corresponds to a possitive example denoted as "•", and $\langle -1, 0\rangle$ corresponds to a negative example denoted as "*". Since the transformed classification problem is not linearly separable, there does not exist a single weight which can obtain e_{11} and e_{21} as translation results meanwhile. However, one can obtain e_{11} and e_{21} with weights: $\langle 1, 1\rangle$ and $\langle -1, 1\rangle$, respectively.

19.04 when the Moses system is tuned on NIST02 by MERT. However, its performance is improved to 21.28 points when tuned on NIST06. The automatic selection of a development set may partially address the problem. However it is inefficient since tuning requires iteratively decoding an entire development set, which is impractical for an online service.

Secondly, translation becomes inconsistent on the sentence level (Ma et al., 2011). Global training method such as MERT tries to optimize the weight towards the best performance for the whole set, and it can not necessarily always obtain good translation for every sentence in the development set. The reason is that different sentences may need different optimal weights, and MERT can not find a single weight to satisfy all of the sentences. **Figure** 1(a) shows such an example, in which a development set contains two sentences f_1 and f_2 with translations e and feature vectors h. When we tune examples in **Figure** 1(a) by MERT, it can be regarded as a non-linearly separable classification problem illustrated in **Figure** 1(b). Therefore, there exists no single weight W which simultaneously obtains e_{11} and e_{21} as translation for f_1 and f_2 via Equation (1). However, we can achieve this with two weights: $\langle 1, 1\rangle$ for f_1 and $\langle -1, 1\rangle$ for f_2.

In this paper, inspired by KNN-SVM (Zhang et al., 2006), we propose a local training method, which trains sentence-wise weights instead of a single weight, to address the above two problems. Compared with global training methods, such as MERT, in which training and testing are separated, our method works in an online fashion, in which training is performed during testing. This online fashion has an advantage in that it can adapt the weights for each of the test sentences, by dynamically tuning the weights on translation examples which are similar to these test sentences. Similar to the method of development set automatical selection, the local training method may also suffer the problem of efficiency. To put it into practice, we propose incremental training methods which avoid retraining and iterative decoding on a development set.

Our local training method has two advantages: firstly, it significantly outperforms MERT, especially when test set is different from the development set; secondly, it improves the translation consistency. Experiments on NIST Chinese-to-English translation tasks show that our local training method significantly gains over MERT, with the maximum improvements up to 2.0 BLEU, and its efficiency is comparable to that of the global training method.

2 Local Training and Testing

The local training method (Bottou and Vapnik, 1992) is widely employed in computer vision (Zhang et al., 2006; Cheng et al., 2010). Compared with the global training method which tries to fit a single weight on the training data, the local one learns weights based on the local neighborhood information for each test example. It is superior to

the global one when the data sets are not evenly distributed (Bottou and Vapnik, 1992; Zhang et al., 2006).

Algorithm 1 Naive Local Training Method

Input: $T = \{t_i\}_{i=1}^{N}$(test set), K (retrieval size), Dev(development set), D(retrieval data)
Output: Translation results of T

1: **for all** sentence t_i such that $1 \leq i \leq N$ **do**
2: Retrieve the training examples D^i with size K for t_i from D according to a similarity;
3: Train a local weight W^i based on Dev and D^i;
4: Decode t_i with W^i;
5: **end for**

Suppose T be a test set, Dev a development set, and D a retrieval data. The local training in SMT is described in the **Algorithm** 1. For each sentence t_i in test set, training examples D^i is retrieved from D using a similarity measure (line 2), a weight W^i is optimized on Dev and D^i (line 3)[1], and, finally, t_i is decoded with W^i for testing (line 4). At the end of this algorithm, it returns the translation results for T. Note that weights are adapted for each test sentence t_i in line 3 by utilizing the translation examples D^i which are similar to t_i. Thus, our local training method can be considered as an adaptation of translation weights.

Algorithm 1 suffers a problem of training efficiency in line 3. It is impractical to train a weight W^i on Dev and D^i from scratch for every sentence, since iteratively decoding Dev and D^i is time consuming when we apply MERT. To address this problem, we propose a novel incremental approach which is based on a two-phase training.

On the first phase, we use a global training method, like MERT, to tune a baseline weight on the development set Dev in an offline manner. On the second phase, we utilize the retrieved examples to incrementally tune sentence-wise local weights based on the baseline weight. This method can not only consider the common characteristics learnt from the Dev, but also take into account the knowledge for each individual sentence learnt from similar examples during testing. On the phase of incremental training, we perform decoding only once for retrieved examples D^i, though several rounds of decoding are possible and potentially better if one does not seriously care about training speed. Furthermore, instead of on-the-fly decoding, we decode the retrieval data D offline using the parameter from our baseline weight and its nbest translation candidates are saved with training examples to increase the training efficiency.

[1]Usually, the quality of development set Dev is high, since it is manually produced with multiple references. This is the main reason why Dev is used as a part of new development set to train W^i.

Algorithm 2 Local Training Method Based on Incremental Training

Input: $T = \{t_i\}_{i=1}^{N}$ (test set), K (retrieval size), Dev (development set), $D = \{\langle f_s, \mathbf{r}_s \rangle\}_{s=1}^{s=S}$ (retrieval data),
Output: Translation results of T

1: Run global Training (such as MERT) on Dev to get a baseline weight W_b; **// Phase 1**
2: Decode each sentence in D to get $D = \{\langle f_s, \mathbf{c}_s, \mathbf{r}_s \rangle\}_{s=1}^{s=S}$;
3: **for all** sentence t_i such that $1 \leq i \leq N$ **do**
4: Retrieve K training examples $D^i = \{\langle f_j^i, \mathbf{c}_j^i, \mathbf{r}_j^i \rangle\}_{j=1}^{j=K}$ for t_i from D according to a similarity;
5: Incrementally train a local weight W^i based on W_b and D^i; **// Phase 2**
6: Decode t_i with W^i;
7: **end for**

The two-phase local training algorithm is described in **Algorithm** 2, where $\mathbf{c}_s$ and $\mathbf{r}_s$ denote the translation candidate set and reference set for each sentence f_s in retrieval data, respectively, and K is the retrieval size. It globally trains a baseline weight W_b (line 1), and decodes each sentence in retrieval data D with the weight W_b (line 2). For each sentence t_i in test set T, it first retrieves training examples D^i from D (line 4), and then it runs local training to tune a local weight W^i (line 5) and performs testing with W^i for t_i (line 6). Please note that the two-phase training contains global training in line 1 and local training in line 5.

From Algorithm 2, one can see that our method is effective even if the test set is unknow, for example, in the scenario of online translation services, since the global training on development set and decoding

on retrieval data can be performed offline.

In the next two sections, we will discuss the details about the similarity metric in line 4 and the incremental training in line 5 of Algorithm 2.

3 Acquiring Training Examples

In line 4 of Algorithm 2, to retrieve training examples for the sentence t_i , we first need a metric to retrieve similar translation examples. We assume that the metric satisfy the property: more similar the test sentence and translation examples are, the better translation result one obtains when decoding the test sentence with the weight trained on the translation examples.

The metric we consider here is derived from an example-based machine translation. To retrieve translation examples for a test sentence, (Watanabe and Sumita, 2003) defined a metric based on the combination of edit distance and TF-IDF (Manning and Schütze, 1999) as follows:

$$\begin{aligned}\text{dist}(f_1, f_2) = \theta \times \text{edit-dist}(f_1, f_2) + \\ (1-\theta) \times \text{tf-idf}(f_1, f_2),\end{aligned} \quad (2)$$

where $\theta (0 \leq \theta \leq 1)$ is an interpolation weight, $f_i (i = 1, 2)$ is a word sequence and can be also considered as a document. In this paper, we extract similar examples from training data. Like example-based translation in which similar source sentences have similar translations, we assume that the optimal translation weights of the similar source sentences are closer.

4 Incremental Training Based on Ultraconservative Update

Compared with retraining mode, incremental training can improve the training efficiency. In the field of machine learning research, incremental training has been employed in the work (Cauwenberghs and Poggio, 2001; Shilton et al., 2005), but there is little work for tuning parameters of statistical machine translation. The biggest difficulty lies in that the feature vector of a given training example, i.e. translation example, is unavailable until actually decoding the example, since the derivation is a latent variable. In this section, we will investigate the incremental training methods in SMT scenario.

Following the notations in Algorithm 2, W_b is the baseline weight, $D^i = \{\langle f_j^i, \mathbf{c}_j^i, \mathbf{r}_j^i \rangle\}_{j=1}^K$ denotes training examples for t_i. For the sake of brevity, we will drop the index i, $D^i = \{\langle f_j, \mathbf{c}_j, \mathbf{r}_j \rangle\}_{j=1}^K$, in the rest of this paper. Our goal is to find an optimal weight, denoted by W^i, which is a local weight and used for decoding the sentence t_i. Unlike the global method which performs tuning on the whole development set $Dev + D^i$ as in Algorithm 1, W^i can be incrementally learned by optimizing on D^i based on W_b. We employ the idea of ultraconservative update (Crammer and Singer, 2003; Crammer et al., 2006) to propose two incremental methods for local training in Algorithm 2 as follows.

Ultraconservative update is an efficient way to consider the trade-off between the progress made on development set Dev and the progress made on D^i. It desires that the optimal weight W^i is not only close to the baseline weight W_b, but also achieves the low loss over the retrieved examples D^i. The idea of ultraconservative update can be formalized as follows:

$$\min_W \Big\{ \mathbf{d}(W, W_b) + \lambda \cdot \mathbf{Loss}(D^i, W) \Big\}, \quad (3)$$

where $\mathbf{d}(W, W_b)$ is a distance metric over a pair of weights W and W_b. It penalizes the weights far away from W_b and it is L_2 norm in this paper. $\mathbf{Loss}(D^i, W)$ is a loss function of W defined on D^i and it evaluates the performance of W over D^i. λ is a positive hyperparameter. If D^i is more similar to the test sentence t_i, the better performance will be achieved for the larger λ. In particular, if D^i consists of only a single sentence t_i, the best performance will be obtained when λ goes to infinity.

4.1 Margin Based Ultraconservative Update

MIRA(Crammer and Singer, 2003; Crammer et al., 2006) is a form of ultraconservative update in (3) whose **Loss** is defined as hinge loss based on margin over the pairwise translation candiates in D^i. It tries to minimize the following quadratic program:

$$\frac{1}{2}||W - W_b||^2 + \frac{\lambda}{K} \sum_{j=1}^{K} \max_{1 \leq n \leq |\mathbf{c}_j|} \big(\ell_{jn} - W \cdot \Delta h(f_j, e_{jn})\big)$$

with

$$\Delta h(f_j, e_{jn}) = h(f_j, e_{j\cdot}) - h(f_j, e_{jn}), \quad (4)$$

where $h(f_j, e)$ is the feature vector of candidate e, e_{jn} is a translation member of f_j in $\mathbf{c}_j$, $e_{j\cdot}$ is the oracle one in $\mathbf{c}_j$, ℓ_{jn} is a loss between $e_{j\cdot}$ and e_{jn} and it is the same as referred in (Chiang et al., 2008), and $|\mathbf{c}_j|$ denotes the number of members in $\mathbf{c}_j$.

Different from (Watanabe et al., 2007; Chiang et al., 2008) employing the MIRA to globally train SMT, in this paper, we apply MIRA as one of local training method for SMT and we call it as margin based ultraconservative update (MBUU for shortly) to highlight its advantage of incremental training in line 5 of Algorithm 2.

Further, there is another difference between MBUU and MIRA in (Watanabe et al., 2007; Chiang et al., 2008). MBUU is a batch update mode which updates the weight with all training examples, but MIRA is an online one which updates with each example (Watanabe et al., 2007) or part of examples (Chiang et al., 2008). Therefore, MBUU is more ultraconservative.

4.2 Error Rate Based Ultraconservative Update

Instead of taking into account the margin-based hinge loss between a pair of translations as the $\mathbf{L}oss$ in (3), we directly optimize the error rate of translation candidates with respect to their references in D^i. Formally, the objective function of error rate based ultraconservative update (EBUU) is as follows:

$$\frac{1}{2}\|W - W_b\|^2 + \frac{\lambda}{K}\sum_{j=1}^{K}\mathbf{Error}(\mathbf{r}_j; \hat{e}(f_j; W)), \quad (5)$$

where $\hat{e}(f_j; W)$ is defined in Equation (1), and $\mathbf{Error}(\mathbf{r}_j, e)$ is the sentence-wise minus BLEU (Papineni et al., 2002) of a candidate e with respect to $\mathbf{r}_j$.

Due to the existence of L_2 norm in objective function (5), the optimization algorithm MERT can not be applied for this question since the exact line search routine does not hold here. Motivated by (Och, 2003; Smith and Eisner, 2006), we approximate the $\mathbf{Error}$ in (5) by the expected loss, and then derive the following function:

$$\frac{1}{2}\|W-W_b\|^2+\frac{\lambda}{K}\sum_{j=1}^{K}\sum_{e}\mathbf{Error}(\mathbf{r}_j; e)\mathbf{P}_\alpha(e|f_j; W), \quad (6)$$

Systems	NIST02	NIST05	NIST06	NIST08
Moses	30.39	26.31	25.34	19.07
Moses_hier	33.68	26.94	26.28	18.65
In-Hiero	31.24	27.07	26.32	19.03

Table 1: The performance comparison of the baseline In-Hiero VS Moses and Moses_hier.

with

$$\mathbf{P}_\alpha(e|f_j; W) = \frac{\exp[\alpha W \cdot h(f_j, e)]}{\sum_{e' \in \mathbf{c}_j}\exp[\alpha W \cdot h(f_j, e')]}, \quad (7)$$

where $\alpha > 0$ is a real number valued smoother. One can see that, in the extreme case, for $\alpha \to \infty$, (6) converges to (5).

We apply the gradient decent method to minimize the function (6), as it is smooth with respect to λ. Since the function (6) is non-convex, the solution obtained by gradient descent method may depend on the initial point. In this paper, we set the initial point as W_b in order to achieve a desirable solution.

5 Experiments and Results

5.1 Setting

We conduct our experiments on the Chinese-to-English translation task. The training data is FBIS corpus consisting of about 240k sentence pairs. The development set is NIST02 evaluation data, and the test datasets are NIST05, NIST06,and NIST08.

We run GIZA++ (Och and Ney, 2000) on the training corpus in both directions (Koehn et al., 2003) to obtain the word alignment for each sentence pair. We train a 4-gram language model on the Xinhua portion of the English Gigaword corpus using the SRILM Toolkits (Stolcke, 2002) with modified Kneser-Ney smoothing (Chen and Goodman, 1998). In our experiments the translation performances are measured by case-insensitive BLEU4 metric (Papineni et al., 2002) and we use mteval-v13a.pl as the evaluation tool. 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 system, and we denote it as **In-Hiero**. To obtain satisfactory baseline performance, we tune In-Hiero system for 5 times using MERT, and then se-

Methods	Steps	Seconds
Global method	Decoding	2.0
Local method	Retrieval	+0.6
	Local training	+0.3

Table 2: The efficiency of the local training and testing measured by sentence averaged runtime.

Methods		NIST05	NIST06	NIST08
Global	MERT	27.07	26.32	19.03
Local	MBUU	27.75^{+}	27.88^{+}	20.84^{+}
	EBUU	27.85^{+}	27.99^{+}	21.08^{+}

Table 3: The performance comparison of local training methods (MBUU and EBUU) and a global method (MERT). NIST05 is the set used to tune λ for MBUU and EBUU, and NIST06 and NIST08 are test sets. + means the local method is significantly better than MERT with $p < 0.05$.

lect the best-performing one as our baseline for the following experiments. As Table 1 indicates, our baseline In-Hiero is comparable to the phrase-based MT (Moses) and the hierarchical phrase-based MT (Moses_hier) implemented in Moses, an open source MT toolkit[2] (Koehn et al., 2007). Both of these systems are with default setting. All three systems are trained by MERT with 100 best candidates.

To compare the local training method in Algorithm 2, we use a standard global training method, MERT, as the baseline training method. We do not compare with Algorithm 1, in which retraining is performed for each input sentence, since retraining for the whole test set is impractical given that each sentence-wise retraining may take some hours or even days. Therefore, we just compare Algorithm 2 with MERT.

5.2 Runtime Results

To run the Algorithm 2, we tune the baseline weight W_b on NIST02 by MERT[3]. The retrieval data is set as the training data, i.e. FBIS corpus, and the retrieval size is 100. We translate retrieval data with W_b to obtain their 100 best translation candidates. We use the simple linear interpolated TF-IDF metric with $\theta = 0.1$ in Section 3 as the retrieval metric.

[2]See web: http://www.statmt.org

[3]W_b is exactly the weight of **In-Hiero** in Table 1.

	NIST05	NIST06	NIST08
NIST02	0.665	0.571	0.506

Table 4: The similarity of development and three test datasets.

For an efficient tuning, the retrieval process is parallelized as follows: the examples are assigned to 4 CPUs so that each CPU accepts a query and returns its top-100 results, then all these top-100 results are merged into the final top-100 retrieved examples together with their translation candidates. In our experiments, we employ the two incremental training methods, i.e. MBUU and EBUU. Both of the hyperparameters λ are tuned on NIST05 and set as 0.018 and 0.06 for MBUU and EBUU, respectively. In the incremental training step, only one CPU is employed.

Table 2 depicts that testing each sentence with local training method takes 2.9 seconds, which is comparable to the testing time 2.0 seconds with global training method[4]. This shows that the local method is efficient. Further, compared to the retrieval, the local training is not the bottleneck. Actually, if we use LSH technique (Andoni and Indyk, 2008) in retrieval process, the local method can be easily scaled to a larger training data.

5.3 Results and Analysis

Table 3 shows the main results of our local training methods. The EBUU training method significantly outperforms the MERT baseline, and the improvement even achieves up to 2.0 BLEU points on NIST08. We can also see that EBUU and MBUU are comparable on these three test sets. Both of these two local training methods achieve significant improvements over the MERT baseline, which proves the effectiveness of our local training method over global training method.

Although both local methods MBUU and EBUU achieved improvements on all the datasets, their gains on NIST06 and NIST08 are significantly higher than those achieved on NIST05 test dataset. We conjecture that, the more different a test set and a development set are, the more potential improvem-

[4]The runtime excludes the time of tuning and decoding on D in Algorithm 2, since both of them can be performanced offline.

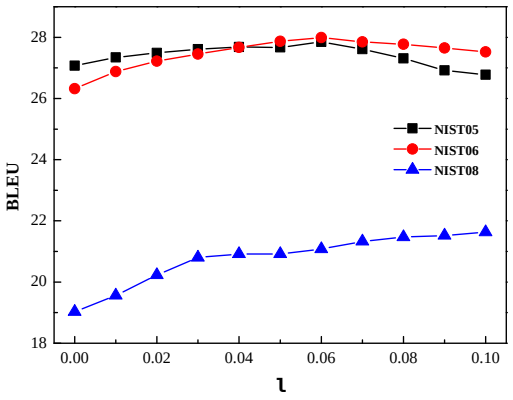


Figure 2: The peformance of EBUU for different λ over all the test datasets. The horizontal axis denotes the values of λ in function (6), and the vertical one denotes the BLEU points.

Metthods	Dev	NIST08
MERT	NIST02	19.03
	NIST05	20.06
	NIST06	21.28
EBUU	NIST02	21.08

Table 5: The comparison of MERT with different development datasets and local training method based on EBUU.

nts local training has for the sentences in this test set. To test our hypothesis, we measured the similarity between the development set and a test set by the average value[5] of accumulated TF-IDF scores of development dataset and each sentence in test datasets. Table 4 shows that NIST06 and NIST08 are more different from NIS02 than NIST05, thus, this is potentially the reason why local training is more effective on NIST06 and NIST08.

As mentioned in Section 1, the global training methods such as MERT are highly dependent on development sets, which can be seen in Table 5. Therefore, the translation performance will be degraded if one chooses a development data which is not close

[5]Instead of using the similarity between two documents development and test datasets, we define the similarity as the average similarity of the development set and the sentences in test set. The reason is that it reduces its dependency on the number of sentences in test dataset, which may cause a bias.

Methods	Number	Percents
MERT	1735	42.3%
EBUU	1606	39.1%

Table 6: The statistics of sentences with 0.0 sentence-level BLEU points over three test datasets.

to the test data. We can see that, with the help of the local training, we still gain much even if we selected an unsatisfactory development data.

As also mentioned in Section 1, the global methods do not care about the sentence level performance. Table 6 depicts that there are 1735 sentences with zero BLEU points in all the three test datasets for MERT. Besides obtaining improvements on document level as referred in Table 3, the local training methods can also achieve consistent improvements on sentence level and thus can improve the users' experiences.

The hyperparameters λ in both MBUU (4) and EBUU (6) has an important influence on translation performance. Figure 2 shows such influence for EBUU on the test datasets. We can see that, the performances on all these datasets improve as λ becomes closer to 0.06 from 0, and the performance continues improving when λ passes over 0.06 on NIST08 test set, where the performance constantly improves up to 2.6 BLEU points over baseline. As mentioned in Section 4, if the retrieved examples are very similar to the test sentence, the better performance will be achieved with the larger λ. Therefore, it is reasonable that the performances improved when λ increased from 0 to 0.06. Further, the turning point appearing at 0.06 proves that the ultra-conservative update is necessary. We can also see that the performance on NIST08 consistently improves and achieves the maximum gain when λ arrives at 0.1, but those on both NIST05 and NIST06 achieves the best when it arrives at 0.06. This phenomenon can also be interpreted in Table 4 as the lowest similarity between the development and NIST08 datasets.

Generally, the better performance may be achieved when more examples are retrieved. Actually, in Table 7 there seems to be little dependency between the numbers of examples retrieved and the translation qualities, although they are positively re-

Retrieval Size	NIST05	NIST06	NIST08
40	27.66	27.81	20.87
70	27.77	27.93	21.08
100	27.85	27.99	21.08

Table 7: The performance comparison by varying retrieval size in Algorithm 2 based on EBUU.

Methods	NIST05	NIST06	NIST08
MERT	27.07	26.32	19.03
EBUU	27.85	27.99	21.08
Oracle	29.46	29.35	22.09

Table 8: The performance of Oracle of 2-best results which consist of 1-best resluts of MERT and 1-best resluts of EBUU.

lated approximately.

Table 8 presents the performance of the oracle translations selected from the 1-best translation results of MERT and EBUU. Clearly, there exists more potential improvement for local training method.

6 Related Work

Several works have proposed discriminative techniques to train log-linear model for SMT. (Och and Ney, 2002; Blunsom et al., 2008) used maximum likelihood estimation to learn weights for MT. (Och, 2003; Moore and Quirk, 2008; Zhao and Chen, 2009; Galley and Quirk, 2011) employed an evaluation metric as a loss function and directly optimized it. (Watanabe et al., 2007; Chiang et al., 2008; Hopkins and May, 2011) proposed other optimization objectives by introducing a margin-based and ranking-based indirect loss functions.

All the methods mentioned above train a single weight for the whole development set, whereas our local training method learns a weight for each sentence. Further, our translation framework integrates the training and testing into one unit, instead of treating them separately. One of the advantages is that it can adapt the weights for each of the test sentences.

Our method resorts to some translation examples, which is similar as example-based translation or translation memory (Watanabe and Sumita, 2003; He et al., 2010; Ma et al., 2011). Instead of using translation examples to construct translation rules for enlarging the decoding space, we employed them to discriminatively learn local weights.

Similar to (Hildebrand et al., 2005; Lü et al., 2007), our method also employes IR methods to retrieve examples for a given test set. Their methods utilize the retrieved examples to acquire translation model and can be seen as the adaptation of translation model. However, ours uses the retrieved examples to tune the weights and thus can be considered as the adaptation of tuning. Furthermore, since ours does not change the translation model which needs to run GIZA++ and it incrementally trains local weights, our method can be applied for online translation service.

7 Conclusion and Future Work

This paper proposes a novel local training framework for SMT. It has two characteristics, which are different from global training methods such as MERT. First, instead of training only one weight for document level, it trains a single weight for sentence level. Second, instead of considering the training and testing as two separate units, we unify the training and testing into one unit, which can employ the information of test sentences and perform sentence-wise local adaptation of weights.

Local training can not only alleviate the problem of the development data selection, but also reduce the risk of sentence-wise bad translation results, thus consistently improve the translation performance. Experiments show gains up to 2.0 BLEU points compared with a MERT baseline. With the help of incremental training methods, the time incurred by local training was negligible and the local training and testing totally took 2.9 seconds for each sentence.

In the future work, we will further investigate the local training method, since there are more room for improvements as observed in our experiments. We will test our method on other translation models and larger training data[6].

Acknowledgments

We would like to thank Hongfei Jiang and Shujie Liu for many valuable discussions and thank three

[6]Intuitionally, when the corpus of translation examples is larger, the retrieval results in Algorithm 2 are much similar as the test sentence. Therefore our method may favor this.

anonymous reviewers for many valuable comments and helpful suggestions. This work was supported by National Natural Science Foundation of China (61173073,61100093), and the Key Project of the National High Technology Research and Development Program of China (2011AA01A207), and the Fundamental Research Funds for Central Universites (HIT.NSRIF.2013065).

References

Alexandr Andoni and Piotr Indyk. 2008. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Commun. ACM*, 51(1):117–122, January.

Phil Blunsom, Trevor Cohn, and Miles Osborne. 2008. A discriminative latent variable model for statistical machine translation. In *Proceedings of ACL*, pages 200–208, Columbus, Ohio, June. Association for Computational Linguistics.

Léon Bottou and Vladimir Vapnik. 1992. Local learning algorithms. *Neural Comput.*, 4:888–900, November.

G. Cauwenberghs and T. Poggio. 2001. Incremental and decremental support vector machine learning. In *Advances in Neural Information Processing Systems (NIPS*2000)*, volume 13.

Stanley F Chen and Joshua Goodman. 1998. An empirical study of smoothing techniques for language modeling. In *Technical Report TR-10-98*. Harvard University.

Haibin Cheng, Pang-Ning Tan, and Rong Jin. 2010. Efficient algorithm for localized support vector machine. *IEEE Trans. on Knowl. and Data Eng.*, 22:537–549, April.

David Chiang, Yuval Marton, and Philip Resnik. 2008. Online large-margin training of syntactic and structural translation features. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 224–233, Stroudsburg, PA, USA. Association for Computational Linguistics.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 263–270, Stroudsburg, PA, USA. Association for Computational Linguistics.

Koby Crammer and Yoram Singer. 2003. Ultraconservative online algorithms for multiclass problems. *J. Mach. Learn. Res.*, 3:951–991, March.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *J. Mach. Learn. Res.*, 7:551–585, December.

Michel Galley and Chris Quirk. 2011. Optimal search for minimum error rate training. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 38–49, Edinburgh, Scotland, UK., July. Association for Computational Linguistics.

Yifan He, Yanjun Ma, Josef van Genabith, and Andy Way. 2010. Bridging smt and tm with translation recommendation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 622–630, Uppsala, Sweden, July. Association for Computational Linguistics.

S. Hildebrand, M. Eck, S. Vogel, and Alex Waibel. 2005. Adaptation of the translation model for statistical machine translation based on information retrieval. In *Proceedings of EAMT*. Association for Computational Linguistics.

Mark Hopkins and Jonathan May. 2011. Tuning as ranking. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1352–1362, Edinburgh, Scotland, UK., July. Association for Computational Linguistics.

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

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ACL '07, pages 177–180, Stroudsburg, PA, USA. Association for Computational Linguistics.

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 *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 662–670, Stroudsburg, PA, USA. Association for Computational Linguistics.

Yajuan Lü, Jin Huang, and Qun Liu. 2007. Improving statistical machine translation performance by training data selection and optimization. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages

343–350, Prague, Czech Republic, June. Association for Computational Linguistics.

Yanjun Ma, Yifan He, Andy Way, and Josef van Genabith. 2011. Consistent translation using discriminative learning - a translation memory-inspired approach. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1239–1248, Portland, Oregon, USA, June. Association for Computational Linguistics.

Christopher D. Manning and Hinrich Schütze. 1999. *Foundations of statistical natural language processing*. MIT Press, Cambridge, MA, USA.

Robert C. Moore and Chris Quirk. 2008. Random restarts in minimum error rate training for statistical machine translation. In *Proceedings of the 22nd International Conference on Computational Linguistics - Volume 1*, COLING '08, pages 585–592, Stroudsburg, PA, USA. Association for Computational Linguistics.

Franz Josef Och and Hermann Ney. 2000. Improved statistical alignment models. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, ACL '00, pages 440–447, Stroudsburg, PA, USA. Association for Computational Linguistics.

Franz Josef Och and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 295–302, Stroudsburg, PA, USA. Association for Computational Linguistics.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 160–167, Sapporo, Japan, July. Association for Computational Linguistics.

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

Adam Pauls, John Denero, and Dan Klein. 2009. Consensus training for consensus decoding in machine translation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1418–1427, Singapore, August. Association for Computational Linguistics.

Alistair Shilton, Marimuthu Palaniswami, Daniel Ralph, and Ah Chung Tsoi. 2005. Incremental training of support vector machines. *IEEE Transactions on Neural Networks*, 16(1):114–131.

Andreas Stolcke. 2002. Srilm - an extensible language modeling toolkit. In *Proc. of ICSLP*.

Taro Watanabe and Eiichiro Sumita. 2003. Example-based decoding for statistical machine translation. In *Proc. of MT Summit IX*, pages 410–417.

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 Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 764–773, Prague, Czech Republic, June. Association for Computational Linguistics.

Hao Zhang, Alexander C. Berg, Michael Maire, and Jitendra Malik. 2006. Svm-knn: Discriminative nearest neighbor classification for visual category recognition. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2*, CVPR '06, pages 2126–2136, Washington, DC, USA. IEEE Computer Society.

Bing Zhao and Shengyuan Chen. 2009. A simplex armijo downhill algorithm for optimizing statistical machine translation decoding parameters. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, NAACL-Short '09, pages 21–24, Stroudsburg, PA, USA. Association for Computational Linguistics.