Boosting-based System Combination for Machine Translation

Tong Xiao, Jingbo Zhu, Muhua Zhu, Huizhen Wang

Natural Language Processing Lab.
Northeastern University, China
{xiaotong,zhujingbo,wanghuizhen}@mail.neu.edu.cn
zhumuhua@gmail.com

Abstract

In this paper, we present a simple and effective method to address the issue of how to generate diversified translation systems from a single Statistical Machine Translation (SMT) engine for system combination. Our method is based on the framework of boosting. First, a sequence of *weak* translation systems is generated from a baseline system in an iterative manner. Then, a *strong* translation system is built from the ensemble of these weak translation systems. To adapt boosting to SMT system combination, several key components of the original boosting algorithms are redesigned in this work. We evaluate our method on Chinese-to-English Machine Translation (MT) tasks in three baseline systems, including a phrase-based system, a hierarchical phrase-based system and a syntax-based system. The experimental results on three NIST evaluation test sets show that our method leads to significant improvements in translation accuracy over the baseline systems.

1 Introduction

Recent research on Statistical Machine Translation (SMT) has achieved substantial progress. Many SMT frameworks have been developed, including phrase-based SMT (Koehn et al., 2003), hierarchical phrase-based SMT (Chiang, 2005), syntax-based SMT (Eisner, 2003; Ding and Palmer, 2005; Liu et al., 2006; Galley et al., 2006; Cowan et al., 2006), etc. With the emergence of various structurally different SMT systems, more and more studies are focused on combining multiple SMT systems for achieving higher translation accuracy rather than using a single translation system.

The basic idea of system combination is to extract or generate a translation by voting from an ensemble of translation outputs. Depending on how the translation is combined and what voting strategy is adopted, several methods can be used for system combination, e.g. sentence-level combination (Hildebrand and Vogel, 2008) simply selects one from original translations, while some more sophisticated methods, such as word-level and phrase-level combination (Matusov et al., 2006; Rosti et al., 2007), can generate new translations differing from any of the original translations.

One of the key factors in SMT system combination is the diversity in the ensemble of translation outputs (Macherey and Och, 2007). To obtain diversified translation outputs, most of the current system combination methods require multiple translation engines based on different models. However, this requirement cannot be met in many cases, since we do not always have the access to multiple SMT engines due to the high cost of developing and tuning SMT systems. To reduce the burden of system development, it might be a nice way to combine a set of translation systems built from a single translation engine. A key issue here is how to generate an ensemble of diversified translation systems from a single translation engine in a principled way.

Addressing this issue, we propose a boosting-based system combination method to learn a combined translation system from a single SMT engine. In this method, a sequence of *weak* translation systems is generated from a baseline system in an iterative manner. In each iteration, a new weak translation system is learned, focusing more on the sentences that are relatively poorly translated by the previous weak translation system. Finally, a *strong* translation system is built from the ensemble of the weak translation systems.

Our experiments are conducted on Chinese-to-English translation in three state-of-the-art SMT systems, including a phrase-based system, a hierarchical phrase-based system and a syntax-based

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 739–748,
Uppsala, Sweden, 11-16 July 2010.

Input: a model u, a sequence of (training) samples $\{(f_1, \mathbf{r}_1), ..., (f_m, \mathbf{r}_m)\}$ where f_i is the i-th source sentence, and $\mathbf{r}_i$ is the set of reference translations for f_i.
Output: a new translation system

Initialize: $D_1(i) = 1 / m$ for all $i = 1, ..., m$
For $t = 1, ..., T$

1. Train a translation system $u(\boldsymbol{\lambda}^*_t)$ on $\{(f_i, \mathbf{r}_i)\}$ using distribution D_t
2. Calculate the error rate ε_t of $u(\boldsymbol{\lambda}^*_t)$ on $\{(f_i, \mathbf{r}_i)\}$
3. Set

$$\alpha_t = \frac{1}{2}\ln(\frac{1+\varepsilon_t}{\varepsilon_t}) \quad (3)$$

4. Update weights

$$D_{t+1}(i) = \frac{D_t(i)e^{\alpha_t \cdot l_i}}{Z_t} \quad (4)$$

where l_i is the loss on the i-th training sample, and Z_t is the normalization factor.

Output the final system:

$$v(u(\boldsymbol{\lambda}^*_1), ..., u(\boldsymbol{\lambda}^*_T))$$

Figure 1: Boosting-based System Combination

system. All the systems are evaluated on three NIST MT evaluation test sets. Experimental results show that our method leads to significant improvements in translation accuracy over the baseline systems.

2 Background

Given a source string f, the goal of SMT is to find a target string e^* by the following equation.

$$e^* = \arg\max_e(\Pr(e \mid f)) \quad (1)$$

where $\Pr(e \mid f)$ is the probability that e is the translation of the given source string f. To model the posterior probability $\Pr(e \mid f)$, most of the state-of-the-art SMT systems utilize the log-linear model proposed by Och and Ney (2002), as follows,

$$\Pr(e \mid f) = \frac{\exp(\sum_{m=1}^{M}\lambda_m \cdot h_m(f,e))}{\sum_{e'}\exp(\sum_{m=1}^{M}\lambda_m \cdot h_m(f,e'))} \quad (2)$$

where $\{h_m(f, e) \mid m = 1, ..., M\}$ is a set of *features*, and λ_m is the *feature weight* corresponding to the m-th feature. $h_m(f, e)$ can be regarded as a function that maps every pair of source string f and target string e into a non-negative value, and λ_m can be viewed as the contribution of $h_m(f, e)$ to the overall score $\Pr(e \mid f)$.

In this paper, u denotes a log-linear model that has M fixed features $\{h_1(f,e), ..., h_M(f,e)\}$, $\boldsymbol{\lambda} = \{\lambda_1, ..., \lambda_M\}$ denotes the M parameters of u, and $u(\boldsymbol{\lambda})$ denotes a SMT system based on u with parameters $\boldsymbol{\lambda}$. Generally, $\boldsymbol{\lambda}$ is trained on a training data set[1] to obtain an optimized weight vector $\boldsymbol{\lambda}^*$ and consequently an optimized system $u(\boldsymbol{\lambda}^*)$.

3 Boosting-based System Combination for Single Translation Engine

Suppose that there are T available SMT systems $\{u_1(\boldsymbol{\lambda}^*_1), ..., u_T(\boldsymbol{\lambda}^*_T)\}$, the task of system combination is to build a new translation system $v(u_1(\boldsymbol{\lambda}^*_1), ..., u_T(\boldsymbol{\lambda}^*_T))$ from $\{u_1(\boldsymbol{\lambda}^*_1), ..., u_T(\boldsymbol{\lambda}^*_T)\}$. Here $v(u_1(\boldsymbol{\lambda}^*_1), ..., u_T(\boldsymbol{\lambda}^*_T))$ denotes the *combination system* which combines translations from the ensemble of the output of each $u_i(\boldsymbol{\lambda}^*_i)$. We call $u_i(\boldsymbol{\lambda}^*_i)$ a *member system* of $v(u_1(\boldsymbol{\lambda}^*_1), ..., u_T(\boldsymbol{\lambda}^*_T))$. As discussed in Section 1, the diversity among the outputs of member systems is an important factor to the success of system combination. To obtain diversified member systems, traditional methods concentrate more on using structurally different member systems, that is $u_1 \neq u_2 \neq ... \neq u_T$. However, this constraint condition cannot be satisfied when multiple translation engines are not available.

In this paper, we argue that the diversified member systems can also be generated from a single engine $u(\boldsymbol{\lambda}^*)$ by adjusting the weight vector $\boldsymbol{\lambda}^*$ in a principled way. In this work, we assume that $u_1 = u_2 = ... = u_T = u$. Our goal is to find a series of $\boldsymbol{\lambda}^*_i$ and build a combined system from $\{u(\boldsymbol{\lambda}^*_i)\}$. To achieve this goal, we propose a

[1] The data set used for weight training is generally called *development set* or *tuning set* in the SMT field. In this paper, we use the term *training set* to emphasize the training of log-linear model.

boosting-based system combination method (Figure 1).

Like other boosting algorithms, such as AdaBoost (Freund and Schapire, 1997; Schapire, 2001), the basic idea of this method is to use weak systems (*member systems*) to form a strong system (*combined system*) by repeatedly calling weak system trainer on different distributions over the training samples. However, since most of the boosting algorithms are designed for the classification problem that is very different from the translation problem in natural language processing, several key components have to be redesigned when boosting is adapted to SMT system combination.

3.1 Training

In this work, Minimum Error Rate Training (MERT) proposed by Och (2003) is used to estimate feature weights $\boldsymbol{\lambda}$ over a series of training samples. As in other state-of-the-art SMT systems, BLEU is selected as the accuracy measure to define the error function used in MERT. Since the weights of training samples are not taken into account in BLEU[2], we modify the original definition of BLEU to make it sensitive to the distribution $D_t(i)$ over the training samples. The modified version of BLEU is called *weighted BLEU* (WBLEU) in this paper.

Let $E = e_1 \ldots e_m$ be the translations produced by the system, $R = \boldsymbol{r}_1 \ldots \boldsymbol{r}_m$ be the reference translations where $\boldsymbol{r}_i = \{r_{i1}, \ldots, r_{iN}\}$, and $D_t(i)$ be the weight of the i-th training sample $(f_i, \boldsymbol{r}_i)$. The weighted BLEU metric has the following form:

$$\begin{aligned}&\mathrm{WBLEU}(E,R)\\ &= \exp\left(1-\max\left\{1,\frac{\sum_{i=1}^{m} D_t(i)\min_{1\le j\le N}\{|\,g_1(r_{ij})\,|\}}{\sum_{i=1}^{m} D_t(i)\,|\,g_1(e_i)\,|}\right\}\right)\times\\ &\prod_{n=1}^{4}\left(\frac{\sum_{i=1}^{\mathrm{m}} D_t(i)\left|g_n(e_i)\bigcap\left(\bigcup_{j=1}^{N} g_n(r_{ij})\right)\right|}{\sum_{i=1}^{\mathrm{m}} D_t(i)\left|g_n(e_i)\right|}\right)^{1/4}\end{aligned} \quad (5)$$

where $g_n(s)$ is the multi-set of all n-grams in a string s. In this definition, n-grams in e_i and $\{r_{ij}\}$ are weighted by $D_t(i)$. If the i-th training sample has a larger weight, the corresponding n-grams will have more contributions to the overall score $\mathrm{WBLEU}(E,R)$. As a result, the i-th training sample gains more importance in MERT. Obviously the original BLEU is just a special case of WBLEU when all the training samples are equally weighted.

As the weighted BLEU is used to measure the translation accuracy on the training set, the error rate is defined to be:

$$\varepsilon_t = 1 - \mathrm{WBLEU}(E,R) \quad (6)$$

3.2 Re-weighting

Another key point is the maintaining of the distribution $D_t(i)$ over the training set. Initially all the weights of training samples are set equally. On each round, we increase the weights of the samples that are relatively poorly translated by the current weak system so that the MERT-based trainer can focus on the hard samples in next round. The update rule is given in Equation 4 with two parameters α_t and l_i in it.

α_t can be regarded as a measure of the importance that the t-th weak system gains in boosting. The definition of α_t guarantees that α_t always has a positive value[3]. A main effect of α_t is to scale the weight updating (e.g. a larger α_t means a greater update).

l_i is the loss on the i-th sample. For each i, let $\{e_{i1}, \ldots, e_{in}\}$ be the n-best translation candidates produced by the system. The loss function is defined to be:

$$l_i = \mathrm{BLEU}(e_i^*, \boldsymbol{r}_i) - \frac{1}{k}\sum_{j=1}^{k}\mathrm{BLEU}(e_{ij}, \boldsymbol{r}_i) \quad (7)$$

where $\mathrm{BLEU}(e_{ij}, \boldsymbol{r}_i)$ is the smoothed sentence-level BLEU score (Liang et al., 2006) of the translation e with respect to the reference translations $\boldsymbol{r}_i$, and e_i^* is the oracle translation which is selected from $\{e_{i1}, \ldots, e_{in}\}$ in terms of $\mathrm{BLEU}(e_{ij}, \boldsymbol{r}_i)$. l_i can be viewed as a measure of the average cost that we guess the top-k translation candidates instead of the oracle translation. The value of l_i counts for the magnitude of weight update, that is, a larger l_i means a larger weight update on $D_t(i)$. The definition of the loss function here is similar to the one used in (Chiang et al., 2008) where only the top-1 translation candidate (i.e. $k = 1$) is taken into account.

3.3 System Combination Scheme

In the last step of our method, a strong translation system $v(u(\boldsymbol{\lambda}^*_1), \ldots, u(\boldsymbol{\lambda}^*_T))$ is built from the

[2] In this paper, we use the NIST definition of BLEU where the *effective reference length* is the length of the shortest reference translation.

[3] Note that the definition of α_t here is different from that in the original AdaBoost algorithm (Freund and Schapire, 1997; Schapire, 2001) where α_t is a negative number when $\varepsilon_t > 0.5$.

ensemble of member systems $\{u(\boldsymbol{\lambda}^*_1), ..., u(\boldsymbol{\lambda}^*_T)\}$. In this work, a sentence-level combination method is used to select the best translation from the pool of the *n*-best outputs of all the member systems.

Let $H(u(\boldsymbol{\lambda}^*_t))$ (or H_t for short) be the set of the *n*-best translation candidates produced by the *t*-th member system $u(\boldsymbol{\lambda}^*_t)$, and $H(v)$ be the union set of all H_t (i.e. $H(v) = \bigcup H_t$). The final translation is generated from $H(v)$ based on the following scoring function:

$$e^* = \arg\max_{e \in H(v)} \sum_{t=1}^{T} \beta_t \cdot \phi_t(e) + \psi(e, H(v)) \quad (8)$$

where $\phi_t(e)$ is the log-scaled model score of *e* in the *t*-th member system, and β_t is the corresponding feature weight. It should be noted that $e \in H_i$ may not exist in any $H_{i' \neq i}$. In this case, we can still calculate the model score of *e* in any other member systems, since all the member systems are based on the same model and share the same feature space. $\psi(e, H(v))$ is a consensus-based scoring function which has been successfully adopted in SMT system combination (Duan et al., 2009; Hildebrand and Vogel, 2008; Li et al., 2009). The computation of $\psi(e, H(v))$ is based on a linear combination of a set of *n*-gram consensuses-based features.

$$\psi(e, H(v)) = \sum_n \theta_n^+ \cdot h_n^+(e, H(v)) + \sum_n \theta_n^- \cdot h_n^-(e, H(v)) \quad (9)$$

For each order of *n*-gram, $h_n^+(e, H(v))$ and $h_n^-(e, H(v))$ are defined to measure the *n*-gram agreement and disagreement between *e* and other translation candidates in $H(v)$, respectively. θ_n^+ and θ_n^- are the feature weights corresponding to $h_n^+(e, H(v))$ and $h_n^-(e, H(v))$. As $h_n^+(e, H(v))$ and $h_n^-(e, H(v))$ used in our work are exactly the same as the features used in (Duan et al., 2009) and similar to the features used in (Hildebrand and Vogel, 2008; Li et al., 2009), we do not present the detailed description of them in this paper.

If *p* orders of *n*-gram are used in computing $\psi(e, H(v))$, the total number of features in the system combination will be $T + 2 \times p$ (*T* model-score-based features defined in Equation 8 and $2 \times p$ consensus-based features defined in Equation 9). Since all these features are combined linearly, we use MERT to optimize them for the combination model.

4 Optimization

If implemented naively, the translation speed of the final translation system will be very slow. For a given input sentence, each member system has to encode it individually, and the translation speed is inversely proportional to the number of member systems generated by our method. Fortunately, with the thought of computation, there are a number of optimizations that can make the system much more efficient in practice.

A simple solution is to run member systems in parallel when translating a new sentence. Since all the member systems share the same data resources, such as language model and translation table, we only need to keep one copy of the required resources in memory. The translation speed just depends on the computing power of parallel computation environment, such as the number of CPUs.

Furthermore, we can use joint decoding techniques to save the computation of the equivalent translation hypotheses among member systems. In joint decoding of member systems, the search space is structured as a translation hypergraph where the member systems can share their translation hypotheses. If more than one member systems share the same translation hypothesis, we just need to compute the corresponding feature values only once, instead of repeating the computation in individual decoders. In our experiments, we find that over 60% translation hypotheses can be shared among member systems when the number of member systems is over 4. This result indicates that promising speed improvement can be achieved by using the joint decoding and hypothesis sharing techniques.

Another method to speed up the system is to accelerate *n*-gram language model with *n*-gram caching techniques. In this method, a *n*-gram cache is used to store the most frequently and recently accessed *n*-grams. When a new *n*-gram is accessed during decoding, the cache is checked first. If the required *n*-gram hits the cache, the corresponding *n*-gram probability is returned by the cached copy rather than refetching the original data in language model. As the translation speed of SMT system depends heavily on the computation of *n*-gram language model, the acceleration of *n*-gram language model generally leads to substantial speed-up of SMT system. In our implementation, the *n*-gram caching in general brings us over 30% speed improvement of the system.

5 Experiments

Our experiments are conducted on Chinese-to-English translation in three SMT systems.

5.1 Baseline Systems

The first SMT system is a phrase-based system with two reordering models including the maximum entropy-based lexicalized reordering model proposed by Xiong et al. (2006) and the hierarchical phrase reordering model proposed by Galley and Manning (2008). In this system all phrase pairs are limited to have source length of at most 3, and the reordering limit is set to 8 by default[4].

The second SMT system is an in-house reimplementation of the Hiero system which is based on the hierarchical phrase-based model proposed by Chiang (2005).

The third SMT system is a syntax-based system based on the string-to-tree model (Galley et al., 2006; Marcu et al., 2006), where both the minimal GHKM and SPMT rules are extracted from the bilingual text, and the composed rules are generated by combining two or three minimal GHKM and SPMT rules. Synchronous binarization (Zhang et al., 2006; Xiao et al., 2009) is performed on each translation rule for the CKY-style decoding.

In this work, baseline system refers to the system produced by the boosting-based system combination when the number of iterations (i.e. T) is set to 1. To obtain satisfactory baseline performance, we train each SMT system for 5 times using MERT with different initial values of feature weights to generate a group of *baseline candidates*, and then select the best-performing one from this group as the final baseline system (i.e. the starting point in the boosting process) for the following experiments.

5.2 Experimental Setup

Our bilingual data consists of 140K sentence pairs in the FBIS data set[5]. GIZA++ is employed to perform the bi-directional word alignment between the source and target sentences, and the final word alignment is generated using the intersect-diag-grow method. All the word-aligned bilingual sentence pairs are used to extract phrases and rules for the baseline systems. A 5-gram language model is trained on the target-side of the bilingual data and the Xinhua portion of English Gigaword corpus. Berkeley Parser is used to generate the English parse trees for the rule extraction of the syntax-based system. The data set used for weight training in boosting-based system combination comes from NIST MT03 evaluation set. To speed up MERT, all the sentences with more than 20 Chinese words are removed. The test sets are the NIST evaluation sets of MT04, MT05 and MT06. The translation quality is evaluated in terms of case-insensitive NIST version BLEU metric. Statistical significant test is conducted using the bootstrap re-sampling method proposed by Koehn (2004).

Beam search and cube pruning (Huang and Chiang, 2007) are used to prune the search space in all the three baseline systems. By default, both of the beam size and the size of n-best list are set to 20.

In the settings of boosting-based system combination, the maximum number of iterations is set to 30, and k (in Equation 7) is set to 5. The n-gram consensuses-based features (in Equation 9) used in system combination ranges from unigram to 4-gram.

5.3 Evaluation of Translations

First we investigate the effectiveness of the boosting-based system combination on the three systems.

Figures 2-5 show the BLEU curves on the development and test sets, where the X-axis is the iteration number, and the Y-axis is the BLEU score of the system generated by the boosting-based system combination. The points at iteration 1 stand for the performance of the baseline systems. We see, first of all, that all the three systems are improved during iterations on the development set. This trend also holds on the test sets. After 5, 7 and 8 iterations, relatively stable improvements are achieved by the phrase-based system, the Hiero system and the syntax-based system, respectively. The BLEU scores tend to converge to the stable values after 20 iterations for all the systems. Figures 2-5 also show that the boosting-based system combination seems to be more helpful to the phrase-based system than to the Hiero system and the syntax-based system. For the phrase-based system, it yields over 0.6 BLEU point gains just after the 3rd iteration on all the data sets.

Table 1 summarizes the evaluation results, where the BLEU scores at iteration 5, 10, 15, 20 and 30 are reported for the comparison. We see that the boosting-based system method stably ac-

[4] Our in-house experimental results show that this system performs slightly better than Moses on Chinese-to-English translation tasks.

[5] LDC catalog number: LDC2003E14

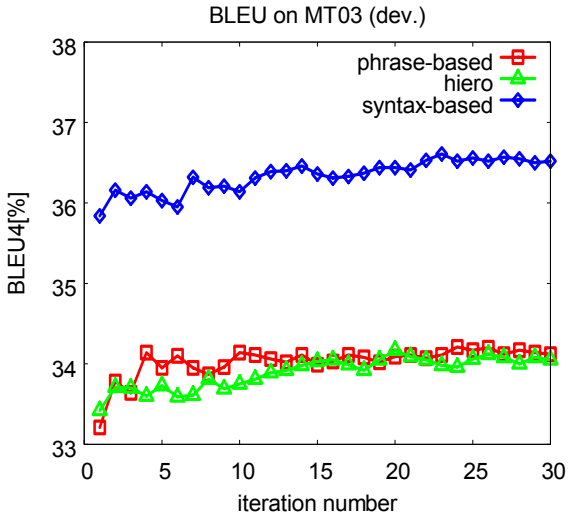

Figure 2: BLEU scores on the development set

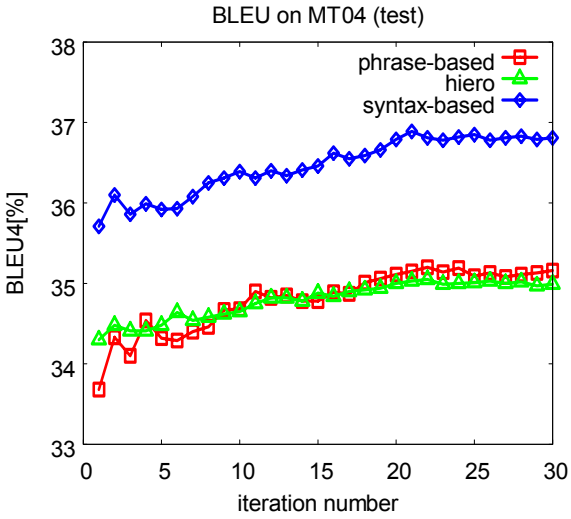

Figure 3: BLEU scores on the test set of MT04

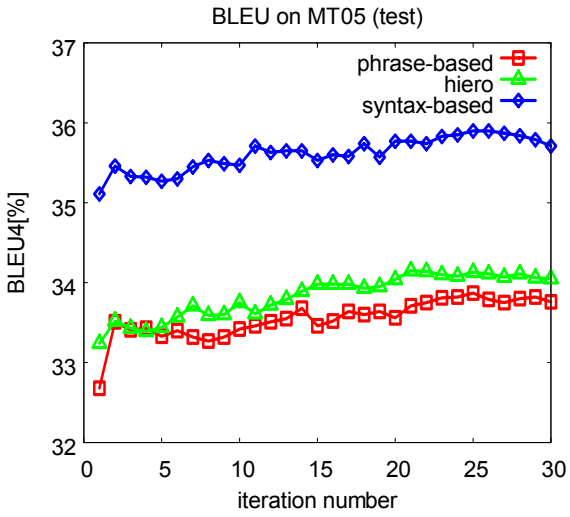

Figure 4: BLEU scores on the test set of MT05

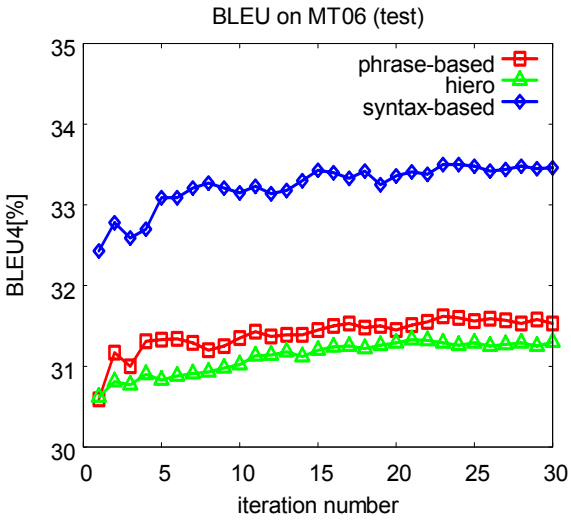

Figure 5: BLEU scores on the test set of MT06

	Phrase-based				Hiero				Syntax-based			
	Dev.	MT04	MT05	MT06	Dev.	MT04	MT05	MT06	Dev.	MT04	MT05	MT06
Baseline	33.21	33.68	32.68	30.59	33.42	34.30	33.24	30.62	35.84	35.71	35.11	32.43
Baseline+600best	33.32	33.93	32.84	30.76	33.48	34.46	33.39	30.75	35.95	35.88	35.23	32.58
Boosting-5Iterations	33.95*	34.32*	33.33*	31.33*	33.73	34.48	33.44	30.83	36.03	35.92	35.27	33.09
Boosting-10Iterations	34.14*	34.68*	33.42*	31.35*	33.75	34.65	33.75*	31.02	36.14	36.39*	35.47	33.15*
Boosting-15Iterations	33.99*	34.78*	33.46*	31.45*	34.03*	34.88*	33.98*	31.20*	36.36*	36.46*	35.53*	33.43*
Boosting-20Iterations	34.09*	35.11*	33.56*	31.45*	34.17*	35.00*	34.04*	31.29*	36.44*	36.79*	35.77*	33.36*
Boosting-30Iterations	34.12*	35.16*	33.76*	31.59*	34.05*	34.99*	34.05*	31.30*	36.52*	36.81*	35.71*	33.46*

Table 1: Summary of the results (BLEU4[%]) on the development and test sets. * = significantly better than baseline ($p < 0.05$).

hieves significant BLEU improvements after 15 iterations, and the highest BLEU scores are generally yielded after 20 iterations.

Also as shown in Table 1, over 0.7 BLEU point gains are obtained on the phrase-based system after 10 iterations. The largest BLEU improvement on the phrase-based system is over 1 BLEU point in most cases. These results reflect that our method is relatively more effective for the phrase-based system than for the other two systems, and thus confirms the fact we observed in Figures 2-5.

We also investigate the impact of *n*-best list size on the performance of baseline systems. For the comparison, we show the performance of the baseline systems with the *n*-best list size of 600 (*Baseline+600best* in Table 1) which equals to the maximum number of translation candidates accessed in the final combination system (combine 30 member systems, i.e. *Boosing-30Iterations*).

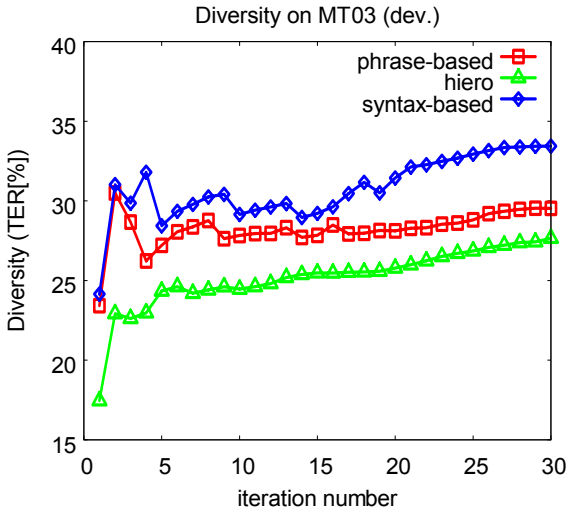

Figure 6: Diversity on the development set

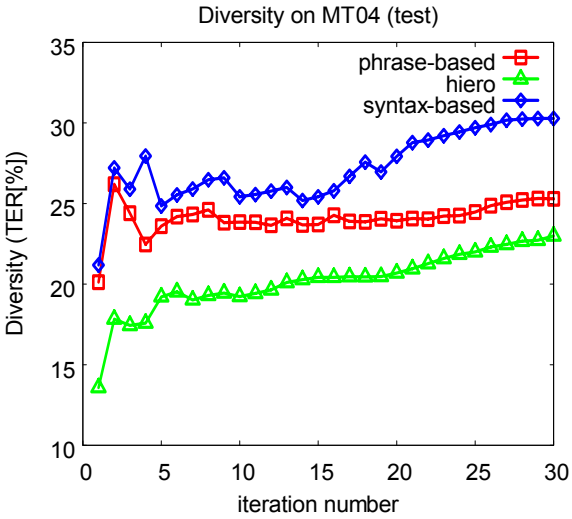

Figure 7: Diversity on the test set of MT04

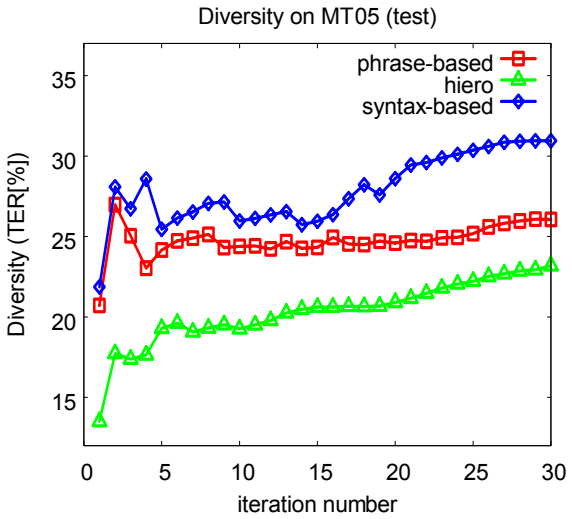

Figure 8: Diversity on the test set of MT05

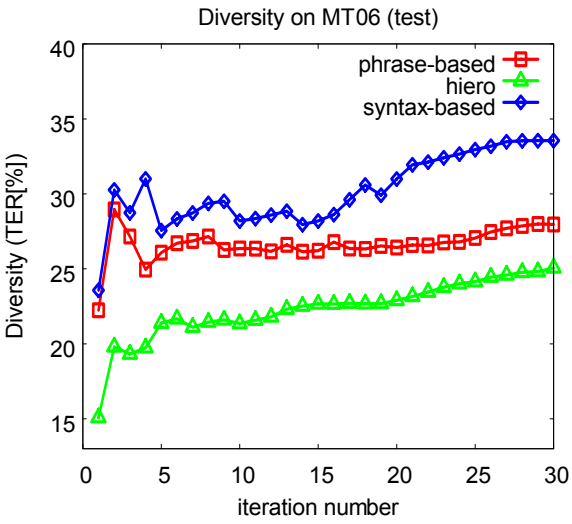

Figure 9: Diversity on the test set of MT06

As shown in Table 1, *Baseline+600best* obtains stable improvements over *Baseline*. It indicates that the access to larger *n*-best lists is helpful to improve the performance of baseline systems. However, the improvements achieved by *Baseline+600best* are modest compared to the improvements achieved by *Boosting-30Iterations*. These results indicate that the SMT systems can benefit more from the diversified outputs of member systems rather than from larger *n*-best lists produced by a single system.

5.4 Diversity among Member Systems

We also study the change of diversity among the outputs of member systems during iterations. The diversity is measured in terms of the Translation Error Rate (TER) metric proposed in (Snover et al., 2006). A higher TER score means that more edit operations are performed if we transform one translation output into another translation output, and thus reflects a larger diversity between the two outputs. In this work, the TER score for a given group of member systems is calculated by averaging the TER scores between the outputs of each pair of member systems in this group.

Figures 6-9 show the curves of diversity on the development and test sets, where the X-axis is the iteration number, and the Y-axis is the diversity. The points at iteration 1 stand for the diversities of baseline systems. In this work, the baseline's diversity is the TER score of the group of baseline candidates that are generated in advance (Section 5.1).

We see that the diversities of all the systems increase during iterations in most cases, though a few drops occur at a few points. It indicates that our method is very effective to generate diversified member systems. In addition, the diversities of baseline systems (iteration 1) are much lower

than those of the systems generated by boosting (iterations 2-30). Together with the results shown in Figures 2-5, it confirms our motivation that the diversified translation outputs can lead to performance improvements over the baseline systems.

Also as shown in Figures 6-9, the diversity of the Hiero system is much lower than that of the phrase-based and syntax-based systems at each individual setting of iteration number. This interesting finding supports the observation that the performance of the Hiero system is relatively more stable than the other two systems as shown in Figures 2-5. The relative lack of diversity in the Hiero system might be due to the *spurious ambiguity* in Hiero derivations which generally results in very few different translations in translation outputs (Chiang, 2007).

5.5 Evaluation of Oracle Translations

In this set of experiments, we evaluate the oracle performance on the *n*-best lists of the baseline systems and the combined systems generated by boosting-based system combination. Our primary goal here is to study the impact of our method on the upper-bound performance.

Table 2 shows the results, where *Baseline+600best* stands for the top-600 translation candidates generated by the baseline systems, and *Boosting-30iterations* stands for the ensemble of 30 member systems' top-20 translation candidates. As expected, the oracle performance of *Boosting-30Iterations* is significantly higher than that of *Baseline+600best*. This result indicates that our method can provide much "better" translation candidates for system combination than enlarging the size of *n*-best list naively. It also gives us a rational explanation for the significant improvements achieved by our method as shown in Section 5.3.

Data Set	Method	Phrase-based	Hiero	Syntax-based
Dev.	Baseline+600best	46.36	46.51	46.92
	Boosting-30Iterations	47.78*	47.44*	48.70*
MT04	Baseline+600best	43.94	44.52	46.88
	Boosting-30Iterations	45.97*	45.47*	49.40*
MT05	Baseline+600best	42.32	42.47	45.21
	Boosting-30Iterations	44.82*	43.44*	47.02*
MT06	Baseline+600best	39.47	39.39	40.52
	Boosting-30Iterations	41.51*	40.10*	41.88*

Table 2: Oracle performance of various systems. * = significantly better than baseline ($p < 0.05$).

6 Related Work

Boosting is a machine learning (ML) method that has been well studied in the ML community (Freund, 1995; Freund and Schapire, 1997; Collins et al., 2002; Rudin et al., 2007), and has been successfully adopted in natural language processing (NLP) applications, such as document classification (Schapire and Singer, 2000) and named entity classification (Collins and Singer, 1999). However, most of the previous work did not study the issue of how to improve a single SMT engine using boosting algorithms. To our knowledge, the only work addressing this issue is (Lagarda and Casacuberta, 2008) in which the boosting algorithm was adopted in phrase-based SMT. However, Lagarda and Casacuberta (2008)'s method calculated errors over the phrases that were chosen by phrase-based systems, and could not be applied to many other SMT systems, such as hierarchical phrase-based systems and syntax-based systems. Differing from Lagarda and Casacuberta's work, we are concerned more with proposing a general framework which can work with most of the current SMT models and empirically demonstrating its effectiveness on various SMT systems.

There are also some other studies on building diverse translation systems from a single translation engine for system combination. The first attempt is (Macherey and Och, 2007). They empirically showed that diverse translation systems could be generated by changing parameters at early-stages of the training procedure. Following Macherey and Och (2007)'s work, Duan et al. (2009) proposed a feature subspace method to build a group of translation systems from various different sub-models of an existing SMT system. However, Duan et al. (2009)'s method relied on the heuristics used in feature sub-space selection. For example, they used the remove-one-feature strategy and varied the order of *n*-gram language model to obtain a satisfactory group of diverse systems. Compared to Duan et al. (2009)'s method, a main advantage of our method is that it can be applied to most of the SMT systems without designing any heuristics to adapt it to the specified systems.

7 Discussion and Future Work

Actually the method presented in this paper is doing something rather similar to Minimum Bayes Risk (MBR) methods. A main difference lies in that the consensus-based combination method here does not model the posterior probability of each hypothesis (i.e. all the hypotheses are assigned an equal posterior probability when we calculate the consensus-based features).

Greater improvements are expected if MBR methods are used and consensus-based combination techniques smooth over noise in the MERT pipeline.

In this work, we use a sentence-level system combination method to generate final translations. It is worth studying other more sophisticated alternatives, such as word-level and phrase-level system combination, to further improve the system performance.

Another issue is how to determine an appropriate number of iterations for boosting-based system combination. It is especially important when our method is applied in the real-world applications. Our empirical study shows that the stable and satisfactory improvements can be achieved after 6-8 iterations, while the largest improvements can be achieved after 20 iterations. In our future work, we will study in-depth principled ways to determine the appropriate number of iterations for boosting-based system combination.

8 Conclusions

We have proposed a boosting-based system combination method to address the issue of building a strong translation system from a group of weak translation systems generated from a single SMT engine. We apply our method to three state-of-the-art SMT systems, and conduct experiments on three NIST Chinese-to-English MT evaluations test sets. The experimental results show that our method is very effective to improve the translation accuracy of the SMT systems.

Acknowledgements

This work was supported in part by the National Science Foundation of China (60873091) and the Fundamental Research Funds for the Central Universities (N090604008). The authors would like to thank the anonymous reviewers for their pertinent comments, Tongran Liu, Chunliang Zhang and Shujie Yao for their valuable suggestions for improving this paper, and Tianning Li and Rushan Chen for developing parts of the baseline systems.

References

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proc. of* ACL 2005, Ann Arbor, Michigan, pages 263-270.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201-228.

David Chiang, Yuval Marton and Philip Resnik. 2008. Online Large-Margin Training of Syntactic and Structural Translation Features. In *Proc. of* EMNLP 2008, Honolulu, pages 224-233.

Michael Collins and Yoram Singer. 1999. Unsupervised Models for Named Entity Classification. In *Proc. of* EMNLP/VLC 1999, pages 100-110.

Michael Collins, Robert Schapire and Yoram Singer. 2002. Logistic Regression, AdaBoost and Bregman Distances. *Machine Learning*, 48(3): 253-285.

Brooke Cowan, Ivona Kučerová and Michael Collins. 2006. A discriminative model for tree-to-tree translation. In *Proc. of* EMNLP 2006, pages 232-241.

Yuan Ding and Martha Palmer. 2005. Machine translation using probabilistic synchronous dependency insertion grammars. In *Proc. of* ACL 2005, Ann Arbor, Michigan, pages 541-548.

Nan Duan, Mu Li, Tong Xiao and Ming Zhou. 2009. The Feature Subspace Method for SMT System Combination. In *Proc. of* EMNLP 2009, pages 1096-1104.

Jason Eisner. 2003. Learning non-isomorphic tree mappings for machine translation. In *Proc. of* ACL 2003, pages 205-208.

Yoav Freund. 1995. Boosting a weak learning algorithm by majority. *Information and Computation*, 121(2): 256-285.

Yoav Freund and Robert Schapire. 1997. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119-139.

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang and Ignacio Thayer. 2006. Scalable inferences and training of context-rich syntax translation models. In *Proc. of* ACL 2006, Sydney, Australia, pages 961-968.

Michel Galley and Christopher D. Manning. 2008. A Simple and Effective Hierarchical Phrase Reordering Model. In *Proc. of* EMNLP 2008, Hawaii, pages 848-856.

Almut Silja Hildebrand and Stephan Vogel. 2008. Combination of machine translation systems via hypothesis selection from combined n-best lists. In *Proc. of* the 8th AMTA conference, pages 254-261.

Liang Huang and David Chiang. 2007. Forest rescoring: Faster decoding with integrated language models. In *Proc. of* ACL 2007, Prague, Czech Republic, pages 144-151.

Philipp Koehn, Franz Och and Daniel Marcu. 2003. Statistical Phrase-Based Translation. In *Proc. of* HLT-NAACL 2003, Edmonton, USA, pages 48-54.

Philipp Koehn. 2004. Statistical Significance Tests for Machine Translation Evaluation. In *Proc. of* EMNLP 2004, Barcelona, Spain, pages 388-395.

Antonio Lagarda and Francisco Casacuberta. 2008. Applying Boosting to Statistical Machine Translation. In *Proc. of* the 12th EAMT conference, pages 88-96.

Mu Li, Nan Duan, Dongdong Zhang, Chi-Ho Li and Ming Zhou. 2009. Collaborative Decoding: Partial Hypothesis Re-Ranking Using Translation Consensus between Decoders. In *Proc. of* ACL-IJCNLP 2009, Singapore, pages 585-592.

Percy Liang, Alexandre Bouchard-Côté, Dan Klein and Ben Taskar. 2006. An end-to-end discriminative approach to machine translation. In *Proc. of* COLING/ACL 2006, pages 104-111.

Yang Liu, Qun Liu and Shouxun Lin. 2006. Tree-to-String Alignment Template for Statistical Machine Translation. In *Proc. of* ACL 2006, pages 609-616.

Wolfgang Macherey and Franz Och. 2007. An Empirical Study on Computing Consensus Translations from Multiple Machine Translation Systems. In *Proc. of* EMNLP 2007, pages 986-995.

Daniel Marcu, Wei Wang, Abdessamad Echihabi and Kevin Knight. 2006. SPMT: Statistical machine translation with syntactified target language phrases. In *Proc. of* EMNLP 2006, Sydney, Australia, pages 44-52.

Evgeny Matusov, Nicola Ueffing and Hermann Ney. 2006. Computing consensus translation from multiple machine translation systems using enhanced hypotheses alignment. In *Proc. of* EACL 2006, pages 33-40.

Franz Och and Hermann Ney. 2002. Discriminative Training and Maximum Entropy Models for Statistical Machine Translation. In *Proc. of* ACL 2002, Philadelphia, pages 295-302.

Franz Och. 2003. Minimum Error Rate Training in Statistical Machine Translation. In *Proc. of* ACL 2003, Japan, pages 160-167.

Antti-Veikko Rosti, Spyros Matsoukas and Richard Schwartz. 2007. Improved Word-Level System Combination for Machine Translation. In *Proc. of* ACL 2007, pages 312-319.

Cynthia Rudin, Robert Schapire and Ingrid Daubechies. 2007. Analysis of boosting algorithms using the smooth margin function. *The Annals of Statistics*, 35(6): 2723-2768.

Robert Schapire and Yoram Singer. 2000. BoosTexter: A boosting-based system for text categorization. *Machine Learning*, 39(2/3):135-168.

Robert Schapire. The boosting approach to machine learning: an overview. 2001. In *Proc. of* MSRI Workshop on Nonlinear Estimation and Classification, Berkeley, CA, USA, pages 1-23.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla and John Makhoul. 2006. A Study of Translation Edit Rate with Targeted Human Annotation. In *Proc. of* the 7th AMTA conference, pages 223-231.

Tong Xiao, Mu Li, Dongdong Zhang, Jingbo Zhu and Ming Zhou. 2009. Better Synchronous Binarization for Machine Translation. In *Proc. of* EMNLP 2009, Singapore, pages 362-370.

Deyi Xiong, Qun Liu and Shouxun Lin. 2006. Maximum Entropy Based Phrase Reordering Model for Statistical Machine Translation. In *Proc. of* ACL 2006, Sydney, pages 521-528.

Hao Zhang, Liang Huang, Daniel Gildea and Kevin Knight. 2006. Synchronous Binarization for Machine Translation. In *Proc. of* HLT-NAACL 2006, New York, USA, pages 256- 263.