Adaptive Development Data Selection for Log-linear Model in Statistical Machine Translation

Mu Li
Microsoft Research Asia
muli@microsoft.com

Yinggong Zhao[*]
Nanjing University
zhaoyg@nlp.nju.edu.cn

Dongdong Zhang
Microsoft Research Asia
dozhang@microsoft.com

Ming Zhou
Microsoft Research Asia
mingzhou@microsoft.com

Abstract

This paper addresses the problem of dynamic model parameter selection for log-linear model based statistical machine translation (SMT) systems. In this work, we propose a principled method for this task by transforming it to a test data dependent development set selection problem. We present two algorithms for automatic development set construction, and evaluated our method on several NIST data sets for the Chinese-English translation task. Experimental results show that our method can effectively adapt log-linear model parameters to different test data, and consistently achieves good translation performance compared with conventional methods that use a fixed model parameter setting across different data sets.

1 Introduction

In recent years, log-linear model (Och and Ney, 2002) has been a mainstream method to formulate statistical models for machine translation. Using this formulation, various kinds of relevant properties and data statistics used in the translation process, either on the monolingual-side or on the bilingual-side, are encoded and used as real-valued *feature functions*, thus it provides an effective mathematical framework to accommodate a large variety of SMT formalisms with different computational linguistic motivations.

Formally, in a log-linear SMT model, given a source sentence f, we are to find a translation e^* with largest posterior probability among all possible translations:

$$e^* = \operatorname*{argmax}_e \Pr(e|f)$$

and the posterior probability distribution $\Pr(e|f)$ is directly approximated by a log-linear formulation:

$$\begin{aligned}\Pr(e|f) &= p_{\boldsymbol{\lambda}}(e|f)\\ &= \frac{\exp(\sum_{m=1}^{M}\lambda_m h_m(e,f))}{\sum_{e'}\exp(\sum_{m=1}^{M}\lambda_m h_m(e',f))} \qquad (1)\end{aligned}$$

in which h_m's are feature functions and $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_M)$ are model parameters (*feature weights*).

For a successful practical log-linear SMT model, it is usually a combined result of the several efforts:

- Construction of well-motivated SMT models
- Accurate estimation of feature functions
- Appropriate scaling of log-linear model features (feature weight tuning).

In this paper, we focus on the last mentioned issue – parameter tuning for log-linear model. In general, log-linear model parameters are optimized on a held-out development data set. Using this method, similarly to many machine learning tasks, the model parameters are solely tuned based on the development data, and the optimality of obtained model on unseen test data relies on the assumption that both development and test data observe identical probabilistic distribution,

[*]This work was done while the author was visiting Microsoft Research Asia.

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 662–670,
Beijing, August 2010

which often does not hold for real-world data. The goal of this paper is to investigate novel methods for test data dependent model parameter selection. We begin with discussing the principle of parameter learning for log-linear SMT models, and explain the rationale of task transformation from parameter selection to development data selection. We describe two algorithms for automatic development set construction, and evaluated our method on several NIST MT evaluation data sets. Experimental results show that our method can effectively adapt log-linear model parameters to different test data and achieves consistent good translation performance compared with conventional methods that use a group of fixed model parameters across different data sets.

2 Model Learning for SMT with Log-linear Models

Model learning refers to the task to estimate a group of suitable log-linear model parameters $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_M)$ for use in Equation 1, which is often formulated as an optimization problem that finds the parameters maximizing certain *goodness* of the translations generated by the learnt model on a development corpus D. The goodness can be measured with either the translations' likelihood or specific machine translation evaluation metrics such as TER or BLEU.

More specifically, let e^* be the most probable translation of D with respect to model parameters $\boldsymbol{\lambda}$, and $E(e^*, \boldsymbol{\lambda}, D)$ be a score function indicating the goodness of translation e^*, then a parameter estimation algorithm will try to find the $\boldsymbol{\lambda}$ which satisfies:

$$\boldsymbol{\lambda}^* = \operatorname*{argmax}_{\boldsymbol{\lambda}} E(e^*, \boldsymbol{\lambda}, D) \tag{2}$$

Note when the goodness scoring function $E(\cdot)$ is specified, the parameter learning criterion in Equation 2 indicates that the derivation of model parameters $\boldsymbol{\lambda}^*$ only depends on development data D, and does not require any knowledge of test data T. The underlying rationale for this rule is that if the test data T observes the same distribution as D, $\boldsymbol{\lambda}^*$ will be optimal for both of them.

On the other side, however, when there are mismatches between development and test data, the translation performance on test data will be suboptimal, which is very common for real-world data. Due to the difference between data sets, generally there is no such $\boldsymbol{\lambda}^*$ that is optimal for multiple data sets at the same time. Table 1 shows some empirical evidences when two data sets are mutually used as development and test data. In this setting, we used a hierarchical phrase based decoder and 2 years' evaluation data of NIST Chinese-to-English machine translation task (for the year 2008 only the newswire subset was used because we want to limit both data sets within the same domain to show that data mismatch also exists even if there is no domain difference), and report results using BLEU scores. Model parameters were tuned using the MERT algorithm (Och, 2003) optimized for BLEU metric.

Dev data	MT05	MT08-nw
MT05	**0.402**	0.306
MT08-nw	0.372	**0.343**

Table 1: Translation performance of cross development/test on two NIST evaluation data sets.

In our work, we present a solution to this problem by using test data dependent model parameters for test data translation. As discussed above, since model parameters are solely determined by development data D, selection of log-linear model parameters is basically equivalent to selecting a set of development data D.

However, automatic development data selection in current SMT research remains a relatively open issue. Manual selection based on human experience and observation is still a common practice.

3 Adaptive Model Parameter Selection

An important heuristic behind manual development data selection is to use the dataset which is as similar to test set as possible in order to work around the data mismatch problem to maximal extent. There are also empirical evidences supporting this heuristics. For instance, it is generally perceived that data set MT03 is more similar to MT05, while MT06-nw is closer to MT08-nw. Table 2 shows experimental results using model parameters induced from MT03 and MT06-nw as development sets with the same settings as in Table 1. As expected, MT06-nw is far more suitable than MT03 as the development data for MT08-nw; yet for test set MT05, the situation is just the opposite.

Dev data	MT05	MT08-nw
MT03	**0.397**	0.306
MT06-nw	0.381	**0.337**

Table 2: Translation performance on different test sets of using different development sets.

In this work, this heuristic is further exploited for automatic development data selection when there is no prior knowledge of the test data available. In the following discussion, we assume the availability of a set of candidate source sentences together with translation references that are qualified for the log-linear model parameter learning task. Let D_F be the full candidate set, given a test set T, the task of selecting a set of development data which can optimize the translation quality on T can be transformed to searching for a suitable subset of D_F which is most similar to T:

$$D^* = \underset{D \subseteq D_F}{\operatorname{argmax}} \operatorname{Sim}(D, T)$$

To achieve this goal, we need to address the following key issues:

- How to define and compute $\operatorname{Sim}(D, T)$, the similarity between different data sets;
- How to extract development data sets from a full candidate set for unseen test data.

3.1 Dataset Similarity

Computing document similarity is a classical task in many research areas such as information retrieval and document classification. However, typical methods for computing document similarity may not be suitable for our purpose. The reasons are two-fold:

1. The sizes of both development and test data are small in usual circumstances, and using similarity measures such as cosine or dice coefficient based on term vectors will suffer from severe data sparseness problems. As a result, the obtained similarity measure will not be statistically reliable.
2. More importantly, what we care about here is not the surface string similarity. Instead, we need a method to measure how similar two data sets are from the view of a log-linear SMT model.

Next we start with discussing the similarity between sentences. Given a source sentence f, we denote its possible translation space with $\mathcal{H}(f)$. In a log-linear SMT model, every translation $e \in \mathcal{H}(f)$ is essentially a feature vector $\boldsymbol{h}(e) = (h_1, \ldots, h_M)$. Accordingly, the similarity between two sentences f_1 and f_2 should be defined on the feature space of the model in use. Let $\boldsymbol{V}(f) = \{\boldsymbol{h}(e) : e \in \mathcal{H}(f)\}$ be the set of feature vectors for all translations in $\mathcal{H}(f)$, we have

$$\operatorname{Sim}(f_1, f_2) = \operatorname{Sim}\Big(\boldsymbol{V}(f_1), \boldsymbol{V}(f_2)\Big) \quad (3)$$

Because it is not practical to compute Equation 3 directly by enumerating all translations in $\mathcal{H}(f_1)$ and $\mathcal{H}(f_2)$ due to the huge search space in SMT tasks, we need to resort to some approximations. A viable solution to this is that if we can use a single feature vector $\tilde{\boldsymbol{h}}(f)$ to represent $\boldsymbol{V}(f)$, then Equation 3 can be simply computed using existing vector similarity measures.

One reasonable method to derive $\tilde{\boldsymbol{h}}(f)$ is to use a feature vector based on the *average* principle – each dimension of the vector is set to the expectation of its corresponding feature value over all translations:

$$\tilde{\boldsymbol{h}}(f) = \sum_{e \in \mathcal{H}(f)} P(e|f)\boldsymbol{h}(e) \quad (4)$$

An alternative and much simpler way to compute $\tilde{\boldsymbol{h}}(f)$ is to employ the *max* principle in which we just use the feature vector of the best translation in $\mathcal{H}(f)$:

$$\tilde{\boldsymbol{h}}(f) = \boldsymbol{h}(e^*) \quad (5)$$

where $e^* = \operatorname{argmax}_e P(e|f)$.

Note that in both Equation 4 and Equation 5 we make use of e's posterior probability $P(e|f)$.

Since the true distribution is unknown, a prelearnt model $\mathcal{M}$ has to be used to assign approximate probabilities to translations, which indicates that the obtained similarity depends on a specific model. As a convention, we use $\mathrm{Sim}_{\mathcal{M}}(f_1, f_2)$ to denote the similarity between f_1 and f_2 based on $\mathcal{M}$, and call $\mathcal{M}$ the reference model of the computed similarity. To avoid unexpected bias caused by a single reference model, multiple reference models can be simultaneously used, and the similarity is defined to be the maximum of all model-dependent similarity values:

$$\mathrm{Sim}(f_1, f_2) = \max_{\mathcal{M}} \mathrm{Sim}_{\mathcal{M}}(f_1, f_2) \tag{6}$$

where $\mathcal{M}$ belongs to $\{\mathcal{M}_1, \dots, \mathcal{M}_n\}$, which is the set of reference models under consideration.

To generalize this method to data set level, we compute the vector $\tilde{\boldsymbol{h}}(S)$ for a data set $S = (f_1, \dots, f_{|S|})$ as follows:

$$\tilde{\boldsymbol{h}}(S) = \sum_{i=1}^{|S|} \tilde{\boldsymbol{h}}(f_i) \tag{7}$$

3.2 Development Sets Pre-construction

In the following, we sketch a method for automatically building a set of development data based on the full candidate set D_F before seeing any test data.

Theoretically, a subset of D_F containing randomly sampled sentences from D_F will not meet our requirement well because it is very probable that it will observe a distribution similar to D_F. What we expect is that the pre-built development sets can approximate as many as possible typical data distributions that can be estimated from subsets of D_F. Our solution is based on the assumption that D_F can be depicted by some mixture models, hence we can use classical clustering methods such as k-means to partition D_F into subsets with different distributions.

Let S_F be the set of extracted development data from D_F. The construction of S_{D_F} proceeds as following:

1. Train a log-linear model $\mathcal{M}_F$ using D_F as development data;
2. Compute a feature vector $\tilde{\boldsymbol{h}}(d)$[1] for each sentence $d \in D_F$ using $\mathcal{M}_F$ as reference model;
3. Cluster sentences in D_F using $\tilde{\boldsymbol{h}}(d)/|d|$ as feature vectors;
4. Add obtained sentence clusters to S_{D_F} as candidate development sets.

In the third step, since the feature vector $\tilde{\boldsymbol{h}}(d)$ is defined at sentence level, it is averaged by the number of words in d so that it is irrelevant to the length of a sentence. Considering the outputs of unsupervised data clustering methods are usually sensitive to initial conditions, we include in S_{D_F} sentence clusters based on different initialization configurations to remove related random effects. An initialization configuration for sentence clustering in our work includes starting point for each cluster and total number of clusters. In fact, the inclusion of more sentence clusters increases the diversity of the resulted S_{D_F} as well.

At decoding time, when a test set T is presented, we compute the similarity between T and each development set $D \in S_{D_F}$, and choose the one with largest similarity score as the development set for T:

$$D^* = \operatorname*{argmax}_{D \in S_{D_F}} \mathrm{Sim}(T, D) \tag{8}$$

When a single reference model is used to compute $\mathrm{Sim}(T, D)$, $\mathcal{M}_F$ is a natural choice. In the multi-model setting as shown in Equation 6, models learnt from the development sets in S_{D_F} can serve this purpose.

Note in this method model learning is not required for every new test set because the model parameters for each development set in S_{D_F} can also be pre-learnt and ready to be used for decoding.

3.3 Dynamic Development Set Construction

In the previous method, test data T is only involved in the process of choosing a development set from a list of candidates but not in process of development set construction. Next we present a

[1] Throughout this paper, a development sentence d generally refers to the source part of it if there is no extra explanation.

method for building a development set on demand based on test data T.

Let $D_F = (d_1, \ldots, d_n)$ be the data set containing all candidate sentences for development data selection. The method is iterative process in which development data and learnt model are alternatively updated. Detailed steps are illustrated as follows:

1. Let $i = 0$, $D_0 = D_F$;
2. Train a model $\mathcal{M}_i$ based on D_i;
3. For each $d_k \in D_F$, compute the similarity score $\mathrm{Sim}_{\mathcal{M}_i}(T, d_k)$ between T and d_k based on model $\mathcal{M}_i$;
4. Select top n candidate sentences with highest similarity scores from D_F to form D_{i+1};
5. Repeat step 2 to step 4 until the similarity between T and latest selected development data converges (the increase in similarity measure is less than a specified threshold compared to last round) or the specified iteration limit is reached.

In step 4, D_{i+1} is greedily extracted from D_F, and there is no guarantee that $\mathrm{Sim}_{\mathcal{M}_i}(T, D_{i+1})$ will increase or decrease after a new sentence is added to D_{i+1}. Thereby the number of selected sentences n needs to be empirically determined. If n is too small, neither the selected data nor the learnt model parameters will be statistically reliable; while if n is too large, we may have to include some sentences that are not suitable for test data in the development data, and miss the opportunity to extract the most desirable development set.

One drawback of this method is the relatively high computational cost because it requires multiple parameter training passes when any test set is presented to the system for translation.

4 Experiments

4.1 Data

Experiments were conducted on the data sets used for NIST Chinese-English machine translation evaluation tasks. MT03 and MT06 data sets, which contain 919 and 1,664 sentences respectively, were used for development data in various settings. MT04, MT05 and MT08 data sets were used for test purpose. In some settings, we also used a test set MT0x, which containing 1,000 sentences randomly sampled from the above 3 data sets. All the translation performance results were measured in terms of case-insensitive BLEU scores.

For all experiments, all parallel corpora available to the constrained track of NIST 2008 Chinese-English MT evaluation task were used for translation model training, which consist of around 5.1M bilingual sentence pairs. GIZA++ was used for word alignment in both directions, which was further refined with the intersec-diag-grow heuristics.

We used a 5-gram language model which was trained from the Xinhua portion of English Gigaword corpus version 3.0 from LDC and the English part of parallel corpora.

4.2 Machine Translation System

We used an in-house implementation of the hierarchical phrase-based decoder as described in Chiang (2005). In addtion to the standard features used in Chiang (2005), we also used a lexicon feature indicating how many word paris in the translation found in a conventional Chinese-English lexicon. Phrasal rules were extracted from all the parallel data, but hierarchical rules were only extracted from the FBIS part of the parallel data which contains around 128,000 sentence pairs. For all the development data, feature weights of the decoder were tuned using the MERT algorithm (Och, 2003).

4.3 Results of Development Data Pre-construction

In the following we first present some overall results using the method of development data preconstruction, then dive into more detailed settings of the experiments.

Table 3 shows the results using 3 different data sets for log-linear model parameter tuning. Elements in the first column indicate the data sets used for parameter tuning, and other columns contain evaluation results on different test sets. In the

Tuning set	MT04	MT05	MT08	MT0x
MT03	0.399 / 0.392	0.395 / 0.390	0.241 / 0.258	0.319 / 0.322
MT06	0.381 / 0.388	0.382 / 0.391	0.275 / 0.283	0.343 / 0.342
MT03+MT06	0.391 / 0.401	0.392 / 0.397	0.265 / 0.281	0.336 / 0.345
Oracle cluster	0.401	0.398	0.293	0.345
Self-training	0.406	0.402	0.298	0.351

Table 3: Translation performance using different methods and data sets for parameter tuning.

third row of the table, MT03+MT06 means combining the data sets of MT03 and MT06 together to form a larger tuning set. The first number in each cell denotes the BLEU score using the tuning set as standard development set D, and the second for using the tuning set as a candidate set D_F.

For all experiment settings in the table, we used cosine value between feature vectors to measure similarity between data sets, and feature vectors were computed according to Equation 5 and Equation 7 using a reference model which is trained on the corresponding candidate set D_F as development set.[2] We adopted the k-means algorithm for data clustering with the number of clusters iterating from 2 to 5. In each iteration, we ran 4 passes of clustering using different initial values. Therefore, in total there are 56 sentence clusters generated in each S_{D_F}.[3]

From the table it can be seen that given the same set of sentences (MT03, MT06 and MT03+MT06), when they are used as the candidate set D_F for the development set preconstruction method, the translation performance is generally better than when they are just used as development sets as a whole. Using MT03 data set as D_F is an exception: there is slight performance drop on test sets MT04 and MT05, but it also helps reduce the performance *see-saw* problem on different test sets as shown in Table 1. Meanwhile, in the other two settings of D_F, we observed significant BLEU score increase on all test sets but MT0x (on which the performance almost kept unchanged). In addition, the fact that using MT03+MT06 as D_F achieves best (or almost best) performance on all test sets implies that it should be a better choice to include as diverse data as possible in D_F.

We also appended two oracle BLEU numbers for each test set in Table 3 for reference. One is denoted with *oracle cluster*, which is the highest possible BLEU that can be achieved on the test set when the development set must be chosen from the sentence clusters in $S_{MT03+MT06}$. The other is labeled as *self-training*, which is the BLEU score that can be obtained when the test data itself is used as development data. This number can serve as actual performance upper bound on the test set.

Next we investigated the impact of using different ways to compute feature vectors presented in Section 3.1. We re-ran some previous experiments on test sets MT04, MT05 and MT08 using MT03+MT06 as D_F. Most settings were kept unchanged except that the feature vector of each sentence was computed according to Equation 4. A 20-best translation list was used to approximate $\mathcal{H}(f)$. The results are shown in Table 4.

Test set	average	max
MT04	0.397	0.401
MT05	0.393	0.397
MT08	0.286	0.281

Table 4: Translation performance when using averaged feature values for similarity computation.

The numbers in the second column are based on Equation 4. Numbers based on Equation 5 are also listed in the third column for comparison. In all the experiment settings we did not observe consistent or significant advantage when using Equation 4 over using Equation 5. Since Equation 5

[2] For example, in all the experiments in the row of MT03 as D_F, we use the same reference model trained with MT03 as development set.

[3] Sometimes some clusters are empty or contain too few sentences, so the actual number may be smaller.

is much simpler, it is a good decision to use it in practice. So did we conduct all following experiments based on Equation 5.

We are also interested in the correlation between two measures: the similarity between development and test data and the actual translation performance on test data.

First we would like to echo the motivating experiment presented in Section 3. Table 5 shows the similarity between the data sets used in the experiment with $\mathcal{M}_{\mathrm{MT03+MT06}}$ as reference model. Obviously the results in Table 2 and Table 5 fit each other very well.

Dev data	MT05	MT08-nw
MT03	**0.99988**	0.99012
MT06-nw	0.99004	**0.99728**

Table 5: Similarity between NIST data sets.

Figure 1 shows the results of a set of more comprehensive experiments on MT05 data set concerning the similarity between development and test sets.

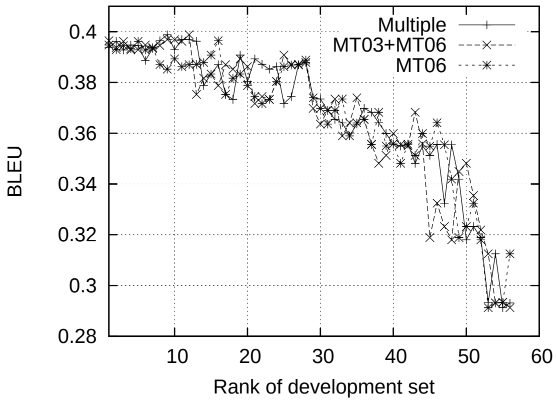

Figure 1: Correlation between similarity and BLEU on MT05 data set

In the figure, every data line shows how BLEU score changes when different pre-built development set in $S_{MT03+MT06}$ is used for model learning. The data points in each line are sorted by the rank of similarity between the development set in use and the MT05 data set. We also compared results based on 3 reference model settings. In the first one (multiple), the similarity was computed using Equation 6, and the reference model set contains all models learnt from the development sets in $S_{MT03+MT06}$. The other two settings use reference models learnt from MT06 and MT03+MT06 data sets respectively.

We can observe from the figure that the correlation between BLEU scores and data set similarity can only be identified on macro scales for all the three similarity settings. Although using data similarity may not be able to select the perfect development data set from S_{D_F}, by picking a development set with highest similarity score, we can usually (almost always) get good enough BLEU scores in our experiments.

4.4 Results of Development Data Dynamic Generation

We ran two sets of experiments for the method of development data dynamic construction.

The first one was designed to investigate how the size of extracted development data affects the translation performance. Using MT05 and MT08 as test sets and MT03+MT06 as D_F, we ran experiments for the algorithm presented in Section 3.3 with $n = 200$ to $n = 1,000$. In this experiment we did not observe significant enough changes in BLEU scores – the difference between the highest and lowest numbers is generally less than 0.005.

The second one aimed at examining how BLEU numbers changes when the extracted development data were iteratively updated. Figure 2 shows one set of results on test sets MT05 and MT08 using MT03+MT06 data set as D_F and n set to 400.

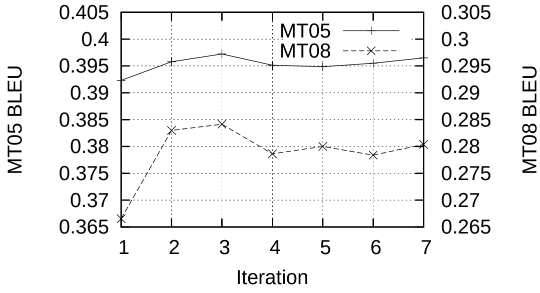

Figure 2: BLEU score as function of iteration in dynamic development data extraction.

The similarity usually converged after 2 to 3 iterations, which is consistent with trend of BLEU scores on test sets. However, in all our experimental settings, we did not observe any results significantly better than using the development set pre-construction method.

5 Discussions

Some of the previous work related to building adaptive SMT systems were discussed in the domain adaptation context, in which one fundamental idea is to estimate a more suitable domain-specific translation model or language model. When the target domain is already known, adding a small amount of domain data (both monolingual and bilingual) to the existing training corpora has been shown to be very effective in practice. But model adaptation is required in more scenarios other than explicitly defined domains. As shown by the results in Table 2, even for the data from the same domain, distribution mismatch can also be a problem.

There are also considerable efforts made to deal with the unknown distribution of text to be translated, and the research topics were still focused on translation and language model adaptation. Typical methods used in this direction include dynamic data selection (Lü et al., 2007; Zhao et al., 2004; Hildebrand et al., 1995) and data weighting (Foster and Kuhn, 2007; Matsoukas et al., 2009). All the mentioned methods use information retrieval techniques to identify relevant training data from the entire training corpora.

Our work presented here also makes no assumption about the distribution of test data, but it differs from the previous methods significantly from a log-linear model's perspective. Adjusting translation and language models based on test data can be viewed as *adaptation of feature values*, while our method is essentially *adaptation of feature weights*. This difference makes these two kinds of methods complementary to each other — it is possible to make further improvement by using both of them in one task.

To our knowledge, there is no dedicated discussion on principled methods to perform development data selection in previous research. In Lü et al. (2007), log-linear model parameters can also be adjusted at decoding time. But in their approach, the adjustment was based on heuristic rules and re-weighted training data distribution. In addition, compared with training data selection, the computational cost of development data selection is much smaller.

From machine learning perspective, both proposed methods can be viewed as certain form of transductive learning applied to the SMT task (Ueffing et al., 2007). But our methods do not rely on surface similarities between training and training/development sentences, and development/test sentences are not used to re-train SMT sub-models.

6 Conclusions and Future Work

In this paper, we addressed the data mismatch issue between training and decoding time of log-linear SMT models, and presented principled methods for dynamically inferring test data dependent model parameters with development set selection. We describe two algorithms for this task, development set pre-construction and dynamic construction, and evaluated our method on the NIST data sets for the Chinese-English translation task. Experimental results show that our methods are capable of consistently achieving good translation performance on multiple test sets with different data distributions without manual tweaking of log-linear model parameters. Though theoretically using the dynamic construction method could bring better results, the pre-construction method performs comparably well in our experimental settings. Considering the fact that the pre-consruction method is computationally cheaper, it should be a better choice in practice.

In the future, we are interested in two directions. One is to explore the possibility to perform data clustering on test set as well and choosing suitable model parameters for each cluster separately. The other involves dynamic SMT model selection – for example, some parts of the test data fit the phrase-based model better while other parts can be better translated using a syntax-based model.

References

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proc. of the 43th Annual Meeting of the Association for Computational Linguistic (ACL).* Ann Arbor, Michigan.

George Foster and Roland Kuhn. 2007. Mixture-Model Adaptation for SMT. In *Proc. of the Second ACL Workshop on Statistical Machine Translation..* Prague, Czech Republic.

Michel Galley, Mark Hopkins, Kevin Knight and Daniel Marcu. 2004. What's in a translation rule? In *Proc. of the Human Language Technology Conf. (HLT-NAACL).* Boston, Massachusetts.

Almut Hildebrand, Matthias Eck, Stephan Vogel, and Alex Waibel. 1995. Adaptation of the Translation Model for Statistical Machine translation Based on Information Retrieval. In *Proc. of EAMT*. Budapest, Hungary.

Philipp Koehn, Franz Och and Daniel Marcu. 2003. Statistical Phrase-Based Translation. In *Proc. of the Human Language Technology Conf. (HLT-NAACL).* Edmonton, Canada.

Yang Liu, Qun Liu, and Shouxun Lin. 2007. Tree-to-string alignment template for statistical machine translation. In *Proc. of the 45th Annual Meeting of the Association for Computational Linguistic (ACL).* Prague, Czech Republic.

Yajuan Lü, Jin Huang and Qun Liu. 2007. Improving Statistical Machine Translation Performance by Training Data Selection and Optimization. In *Proc. of the Conference on Empirical Methods in Natural Language Processing*. Prague, Czech Republic.

Spyros Matsoukas, Antti-Veikko I. Rosti and Bing Zhang. 2009. Discriminative Corpus Weight Estimation for Machine Translation. In *Proc. of the Conference on Empirical Methods in Natural Language Processing*. Singapore.

Franz Och and Hermann Ney. 2002. Discriminative Training and Maximum Entropy Models for Statistical Machine Translation. In *Proc. of the 40th Annual Meeting of the Association for Computational Linguistic (ACL).* Philadelphia, PA.

Franz Och. 2003. Minimum Error Rate Training in Statistical Machine Translation. In *Proc. of the 41th Annual Meeting of the Association for Computational Linguistic (ACL).* Sapporo, Japan.

Nicola Ueffing, Gholamreza Haffari and Anoop Sarkar. 2007. Transductive learning for statistical machine translation. In *Proc. of the Annual Meeting of the Association for Computational Linguistics*. Prague, Czech Republic.

Bing Zhao, Matthias Eck, and Stephan Vogel. 2004. Language Model Adaptation for Statistical Machine Translation with Structured Query Models. In *Proc. of COLING*. Geneva, Switzerland.