A Unified SMT Framework Combining MIRA and MERT

Shujie Liu[†], **Chi-Ho Li**[‡] and **Ming Zhou**[‡]
[†]School of Computer Science and Technology
Harbin Institute of Technology, Harbin, China
shujieliu@mtlab.hit.edu.cn
[‡]Microsoft Research Asia, Beijing, China
{chl, mingzhou}@microsoft.com

Abstract

Translation sub-model is one of the most important components in statistical machine translation, but the conventional approach suffers from two major problems. Firstly, translation sub-model is not optimized with respect to any of automatic evaluation metrics of SMT (such as BLEU). The second problem is over-fitting to training data. This paper presents a new unified framework, by adding a scalable translation sub-model into the conventional framework. The sub-model is optimized with the same criterion as the translation output is evaluated (BLEU), and trained using margin infused relaxed algorithm (MIRA) to handle over-fitting. Under our new framework, MIRA and minimum error rate training (MERT) are unified into an interactive training process. Our approach has not only shown to improve performance over a state-of-the-art baseline, but also generalize well in-domain training data to out-of-domain test data.

1 Introduction

The conventional approach to statistical machine translation (SMT) adopts a log-linear framework to incorporate various features. While there have been a few works in improving the standard minimum error rate training (MERT) for optimizing the weights of the features, most SMT researchers focus on some specific feature used in the log-linear framework. Note that a feature can be a complicated model in itself, such as translation sub-model, distortion sub-model, and language sub-model. The conventional approach trains each sub-model separately, and then integrates them in the log-linear framework. The entire framework is optimized with respect to an evaluation metric, e.g. BLEU, only in the step of feature weight tuning. The training of each model, be it generative or discriminative, has nothing to do with the evaluation metric.

Among the features, translation sub-model plays a key role as it measures the faithfulness of a translation candidate to the source input sentence. Translation sub-model is usually represented as a huge table of their pairs, each of which is a phrase pair, hierarchical phrase pair, or even more complicated structure of synchronous grammar. The translation pairs are usually acquired from word alignment matrices using heuristics. The probabilities of the translation pairs are usually assigned by maximum likelihood estimation (MLE). That is, conventional translation modeling never takes translation evaluation metric into consideration. Moreover, this method of translation modeling suffers from over-fitting, which is critical especially when test data is not similar to training data.

In this paper, we propose a new unified framework to add a discriminative translation sub-model into the conventional linear framework, and the sub-model is optimized with the same criterion as the translation output is evaluated (BLEU in our case). Similar to Blunsom et al. (2009), each translation pair is a feature in itself, and the training method can affect the pairs directly so as to handle over-fitting. Unlike any previous approach(Liang et al., 2006; Arun and Koehn, 2007; Chiang et al., 2008), in which the weights of translation pair features and those features of sub-models are tuned in the same process, we propose a new scalable sub-model which integrates all the translation pairs, and then combine the new sub-model with other sub-models into the conventional framework. The

over-fitting problem of the new scalable sub-model is well tackled by MIRA. The scalable training (MIRA) of the new sub-model and the standard training (MERT) of conventional framework are unified into an interactive training process.

In the following, the previous approaches to translation modeling are reviewed in Section 2. Then we will elaborate the unified translation modeling, the unified training framework and the details of the scalable training methods in Sections 3, 4, and 5 respectively. Experiment result and analysis are given in Section 6.

2 Translation Modeling

The task of translation modeling can be defined as, given a bilingual training corpus, the search of the optimal set of translation pairs with two goals:

1) Explanatory capacity: i.e. the training data can be fully represented by the translation pairs.
2) Generalization capacity: i.e. the translation pairs can also predict the correct translation given unseen source input. In other words, the translation pairs can avoid over-fitting.

The conventional approach to translation modeling comprises three steps (Och, 2003): Firstly, the sentence pairs in training corpus are aligned at word level. Secondly, translation pairs are extracted using a heuristic method. Lastly, MLE is used to compute translation probabilities. There are a few shortcomings of this method: 1) Inconsistent format of translation knowledge: word alignment in training vs. translation pairs (phrase pairs) in decoding. 2) The training process is not oriented towards translation evaluation metric: BLEU is not considered in the scoring of translation pairs. 3) This method may cause over-fitting: it is not considered whether the phrases are extracted from a highly probable phrase alignment or from an unlikely one.

To unify the format of translation knowledge, Marcu and Wong (2002) proposed a phrase-based, joint probability model with EM training to do direct phrase alignment of training data. Since this method suffers from over-fitting, subsequent researchers introduced prior into the generative process: Blunsom et al. (2008) and DeNero et al. (2008) proposed to use dirichlet processing to do structure alignment with sampling training. However, it is still very difficult to integrate various kinds of features, and the model cannot be optimized for SMT performance directly.

To integrate various kinds of features and optimize translation sub-model regarding BLEU directly, Deng et al. (2008) proposed a discriminative phrase extraction method with data-driven features to capture the quality and confidence of phrases and phrase pairs. The feature weights are optimized jointly with the translation engine to maximize the end-to-end system performance. Huang and Xiang (2010) further proposed the use of annotated alignment results to extract annotated translation pairs which are used as training samples for discriminative model training. DeNero and Klein (2010) also used annotated alignment results in a similar way using MIRA training with precision and recall of the translation pairs as training objective (not BLEU-oriented). Such models may still suffer from over-fitting, as the features used may not be powerful enough to separate highly probable translation pairs from unlikely ones.

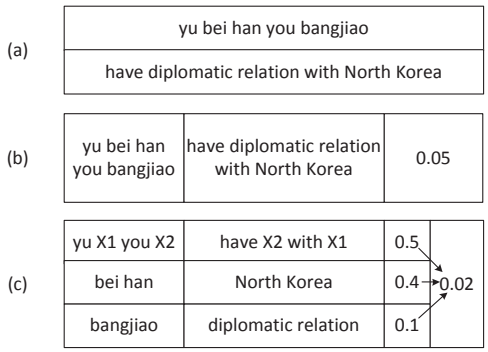

Figure 1. An example for over-fitting

Unlike explanatory capacity, generalization capacity is not well studied for discriminative models (Dirichlet processing is a good choice to handle over-fitting for generative translation modeling). In Figure 1, the sentence pair: "*yu bei han you bangjiao*"/"*have diplomatic relation with North Korea*" has two alternative explanations: 1) this sentence pair is generated by only one translation pair, as shown in (b). 2) this sentence pair is generated by the combination of three translation pairs, as shown in (c). An over-fitted model prefers long translation pairs, and thus overestimates their probabilities while underestimating those of short ones. Such model will fail to generalize on unseen data, such as "*yu riben you jingmao guanxi*"/"*have trading with Japan*".

The only solution to over-fitting for nearly all discriminative translation modeling approaches is the length constraint to the source and/or target side of translation pairs. That is, the long translation pairs, which have weak generalization capacity, are simply filtered away. This solution, however, also discards long but useful phrases like "*I would like to have*".

Blunsom et al. (2008) used a discriminative latent variable model with each translation pair as a feature and a $L2$ regularization to deal with over-fitting, and in order to incorporate language model, sampling method is adopted for training. Wuebker et al. (2010) used leaving-one-out (L1O) to deal with over-fitting and forced alignment to deal with the errors introduced by incorrect word alignment. The basic idea is to use the trained SMT decoder to re-decode the training data, and then use the decoded result to re-calculate translation pair probabilities. Since the correct target sentence (i.e. the target side of training data) is not guaranteed to be generated by SMT decoder, forced alignment is used to generate the correct target sentence by discarding all phrase translation candidates which do not match any sequence in the correct target sentence. Since only the correct target sentence can be generated, language model is useless during decoding, so the weight for language model is set to be zero. Leaving-one-out (L1O) estimation is computed in the following way: For a sentence pair (f_i, e_i), firstly, $(f_i, e_i)$is removed from the training data, and a translation sub-model $TM_i'(\tilde{f}, \tilde{e})$ is trained on the remaining data. Then the partial count[1] for translation pair $(\tilde{f}', \tilde{e}')$ extracted from (f_i, e_i) is defined as the translation probability given by $TM_i'(\tilde{f}', \tilde{e}')$. The final translation sub-model is calculated using MLE based on this partial count.

3 Unified Translation Modeling

Purely discriminative model (PDM) is widely used to integrate various kinds of features into a linear framework, including sub-model features (such as language sub-model and conditional translation probabilities) and fine-grained features (such as translation pairs) (Liang et al., 2006; Arun and Koehn, 2007; Chiang et al., 2008). The weights of the sub-model features cannot be reliably tuned, since the very few sub-model features are overwhelmed by the huge number of fine-grained features. Moreover, each sub-model feature corresponds to a particular kind of knowledge, while the fine-grained features are simply indicator features of the same kind of knowledge (viz. translation sub-model). Therefore it is inappropriate to handle them in a single training process. We introduce a new sub-model which integrate all the indicator features, and then combine the new sub-model with other sub-model features into the conventional framework, so that the explanatory capacity of the indicator features can be maintained and the weights of the sub-models can be balanced.

Based on the linear framework and the conventional sub-models:

$$e_{ci} = \underset{e_i'}{argmax}(\sum_i \lambda_i \psi_i (f_i, e_i'))$$

we introduce a new purely discriminative translation sub-model TM_s:

$$e_{ci} = argmax_{e_i'}(\sum_i \lambda_i \psi_i (f_i, e_i') + \lambda_s TM_s) \quad (1)$$

where f_i is the source sentence and e_i' and e_{ci} are the translation candidates. ψ is the original sub-model vector. λ is the weight vector for ψ and TM_s. The whole model (equation (1)) is called unified translation model (UTM). TM_s is trained with BLEU, and is defined as:

$$TM_s = \sum_j w_j \psi_j' (f_i, e_i') \quad (2)$$

where ψ' is the feature vector, and here we use all the pairs in the translation table. w is the feature weight vector for ψ'.

There are two reasons to take all translation pairs as features. Translation pairs are the primary objects in SMT decoding. It is much better to make the training process directly alter the value of each translation pair rather than through mediation of data-driven features (such as features used in Deng et al. (2008)). Using translation pairs as features also allows training method to affect the pairs directly so as to handle over-fitting, and punish the incorrect translation pairs generated by word alignment errors.

Note that in equation (1), there are only a small number of sub-models, and the sub-model weights in this linear framework can still be tuned by MERT over a small development dataset. In equation (2), however, there are a huge number of fea-

[1] In conventional pipeline, partial count means the exact number of translation pairs extracted from a sentence pair.

tures, and their feature weights can only be trained by scalable method over the entire training data (Chiang, 2008). The complete training dataset is used to train TM_s. In our unified framework, we can combine the MERT and scalable training with a huge number of features, and also use both training data and development data to improve SMT performance.

4 The Unified Training Framework

We train λ by MERT and w by three scalable training methods, which are further discussed in section 5. MERT and scalable training are unified into an interactive training process, as shown in Figure 2.

Algorithm Interactive Training

INPUT: Training data(F, E); Develop data(F_d, E_d)
OUTPUT: λ and w
1: while(BLEU on (F_d, E_d) is improved)
2: fix λ, and train w using scalable method on(F, E)
3: fix w, and train λ using MERT on (F_d, E_d)
4: **Return** λ and w

Figure 2. Training method for UTM

Given initial λ and w, we first fix λ and train w using scalable method on training data (F, E), then we fix the new w and update λ using MERT on development data (F_d, E_d). MERT and scalable training continue interactively until translation performance is not improved on development data (F_d, E_d). We use 0 as initial w and the trained sub-model weights (using MERT) as initial λ. Our scalable training methods are based on the n-best translation candidates of F. Since the n-best lists nearly never use all the features (translation pairs), multiple iterations of scalable training are needed for a stable stagnation point of feature weights.

5 Scalable Training for TM_s

Scalable training methods (Perceptron, MIRA and OWL-QN) are used to train the purely discriminative translation model TM_s with large number of features. In order to optimize SMT performance, the scalable training tunes the weights w to push the best translation candidate upward to be the first one in n-best list according to equation (1).

In order to perform scalable training, the n-best candidates should be ranked according to the similarity with the correct target sentence. BLEU is the most natural choice to be the similarity measure as it is also the ultimate evaluation criterion. However, BLEU is a document-level metric rather than sentence-level. Accordingly, this paper adopts the sentence-level approximated BLEU proposed in Chiang et al. (2008)[2]. The re-ranked best-1 list is called $Oracle_1(F)$ for the training data (F, E) in the following. This ranking process of the n-best candidates is referred as oracle ranking.

5.1 Perceptron

Perceptron (Rosenblatt, 1962) is an incremental training procedure (i.e. stochastic approximation) which optimizes a minimum square error (MSE) loss function. Here, we use the average perceptron (Collins, 2002) for our scalable training, which is shown to be more effective than the standard one. The update rule on an example (f_i, e_{ci}) is:

$$w \leftarrow w + \psi'(f_i, e_{ci}) - \psi'(f_i, e_{oi}) \quad (3)$$

where f_i refers to the i^{th} source sentence in training data, e_{oi} refers to the oracle candidate and e_{ci} refers to the best translation candidate for f_i (in equation (1)). ψ' is the feature vector of the translation candidate (in equation (2)). w is the feature weight for training in purely discriminative translation sub-model TM_s (in equation (2)).

Algorithm Perceptron

INPUT: Training data: (F, E) ; Initial sub-model weights: λ; Initial weights: w
OUTPUT: Trained weights: w
1: while(*BLEU* is improved)
2: decode F to acquire the n-best results $GEN_n(F)$ with λ and w, compute *BLEU*
3: get oracle set $Oracle_1(F)$ from $GEN_n(F)$ using oracle ranking method.
4: while(*BLEU* is improved)
5: $w^0 = w$
6: for each (f_i, e_{ci}, e_{oi}) in $(F, GEN_1(F), Oracle_1(F))$
7: if(e_{ci}!= e_{oi})
8: $w^{i+1} \leftarrow w^i + \psi'(f_i, e_{ci}) - \psi'(f_i, e_{oi})$
9: else do nothing
10: $w = \frac{1}{N}\sum_{i=1}^{N} w^i$
11: re-rank $GEN_n(F)$ with w, and compute BLEU
12: **Return** w

Figure 3. Perceptron for w

The average perceptron algorithm for our scalable training is listed in Figure 3. The training starts with the n-best candidates from an initial w and λ

[2] We also try the method in Watanabe et al. (2007), and it was found that the method made scalable training unstable in our frame work.

(line 2). Then, with the oracle results from oracle ranking (line 3), average perceptron is applied to update w until BLEU reaches stagnation point before re-decoding the training data (line 5-11). Finally, we perform re-decoding of the training data with the updated w, and run perceptron again until it reaches stagnation point. Note that the trained weight vector w should be used as the initial weight vector w^0 for the next perceptron training (line 6), otherwise, the weights for the features which are not used in the new n-best list will be trained to be 0, even though they are harnessed in the n-best lists in previous loops, and the trained model will be unstable.

5.2 MIRA

MIRA is widely used for various NLP tasks, especially for parsing (McDonald et al., 2005) and SMT (Watanabe et al., 2007; Chiang et al., 2008). This paper also uses MIRA to train our pure discriminative translation sub-model TM_s. The gist of MIRA is to keep the norm of the updates to the weight vector as small as possible, while maintaining a margin larger than the loss (BLEU is used to compute loss in equation (5)) of the incorrect classification. The updated rule is to update w to the value of w' which minimizes:

$$\frac{1}{2}\|w' - w\|^2 + C\sum_{i=1}^{m}(l_i - \Delta f_i \cdot w') \quad (4)$$

where C is a positive parameter which controls the influence of the slack term on the objective function and it is set to be 0.01 here. l_i is the (sentence) BLEU loss of the best candidate[3] e_{ci} compared with the oracle result e_{oi} for the i^{th} source sentence f_i:

$$l_i = BLEU_{sent}(f_i, e_{oi}) - BLEU_{sent}(f_i, e_{ci}) \quad (5)$$

and Δf_i is computed by:

$$\Delta f_i = \psi'(f_i, e_{oi}) - \psi'(f_i, e_{ci})$$

By solving equation (4) using the Lagrange dual form, we can get the update rule on sample (f_i, e_{ci}):

$$w \leftarrow w + \tau(\psi'(f_i, e_{ci}) - \psi'(f_i, e_{oi})) \quad (6)$$

where $\tau = \dfrac{\Delta f_i \cdot w + \sqrt{l_i}}{\left\|\psi'(f_i, e_{oi}) - \psi'(f_i, e_{ci})\right\|^2}$

The final training process is also a perceptron like training by replacing the update equation (3) with the new update equation (6).

5.3 OWL-QN

We try to add $L1$ regularizer to enhance the generalization of TM_s. To optimize TM_s with $L1$ is not very easy since its gradient is discontinuous when some weights are equal to zero. Andrew and Gao (2007) described an estimation method with a modified L-BFGS called OWL-QN (orthant-wise limited-memory quasi-newton), and this method can effectively handle the discontinuous gradient.

OWL-QN forbids any two consecutive points forming a line passing through zero, and it uses L-BFGS to approximate the Hessian matrix with $L1$.

Algorithm OWL-QN

INPUT: Training data: (F, E); Model weights: λ; Initial weights: w
OUTPUT: Trained weights: w
1: $GEN(F)$={};
2: while(*BLEU* is improved)
3: decode F to acquire the n-best results $GEN_n(F)$with λ and w, compute *BLEU*
4: $GEN(F)$ += $GEN_n(F)$;
5: build the positive/negative samples with $GEN(F)$ using oracle ranking method
6: train w with OWL-QN
7: **Return** w

Figure 4. OWL-QN for w

Optimization is challenging when expending BLEU with $L1$ as loss function, and here, we use logistic loss instead. The training process using OWL-QN with logistic loss is shown in Figure 4. When we get the n-best candidates for F (line 3), we add them to the candidate pool $GEN(F)$ (line 4). Each iteration adds extra samples to $GEN(F)$, so that more feature weights could be updated. We adopt the oracle ranking method to rank $GEN(F)$, and the top half candidates in $GEN(F)$ are used as the positive samples and the left as the negative samples (line 5). At last, we use OWL-QN to optimize w using the positive and negative samples (line 6).

[3] We find prediction-based (PB) and max-loss (ML) (Crammer et al., 2006) can achieve similar performance in our framework, and we use PB for efficiency.

6 Experiment

The experiments evaluate the performance of our model and training methods in a Chinese-English setting. Translation performance is evaluated by case-insensitive BLEU4.

Our decoder is a state-of-the-art implementation of hierarchical phrase-based model (HPM) (Chiang, 2007) with standard features, including language sub-model, translation sub-model, etc. Our 5-gram language model is trained from the Xinhua section of the Gigaword corpus. FBIS newswire corpus is our training data, which is used to extract translation pairs and train the scalable feature weights w in equation (2). The translation pairs are extracted as in Chiang (2007) from word alignment matrixes, which are generated by running GIZA++ (Och and Ney, 2003) in two directions, and then symmetrized using the grow-diag-final method (Koehn et al., 2003). The idea of unification of MERT and MIRA in UTM is evaluated against PDM, which uses the same features as in UTM. The generalization capacity of UTM using MIRA (UTM_{MIRA}) is evaluated against three baselines, viz. leaving-one-out (L1O), UTM using Perceptron (UTM_{PERC}) and UTM using OWL-QN (UTM_{OWL}).

The NIST'05 test set is used as our development dataset to tune the sub-model weights λ in equation (1) and the NIST'06 and NIST'08 test sets are used as our test sets.

6.1 Explanatory Capacity

As mentioned in section 2, one of the goals of translation modeling is to well represent the training data. Here we use BLEU for **training** data as the measure of explanatory capacity. In the first experiment, we measure the explanatory capacity of TM_s trained by three scalable training methods. The baseline results are HPM, PDM and L1O, shown in Table 1. PDM can get a much better BLEU score on training data compared with other two baselines. The reason may be that BLEU on the training data is the objective function during the PDM training.

System	HPM	PDM	L1O
BLEU on Training	22.94	24.72	24.19

Table 1. Explanatory capacity of baselines

All three scalable training methods are not guaranteed to improve BLEU in any iteration of training phase on the training data. In fact, we found that after 10 iterations, the performance (in BLEU) becomes unstable, so in our experiments we only run the scalable training for 10 iterations on the whole training corpus. Table 2 shows BLEU of the 10 iterations of the three training methods. The n-best size for scalable training is 50. All three scalable training methods can significantly improve the SMT performance on training data. UTM_{PERC} and UTM_{MIRA} get almost the same performance on training data (as they share the same framework), while UTM_{OWL} is slightly worse.

Iteration	UTM_{PERC}	UTM_{MIRA}	UTM_{OWL}
0	22.94	22.94	22.94
1	24.93	24.95	22.33
2	24.62	24.61	23.59
3	25.33	25.45	23.61
4	25.1	25.56	24.16
5	25.45	25.58	24.07
6	25.57	25.59	24.33
7	25.71	25.63	24.45
8	25.67	25.68	24.53
9	25.67	25.70	24.51

Table 2. BLEU for the 10 iterations on **training** dataset

The reason why UTM_{OWL} cannot improve BLEU as significantly as the other two methods could be the loss function we used, which is logistic loss (not BLEU). UTM_{OWL} treats translation as binary classification and uses half of the candidates with higher BLEU as positive samples, while the other half as negative. Binary classification is simply inconsistent with the scaling nature of BLEU.

There are 1,165,405 features (translation pairs) in the end-to-end translation sub-model, and 243,840 features used by the best output (the best candidate for each source sentence in training data) of the baseline system. We distinguish the features used for training (#Feature of Used, the features those have an opportunity to be trained) from the features whose weights are not trained to be zero (#Feature of Trained) in Table 3. Note that, there are a large number of the pairs which are never used in the n-best lists, so the weights for them will always be zero. It is not surprising that UTM_{OWL} leads to much more features, because UTM_{OWL} can use all the candidates in the n-best lists to update the weights, while Perceptron and MIRA can only

use two candidates (the best one and the oracle) of the n-best list for each source sentence in training data.

#Feature	UTM_{PERC}	UTM_{MIRA}	UTM_{OWL}
Used	507,403	513,327	578,625
Trained	483,220	474,054	557,701

Table 3. Numbers of features after training

6.2 Translation Performance

We compare the scalable training methods on test datasets against the baselines of HPM, PDM and L1O. The results are shown in Table 4.

	Nist'06	Nist'08
HPM	30.54	22.51
PDM	30.97	23.34
L1O	30.63(+0.09)	23.23(+0.72)
UTM_{PERC}	31.06(+0.52)	22.95(+0.44)
UTM_{OWL}	31.10(+0.56)	23.52(+1.01)
UTM_{MIRA}	31.49(+0.95)	23.91(+1.40)

Table 4. Final results on test datasets

L1O improves the performance on Nist'08 but not significantly on Nist'06. Since both UTM_{PERC} and UTM_{MIRA} get almost the same performance on training data, and UTM_{MIRA} outperforms UTM_{PERC} on the two test datasets a lot, it is confirmed that UTM_{MIRA} is better in generalization capacity[4] (to be deeply analyzed in the next section). UTM_{OWL} with logistic loss is not very good on Nist'06, while it significantly improves the performance on Nist'08. UTM_{MIRA} gets larger improvement than other methods. Note that the BLEU differences between UTM_{MIRA} and the three baselines are statistically significant (Koehn, 2004). Compared with PDM, UTM_{MIRA} achieves a better improvement, and also the training time of our method (34 hours, for UTM_{MIRA}, UTM_{PERC} and UTM_{PERC})[5] is much shorter than that of PDM (119 hours)[6].

Similar with MERT, the MIRA training for PDM needs to decode the training data iteratively, and it took 20 cycles before convergence. In contrast, UTM requires took only 5 cycles.

6.3 Generalization Capacity

The Nist'08 evaluation set is a mixture of newswire text and web text. 691 sentence pairs in Nist'08 are in the same domain with the training data (newswire), while the rest sentence pairs (web data) are not. We evaluate these two portions separately to confirm the better generalization capacity of our method. The results are shown in Table 5.

	Nist'08(News)	Nist'08(Web)
HPM	27.70	15.49
L1O	28.48(+0.78)	15.91(+0.42)
UTM_{PERC}	28.11(+0.41)	15.87(+0.38)
UTM_{OWL}	28.47(+0.77)	16.72(+1.23)
UTM_{MIRA}	28.84(+1.14)	17.14(+1.65)

Table 5. Evaluation for News/Web portion

It is obvious that L1O performs not very well on out-of-domain data. UTM_{OWL} performs better on web portion (1.23) than on news portion (0.77). For L1O, the improvement regarding out-domain data (0.42) is not significant, which means L1O performs not very well in generalization capacity. For UTM_{PERC}, performance is not improved significantly on both parts. The improvement of UTM_{MIRA} on web text (1.65) is much larger than that of UTM_{PERC} (0.38). The better performance on out-of-domain data confirms the better generalization capacity of UTM_{MIRA}. UTM_{MIRA} also improves significantly on news data (1.14).

7 Conclusion

In order to handle the inconsistence between translation modeling and decoding, we propose a new discriminative translation sub-model, which is optimized with the same criterion as the translation output is evaluated (BLEU). The new translation sub-model uses all translation pairs in the translation table as features, and avoids over-fitting by MIRA training. MIRA for the new sub-model and MERT for the conventional framework are unified into an interactive training process. The unification of MIRA and MERT under our new framework can help the training process to achieve a better stagnation point. Our framework achieves better

[4] This observation is contrary to that in Arun and Koehn (2007), which shows perceptron and MIRA get comparable results on a small in-domain training (one-eighth of our training dataset) and test datasets.

[5] The most time-cost step is the decoding of the training data, and the time of scalable training is almost the same for UTM_{MIRA}, UTM_{PERC} and UTM_{OWL}

[6] Our experiments were run on a machine with 16 CPU cores (each 2520MHz) and 32G RAM. No distributed computing was involved; instead the system is speeded up by multi-threading (16 threads).

performance than the state-of-the-art baseline and purely discriminative model. Within our framework, we also confirm the better generalization capacity using MIRA than perceptron as the scalable training method.

In the future, we will try to compare the effect of different kinds of regularizers ($L1$ and $L2$) for the over-fitting problem of translation modeling, and we will also try to add new scalable distortion sub-model and language sub-model in the same way as the new added translation sub-model.

References

Galen Andrew and Jianfeng Gao. 2007. *Scalable Training of L1-Regularized Log-Linear Models*. In *Proceedings of* ICML.

Abhishek Arun and Philipp Koehn. 2007. *Online Learning Methods For Discriminative Training of Phrase Based Statistical Machine Translation*. In *Proceedings of* MT Summit XI.

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Peitra, Robert L. Mercer. 1993. *The Mathematics of Statistical Machine Translation: Parameter Estimation*. *Computational Linguistics*, 19(2): 263-311.

Phil Blunsom and Miles Osborne. 2008. *Probabilistic Inference for Machine Translation*. In *Proceedings of* EMNLP.

Phil Blunsom, Trevor Cohn, Chris Dyer and Miles Osborne. 2009. *A Gibbs Sampler for Phrasal Synchronous Grammar Induction.* In *Proceedings of* ACL and IJCNLP. Pages: 782-790.

David Chiang. 2007. *Hierarchical Phrase-based Translation.* Computational Linguistics, 33(2).

David Chiang, Yuval Marton and Philip Resnik. 2008. *Online Large-Margin Training of Syntactic and Structural Translation Features.* In *Proceeding of* ACL. Pages: 224-233.

Michael Collins. 2002. *Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms*. In *Proceedings of* EMNLP. Pages:111-118

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2003. *Online passive aggressive algorithms.* In *Proceedings of* NIPS.

John DeNero, Alexandre Bouchard-Cote and Dan Klein. 2008. *Sampling Alignment Structure under a Bayesian Translation Model.* In *Proceedings of* ACL and HLT. Papers: 25-28.

John DeNero and Dan Klein. 2010. *Discriminative Modeling of Extraction Sets for Machine Translation*. In *Proceedings of* ACL. Pages: 1453-1463.

Yonggang Deng, Jia Xu and Yuqing Gao. 2008. *Phrase Table Training For Precision and Recall: What Makes a Good Phrase and a Good Phrase Pair.* In *Preceedings of* ACL. Pages:81-88.

Fei Huang and Bing Xiang. 2010. *Feature-Rich Discriminative Phrase Rescoring for SMT.* In *Proceedings of* COLING, Pages: 492-500.

Philipp Koehn. 2004. *Statistical Significance Tests for Machine Translation Evaluation.* In *Proceedings of* EMNLP, Pages: 388-395.

Percy Liang, Alexandre Bouchard-Cote, Dan Klein, and Ben Taskar. 2006. *An end-to-end discriminative approach to machine translation.* In *Proceedings of* COLING and ACL, Pages:761-768

Daniel Marcu and William Wong. 2002. *A Phrase-Based, Joint Probability Model for Statistical Machine Translation.* In *Proceedings of* EMNLP, Pages: 133-139.

Ryan McDonald, Koby Crammer and Fernando Pereira. 2005. *Online large-margin training of dependency parsers*. In *Proceeding of* ACL. Pages: 91-98.

Franz Josef Och and Hermann Ney. 2003. *A systematic comparison of various statistical alignment models.* Computational Linguistics, 29(1): 19–51.

Franz Josef Och. 2003. *Minimum error rate training in statistical machine translation.* In *Proceedings of* ACL, Pages: 160-167.

Franz Josef Och and Hermann Ney. 2003. *The Alignment Template Approach to Statistical Machine Translation.* Computational Linguistics, 30(4): 417-449.

Stephan Vogel. 2005. *PESA: Phrase Pair Extraction as Sentence Splitting.* In *Proceedings of* MT Summit.

Taro Watanabe, Jun Suzuki, Hajinme Tsukada and Hideki Isozaki. 2007. *Online Large-Margin Training for Statistical Machine Translation.* In *Proceeding of* EMNLP. Pages: 764-773.

Joern Wuebker, Arne Mauser and Hermann Ney. 2010. *Training Phrase Translation Models with Leaving-One-Out.* In *Proceeding of* ACL. Pages: 475-484.

Hao Zhang, Chris Quirk, Robert Moore, and Daniel Gildea. 2008. *Bayesian learning of non-compositional phrases with synchronous parsing.* In *Proceedings of* ACL, Pages: 314-323.