An Empirical Comparison of Joint Optimization Techniques for Speech Translation

Masaya Ohgushi, Graham Neubig, Sakriani Sakti, Tomoki Toda, Satoshi Nakamura

Graduate School of Information Science, Nara Institute of Science and Technology
{masaya-o,neubig,ssakti,tomoki,s-nakamura}@is.naist.jp

Abstract

Speech translation (ST) systems consist of three major components: automatic speech recognition (ASR), machine translation (MT), and speech synthesis (SS). In general the ASR system is tuned independently to minimize word error rate (WER), but previous research has shown that ASR and MT can be jointly optimized to improve translation quality [1]. Independently, many techniques have recently been proposed for the optimization of MT, such as empirical comparison of joint optimization using minimum error rate training (MERT) [2], pairwise ranking optimization (PRO) [3] and the batch margin infused relaxed algorithm (MIRA) [4]. The first contribution of this paper is an empirical comparison of these techniques in the context of joint optimization. As the last two methods are able to use sparse features, we also introduce lexicalized features using the frequencies of recognized words. In addition, motivated by initial results, we propose a hybrid optimization method that changes the translation evaluation measure depending on the features to be optimized. Experimental results for the best combination of algorithm and features show a gain of 1.3 BLEU points at 27% of the computational cost of previous joint optimization methods.

Index Terms: speech translation, machine translation, automatic speech recognition, joint optimization

1. Introduction

Speech translation (ST) is an important technology for cross-lingual oral communication, the role of which is rapidly increasing in the modern interconnected information age. ST systems have three components: automatic speech recognition (ASR), machine translation (MT) and speech synthesis (SS). Traditionally these parts are individually optimized by different evaluation metrics. In ASR the widely used metric is word error rate (WER) [5], and in MT accuracy is usually measured by a wide variety of metrics, the most widely used being BLEU [6].

It is well known that if we are able to recognize speech without error, translation quality improves [7, 8]. However zero word error is impossible to achieve with current technology given the ambiguity included in the speech signal. Previous research has attempted to improve MT accuracy in the face of imperfect recognition by training parameters of the log-linear model to directly optimize the quality to the final translation output measured in BLEU [1], allowing the model to directly select recognition candidates that are easy to translate.

In this previous research, the optimization method is minimum error rate training (MERT) [2], which can only use a limited number of parameters and often has problems with overfitting. In order to fix these problem in the context of MT, a number of optimization methods including pairwise ranking optimization (PRO) [3] and the batch margin infused relaxed algorithm (MIRA) [4] have been proposed. The first contribution of this paper is an empirical comparison of these three algorithms in the framework of joint optimization for speech translation.

We also explore several extensions to these algorithms. First we propose joint optimization using lexicalized features for each word in the recognized sentence, which we hope will increase our power to discriminate between easy-to-translate and difficult-to-translate sentences. In addition, it has been noted that while PRO originally used sentence BLEU+1, this evaluation metric introduces a bias toward short translations [9], a trend than we find particularly problematic when considering multiple ASR hypotheses. We examine Smooth BLEU+1 [9] as a solution to this problem. Finally, based on initial experimental results we propose a hybrid optimization method that uses Smooth BLEU+1 for the optimization of dense features, and regular BLEU+1 for the optimization of sparse features.

Experimental results on a travel conversation corpus [10] showed that PRO with Smooth BLEU+1 as an evaluation measure achieves a significant improvement over the other optimization methods. The proposed hybrid optimization method also saw a small increase in translation quality when using sparse features.

2. Speech Translation Systems

In speech translation, the input speech signal $\boldsymbol{X}$ is first fed into the ASR module, generating the recognition output $\boldsymbol{F}$ in the source language. The recognition hypothesis $\boldsymbol{F}$ is then passed to the MT module to obtain the translation $\boldsymbol{E}$ in the target language.

2.1. Automatic Speech Recognition

In ASR we model the posterior probability $P(\boldsymbol{F}|\boldsymbol{X})$ of $\boldsymbol{F}$ given $\boldsymbol{X}$ through a log-linear model [11]

$$P(\boldsymbol{F}|\boldsymbol{X}, \boldsymbol{\lambda}_{ASR}) = \frac{1}{Z_F}\exp\Big\{\sum_i \lambda_i \phi_i(\boldsymbol{F}, \boldsymbol{X})\Big\}, \quad (1)$$

$$Z_F = \sum_{\boldsymbol{F}} \exp\Big\{\sum_i \lambda_i \phi_i(\boldsymbol{F}, \boldsymbol{X})\Big\}, \quad (2)$$

where $\phi_i(\boldsymbol{F}, \boldsymbol{X})$ are the feature functions, λ_i are the weights corresponding to the feature functions, and Z_F is the normalization term to ensure that probabilities sum to one. Usually, the ASR module uses a small number of features including the word penalty and log probability of the acoustic and source language models. The recognition hypothesis of maximum probability $\hat{\boldsymbol{F}}$ is found by decoding given speech signal $\boldsymbol{X}$:

$$\hat{\boldsymbol{F}} = \arg\max_{\boldsymbol{F}}(P(\boldsymbol{F}|\boldsymbol{X}, \boldsymbol{\lambda_{ASR}})). \quad (3)$$

2.2. Machine Translation

We also model the MT posterior probability $P(\boldsymbol{E}|\boldsymbol{F})$ through a log-linear model

$$P(\boldsymbol{E}|\boldsymbol{F}, \boldsymbol{\lambda}_{MT}) = \frac{1}{Z_E}\exp\left\{\sum_i \lambda_i \phi_i(\boldsymbol{E}, \boldsymbol{F})\right\}, \quad (4)$$

where $\phi_i(\boldsymbol{E}, \boldsymbol{F})$ are feature functions, λ_i are weights corresponding to the features functions, and Z_E is the normalization term. As our baseline translation model, we use a standard set of 14 features functions including word penalty, phrase penalty, log probability of translation model, target language model, and reordering models [12]. The MT module also finds the hypothesis of maximal probability $\hat{\boldsymbol{E}}$ by decoding given recognition hypothesis $\boldsymbol{F}$:

$$\hat{\boldsymbol{E}} = \arg\max_{\boldsymbol{E}}(P(\boldsymbol{E}|\boldsymbol{F}, \boldsymbol{\lambda_{MT}})). \quad (5)$$

3. Training of Feature Weights

In traditional ST systems the parameters of the log-linear model, the weights of features, are trained to minimize the WER in the ASR module

$$\hat{\boldsymbol{\lambda}}_{\boldsymbol{ASR}} = \arg\min_{\boldsymbol{\lambda_{ASR}}}(\text{WER}(\mathcal{F}^*, \hat{\mathcal{F}})), \quad (6)$$

where $\mathcal{F}^* = \{\boldsymbol{F}_1^*, \boldsymbol{F}_2^*, \boldsymbol{F}_3^*, ...\}$ are the source sentence references and $\hat{\mathcal{F}} = \left\{\hat{\boldsymbol{F}}_1, \hat{\boldsymbol{F}}_2, \hat{\boldsymbol{F}}_3, ...\right\}$ are the recognition outputs which is obtained according to Equation (3). WER is calculated as the number of substitutions, deletions, and insertions, divided by the number of words in the reference.

In the MT module the parameters of the log-linear model are trained by maximizing an evaluation measure such as the BLEU score [6]

$$\hat{\boldsymbol{\lambda}}_{\boldsymbol{MT}} = \arg\max_{\boldsymbol{\lambda_{MT}}}(\text{BLEU}(\mathcal{E}^*, \hat{\mathcal{E}})), \quad (7)$$

where $\mathcal{E}^* = \{\boldsymbol{E}_1^*, \boldsymbol{E}_2^*, \boldsymbol{E}_3^*, ...\}$ is the translation reference and $\hat{\mathcal{E}} = \left\{\hat{\boldsymbol{E}}_1, \hat{\boldsymbol{E}}_2, \hat{\boldsymbol{E}}_3, ...\right\}$ is the translation output obtained according to Equation (5). BLEU score is generally calculated as the geometric mean of the n-gram precisions of the system output from one to four, multiplied by a brevity penalty BP

$$\text{BLEU}(\mathcal{E}^*, \hat{\mathcal{E}}) = \text{BP}(\mathcal{E}^*, \hat{\mathcal{E}}) * (\prod_{n=1}^{4} \text{prec}_n(\mathcal{E}^*, \hat{\mathcal{E}}))^{1/4}. \quad (8)$$

The n-gram precision being equal to the number of n-gram matches between the reference and the output, divided by the total number of n-grams in the reference

$$\text{prec}_n(\mathcal{E}^*, \hat{\mathcal{E}}) = \frac{\sum_i \text{match}_n(\boldsymbol{E}_i^*, \hat{\boldsymbol{E}}_i)}{\sum_i \text{total}_n(\boldsymbol{E}_i^*)}. \quad (9)$$

The brevity penalty is added to prevent unreasonably short system outputs from being assigned high precisions, and is calculated using the lengths of the system output and reference as follows:

$$\text{BP}(\mathcal{E}^*, \hat{\mathcal{E}}) = \min\left(1, \exp\left(1 - \frac{\sum_i |\boldsymbol{E}_i^*|}{\sum_i |\hat{\boldsymbol{E}}_i|}\right)\right). \quad (10)$$

The most common way to solve these optimization problems is through batch optimization, which iteratively generates n-best lists through decoding and adjusting weights given the n-best list. We show the example for MT as follows:

1. Decode Corpus: $\hat{\mathcal{E}} = n\text{-bests}(\mathcal{E}|\mathcal{F}; \boldsymbol{\lambda_{MT}}))$
2. Aggregate: $\mathcal{E} \leftarrow \mathcal{E} \cup \hat{\mathcal{E}}$
3. Optimize: $\boldsymbol{\lambda_{MT}} \leftarrow$ Optimize$(\mathcal{E}^*, \mathcal{E})$
4. Repeat

In this section we describe 3 different techniques to solve the optimization problem in Step 3 of the above algorithm.

3.1. Minimum Error Rate Training

Minimum error rate training (MERT) [2] is currently the most widely used technique for optimization in MT. It works by directly finding the parameter setting that maximizes the evaluation measure on the tuning set. In generally evaluation measures such as a BLEU or WER are piecewise linear, and thus can not be maximized using gradient descent, so MERT solves this problem by making a series of tractable one-dimensional line searches optimizing one parameter while keeping all other parameters fixed [13]. MERT is able to directly maximize corpus-wide evaluation measures such as BLEU, but is also inefficient for larger feature sets because learning time increases linearly in proportion to the total number of unique features and is also known to be prone to overfitting to the tuning set.

3.2. Batch Margin Infused Relaxed Algorithm

The batch margin infused relaxed algorithm (MIRA) [4] is a structured training algorithm that holds promise to solve these two problems of MERT. It is much more efficient for sparse features than MERT, and can be expected to prevent over-fitting through the use of regularization and a margin. In contrast to MERT's line search, MIRA minimizes the hinge-losses one sentence at a time using the strucutured SVM algorithm [14]. In order to train the structured SVM, we must define an oracle translation $\boldsymbol{E}^*$ that the SVM will optimize towards, and a loss function for each candidate translation. The oracle translation for a particular sentence is chosen by fixing the oracles and candidates for all other sentences and choosing a candidate the aggregated n-best lists such that corpus-based BLEU is maximized, with ties broken in favor of translations with higher model score. The loss for a translation is defined as the difference in BLEU between the candidate $\hat{\boldsymbol{E}}$ and the oracle $\boldsymbol{E}^*$. With the oracle fixed, the objective becomes a standard structured SVM objective, which can be minimized using a cutting-plane algorithm, as described by [15].

3.3. Pairwise Ranking Optimization

Pairwise ranking optimization (PRO) [3] is a method for optimizing weights in MT that focuses on the ranking of the entire n-best list.

3.3.1. Sentence-Based Translation Accuracy Measures

In order to train the ranking, we must first be able to evaluate the accuracy of the reference $\boldsymbol{E}^*$ and output $\hat{\boldsymbol{E}}$ for individual sentences. Unfortunately, BLEU is not well-suited for this task, as many sentences will have no matches for longer n-grams, resulting in a value of zero for Equation (9), which causes BLEU to also reduce to 0. As a solution for this, [16] proposes a new metric called BLEU+1 that smooths the n-gram precisions for $n \geq 2$ by adding 1 to the numerator and denominator:

$$\text{prec+1}_n(\boldsymbol{E}_i^*, \hat{\boldsymbol{E}}_i) = \frac{\text{match}_n(\boldsymbol{E}_i^*, \hat{\boldsymbol{E}}_i) + 1}{\text{total}_n(\boldsymbol{E}_i^*) + 1}. \quad (11)$$

This ameliorates the problem of zero scores for sentence-wise BLEU, but [9] has also recently shown that optimizing towards BLEU+1 results in overly short sentences. As a solution to this, they propose a new measure Smooth BLEU+1, which adjusts the sentence-wise brevity penalty to favor longer sentences:

$$\text{BP+1}(\boldsymbol{E}_i^*, \hat{\boldsymbol{E}}_i) = \min\left(1, \exp\left(1 - \frac{|\boldsymbol{E}_i^*| + 1}{|\hat{\boldsymbol{E}}_i|}\right)\right). \quad (12)$$

In this paper, we compare the effectiveness of both of these measures in the context of joint optimization.

3.3.2. Optimizing the Ranking of Sentence-based Results

Once we have defined BLEU+1 or Smooth BLEU+1 as a sentence-wise measure of translation quality, we then train a ranker to rank translation hypotheses in order of quality using a varient of the RankSVM algorithm [17]. This is done by choosing pairs of translation candidates of varying quality, and training a binary classifier to distinguish between the hypotheses of better or worse quality. In addition, to reduce training time and to prevent the classifier from having to distinguish between hypotheses with very small differences in quality, PRO uses a sampling algorithm to choose training pairs, focusing on only pairs that have a difference in BLEU+1 that exceeds some threshold [3]. This ranking algorithm has the advantage of allowing PRO to bypass MIRA's oracle selection process, and also consider the difference between not only good and less good hypotheses at the top of the n-best list, but also bad and not-so-bad hypotheses at the bottom of the n-best list. This has the potential benefit of increasing robustness towards unseen data, but it is also not theoretically guaranteed that ordering the n-best list correctly will necessarily result in better final translations. As a result, it is necessary to compare the methods empirically on real data, and in this paper we do so in the framework of joint optimization of ASR and MT, as described in the following section.

4. Joint Decoding and Optimization

In general the ASR and MT modules are individually optimized, and MT translates 1-best ASR results. However, the highest probability recognition candidate may not yield the best translation. Previous research has shown that using n-best recognition candidates in the ST systems improves translation quality [1]. In this framework the posterior probability of the ($\boldsymbol{E}$,$\boldsymbol{F}$) sentence pair given $\boldsymbol{X}$ is modeled through a log linear model as follows:

$$P(\boldsymbol{E}, \boldsymbol{F}|\boldsymbol{X}, \boldsymbol{\lambda_{MT}}, \boldsymbol{\lambda_{ASR}}) = \frac{1}{Z_{E,F}} \exp\left\{\sum_i \lambda_i \phi_i(\boldsymbol{E}, \boldsymbol{F}, \boldsymbol{X})\right\} \quad (13)$$

The maximum probability translation hypothesis is found by decoding given speech signal $\boldsymbol{X}$:

$$\hat{\boldsymbol{E}} = \arg\max_{\boldsymbol{E}}(P(\boldsymbol{E}, \boldsymbol{F}|\boldsymbol{X}, \boldsymbol{\lambda_{MT}}, \boldsymbol{\lambda_{ASR}})) \quad (14)$$

In this paper we use n-best recognition candidate for joint decoding. When searching for translation $\boldsymbol{E}$, we consider the space of $\boldsymbol{F}$ as the n-best recognition candidates. The parameters of the ASR and MT modules are expressed as a single log-linear model, the weights of which are trained by maximizing the BLEU score

$$\hat{\boldsymbol{\lambda}}_{\boldsymbol{MT}}, \hat{\boldsymbol{\lambda}}_{\boldsymbol{ASR}} = \underset{\boldsymbol{\lambda_{MT}}, \boldsymbol{\lambda_{ASR}}}{\arg\max} \quad (\text{BLEU}(\mathcal{E}^*, \hat{\mathcal{E}})). \quad (15)$$

$\boldsymbol{F}$: I would like something for the fever

$$\varphi_I(F) = 1, \varphi_{would}(F) = 1, \varphi_{could}(F) = 0, \varphi_{something}(F) = 1, \varphi_{for}(F) = 1 \ldots.$$

Figure 1: Frequency of Recognition Words

Table 1: Corpus size

Source AM Training Data	486 hours
Source LM Training Data	12k sentences
TM Training Data	162k sentences
Target LM Training Data	162k sentences
Tuning Data	610 sentences
Test Data	610 sentences

As this problem reduces to an optimization of translation accuracy over n-best lists, we can use similar methods to those discussed in the context of MT in the previous section. We adopt MERT, PRO and MIRA to optimize the feature weights in our experiments.

5. Optimization with Sparse Features

In this section we expand the set of features used in joint optimization, and propose a hybrid technique for optimizing them effectively.

5.1. Sparse Features over Recognition Candidates

One of the advantages of MIRA and PRO the ability to use many features. To test the efficacy of sparse features in joint optimization we used the frequency of recognised words in the recognition hypothesis as new features. These features allow us to give larger or smaller weights to words that occur in recognition candidates that produce translations of higher or lower quality. For example, in a situation where there are identical words with different spellings ("color"/"colour") these features could learn to favor the spelling that is better represented in the translation model. We prepare the recognition dictionary for ASR and count the matched words in each recognition hypothesis as in Figure 1.

5.2. Hybrid Optimization

In our experiments, with optimization using PRO, we found that neither BLEU+1 nor Smooth BLEU+1 were ideal for optimization with sparse features. Smooth BLEU+1 allowed for the choice of longer sentences, preventing the length of the generated sentences from being too short, but also reduced the lexical diversity of the hypotheses, with many hypotheses sharing a large number of words. On the other hand, sentences selected with BLEU+1 were too short, but also had more lexical diversity, making it easier to optimize the sparse features.

As one solution to this, we propose a simple hybrid method of using the values of the 17 dense features from ASR and MT achieved by optimizing Smooth BLEU+1 and the values of the sparse features achieved by optimizing BLEU+1. This is somewhat similar to [18], in which it was found effective to use one optimization method to achieve an initial estimate for dense weights, and a different method to refine those weights.

6. Experiment and Discussion

6.1. Experimental Condition

We perform experiments on Japanese-English speech translation using data from the BTEC travel conversation corpus [10]. For ASR we used an HMM acoustic model trained on the Corpus of Spontaneous Japanese [19] with HTK using the standard settings and used Julius [20] as a decoder. The translation model is a phrase-based model created with Moses [12]. Tuning and

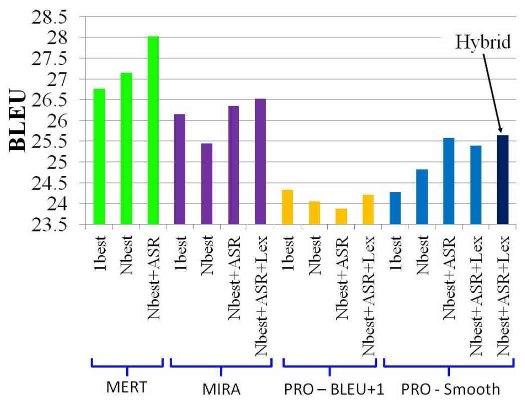

Figure 2: BLEU score in the Tuning Data

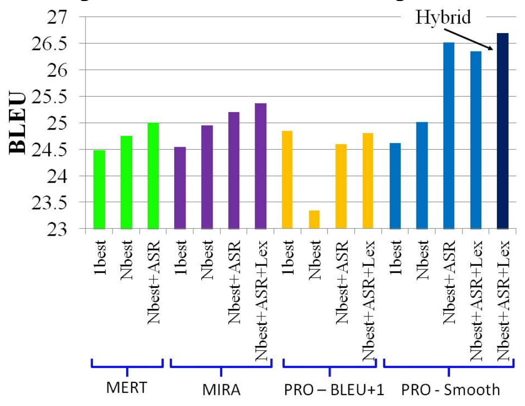

Figure 3: BLEU score in the Test Data

test data are also Japanese speech data from BTEC. Punctuation was removed from all training and testing data. Table 1 notes the sizes of the datasets.

6.2. Tuning settings

We compared MERT, PRO, and MIRA using the implementations provided in Moses [12]. We perform experiments comparing 1-best and 50-best recognition candidates. The default feature set is 14 translation features, [+ASR] indicates adding the 3 features explained in Section 2.1, and [+Lex] indicates new features explained in the Section 5.1. We used the MegaM classifier [21] to train the ranking model for PRO. We use the 50-best translation candidates (for a total of $50 \times 50 = 2500$ when ASR n-best lists are used). The number of iterations is set to 25. We average scores over of three optimization runs to control for optimizer instability [18] and used bootstrap resampling for calculating statistical significance [22].

6.3. Experiment Result and Discussion

Figures 2 and 3 shows the accuracies as measured by BLEU on the tuning and testing data respectively.

First we can see that for most methods joint optimization increases the BLEU score by using the n-best. One exception to this general trend of improvements through joint optimization is optimization with PRO using BLEU+1, for which the translation quality greatly decreases when using the n-best without ASR features. The reason for this is shown in Table 2, which shows the brevity penalty incurred by each optimization method. It can be seen that PRO optimized with BLEU+1 incurs a heavy brevity penalty, particularly when we use n-best. This shows that the problem of optimizing towards short sentences indicated by [9] is even worse when using the large and noisy n-best lists that result from joint optimization. On the other hand, we can see that Smooth BLEU+1 largely resolves this problem.

Table 2: Brevity penalty (values under 0.95 emphasized in bold)

Method	1-best	n-best	n-best+ASR	n-best+ASR+Lex
MERT	1	1	0.999	×
MIRA	0.989	0.968	0.967	0.967
PRO-BLEU+1	0.965	**0.881**	**0.889**	**0.897**
PRO-Smooth	0.969	0.986	0.989	0.989

Table 3: Tuning time in seconds

Method	1-best	n-best	n-best+ASR	n-best+ASR+Lex
MERT	3.4k	210k	230k	×
MIRA	2.9k	21k	61k	61k
PRO-BLEU+1	1.7k	49k	55k	61k
PRO-Smooth	1.7k	45k	46k	57k

Comparing the four optimization types, we can clearly see that on the test set, PRO with Smooth BLEU+1 improves the translation quality over MERT and MIRA. This is in contrast to the tuning set, where MERT over-fitted the tuning data achieving high scores that did not prove to be generalizable. The improvements of PRO with Smooth BLEU+1 and Nbest/ASR features are statistically significant with $P < 0.05$ over MERT, MIRA and PRO with BLEU+1 expect for MIRA with ASR+Lex, which had $P = 0.061$.

As may be expected, adding the ASR features adds an additional gain to all of the methods considered, as without this information it is difficult to discriminate between which of the hypotheses in the n-best list are better than others. The additional sparse features over the recognition hypothesis have the effect of improving the translation quality for MIRA and PRO with BLEU+1 (0.17 and 0.21 points respectively on the test data). The sparse features are not helpful when using PRO with Smooth BLEU+1, but when introducing the hybrid optimization technique we see a small gain of 0.17 points.

Table 3 shows a comparison of time required for the entire tuning process for each method. In general, MERT is the slowest of the methods, particularly for the very large n-best lists that result from joint optimization. The other three methods do not see a large change in speed in the joint optimization setting when ASR features are considered.

7. Conclusion and Future Work

In this paper we performed an empirical study of several modern optimization techniques in the context of joint optimization of ASR and MT. The result shows that PRO with Smooth BLEU+1 provided superior performance to MERT and MIRA. On the other hand, lexical features over the recognition hypothesis yielded only minor improvements in the translation quality.

One major target of future work is a comparison on other data sets. We plan on comparing how the size of the tuning data, accuracy of the ASR, and difficulty of the translation task affect the results presented here. In addition, while we only used n-best lists in this work, it is also possible to perform MT decoding with ASR lattices as input [23], which will potentially lead to further increases in the benefits afforded by joint optimization. Finally, it will be interesting to expand our feature sets to the large number of parameters in the TM, LM, and AM to see if further gains can be achieved.

Acknowledgment: Part of this work was supported by JSPS KAKENHI Grant Number 24240032 and collaborative research with I^{2}R.

8. References

[1] X. He, L. Deng, and A. Acero, “Why word error rate is not a good metric for speech recognizer training for the speech translation task,” in *ICASSP*, 2011.

[2] F. J. Och, “Minimum error rate training in statistical machine translation,” in *ACL*, 2003.

[3] M. Hopkins and J. May, “Tuning as ranking,” in *EMNLP*, 2011.

[4] C. Cherry and G. Foster, “Batch tuning strategies for statistical machine translation,” in *NAACL*, vol. 12, 2012, pp. 34–35.

[5] X. He, L. Deng, and W. Chou, “Discriminative learning in sequential pattern recognition,” *Signal Processing Magazine, IEEE*, vol. 25, no. 5, pp. 14–36, 2008.

[6] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, “BLEU: a method for automatic evaluation of machine translation,” in *ACL*, 2002.

[7] N. Bertoldi, R. Zens, and M. Federico, “Speech translation by confusion network decoding,” in *ICASSP*, 2007.

[8] F. Casacuberta, M. Federico, H. Ney, and E. Vidal, “Recent efforts in spoken language translation,” *Signal Processing Magazine, IEEE*, vol. 25, no. 3, pp. 80–88, 2008.

[9] P. Nakov, F. Guzman, and S. Vogel, “Optimizing for sentence-level BLEU+1 yields short translations,” in *COLING*, 2012, pp. 1979–1994.

[10] T. Takezawa, E. Sumita, F. Sugaya, H. Yamamoto, and S. Yamamoto, “Toward a broad-coverage bilingual corpus for speech translation of travel conversations in the real world,” in *LREC*, 2002.

[11] R. Zhang, G. Kikui, H. Yamamoto, T. Watanabe, F. Soong, and W. Lo, “A unified approach in speech-to-speech translation: integrating features of speech recognition and machine translation,” in *COLING*, 2004.

[12] P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens *et al.*, “Moses: Open source toolkit for statistical machine translation,” in *ACL*, vol. 45, no. 2, 2007, p. 2.

[13] R. Brent, *Algorithms for minimization without derivatives*. Dover Publications, 2002.

[14] C.-N. J. Yu and T. Joachims, “Learning structural SVMs with latent variables,” in *ICML*. ACM, 2009.

[15] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun, “Support vector machine learning for interdependent and structured output spaces,” in *ICML*. ACM, 2004, p. 104.

[16] C.-Y. Kin and F. J. Och, “Orange: a method for evaluating automatic evaluation metrics for machine translation,” in *COLING*, 2004.

[17] R. Herbrich, T. Graepel, and K. Obermayer, “Large margin rank boundaries for ordinal regression,” *NIPS*, pp. 115–132, 1999.

[18] L. Liu, H. Cao, T. Watanabe, T. Zhao, M. Yu, and C. Zhu, “Locally training the log-linear model for SMT,” in *EMNLP*, 2012, pp. 402–411.

[19] K. Maekawa, “Corpus of spontaneous Japanese: Its design and evaluation,” in *ISCA & IEEE Workshop on Spontaneous Speech Processing and Recognition*, 2003.

[20] A. Lee and T. Kawahara, “Recent development of open-source speech recognition engine julius,” in *APSIPA*, 2009, pp. 131–137.

[21] H. Daumé III, “Notes on CG and LM-BFGS optimization of logistic regression,” August 2004, http://hal3.name/megam/.

[22] P. Koehn, “Statistical significance tests for machine translation evaluation,” in *EMNLP*, 2004.

[23] H. Ney, “Speech translation: Coupling of recognition and translation,” in *ICASSP*, 1999.