Discriminative Reranking for Machine Translation

Libin Shen
Dept. of Comp. & Info. Science
Univ. of Pennsylvania
Philadelphia, PA 19104
libin@seas.upenn.edu

Anoop Sarkar
School of Comp. Science
Simon Fraser Univ.
Burnaby, BC V5A 1S6
anoop@cs.sfu.ca

Franz Josef Och
Info. Science Institute
Univ. of Southern California
Marina del Rey, CA 90292
och@isi.edu

Abstract

This paper describes the application of discriminative reranking techniques to the problem of machine translation. For each sentence in the source language, we obtain from a baseline statistical machine translation system, a ranked n-best list of candidate translations in the target language. We introduce two novel perceptron-inspired reranking algorithms that improve on the quality of machine translation over the baseline system based on evaluation using the BLEU metric. We provide experimental results on the NIST 2003 Chinese-English large data track evaluation. We also provide theoretical analysis of our algorithms and experiments that verify that our algorithms provide state-of-the-art performance in machine translation.

1 Introduction

The noisy-channel model (Brown et al., 1990) has been the foundation for statistical machine translation (SMT) for over ten years. Recently so-called reranking techniques, such as maximum entropy models (Och and Ney, 2002) and gradient methods (Och, 2003), have been applied to machine translation (MT), and have provided significant improvements. In this paper, we introduce two novel machine learning algorithms specialized for the MT task.

Discriminative reranking algorithms have also contributed to improvements in natural language parsing and tagging performance. Discriminative reranking algorithms used for these applications include Perceptron, Boosting and Support Vector Machines (SVMs). In the machine learning community, some novel discriminative ranking (also called ordinal regression) algorithms have been proposed in recent years. Based on this work, in this paper, we will present some novel discriminative reranking techniques applied to machine translation. The reranking problem for natural language is neither a classification problem nor a regression problem, and under certain conditions MT reranking turns out to be quite different from parse reranking.

In this paper, we consider the special issues of applying reranking techniques to the MT task and introduce two perceptron-like reranking algorithms for MT reranking. We provide experimental results that show that the proposed algorithms achieve start-of-the-art results on the NIST 2003 Chinese-English large data track evaluation.

1.1 Generative Models for MT

The seminal IBM models (Brown et al., 1990) were the first to introduce generative models to the MT task. The IBM models applied the sequence learning paradigm well-known from Hidden Markov Models in speech recognition to the problem of MT. The source and target sentences were treated as the observations, but the alignments were treated as hidden information learned from parallel texts using the EM algorithm. This source-channel model treated the task of finding the probability $p(\mathbf{e} \mid \mathbf{f})$, where $\mathbf{e}$ is the translation in the target (English) language for a given source (foreign) sentence $\mathbf{f}$, as two generative probability models: the language model $p(\mathbf{e})$ which is a generative probability over candidate translations and the translation model $p(\mathbf{f} \mid \mathbf{e})$ which is a generative conditional probability of the source sentence given a candidate translation $\mathbf{e}$.

The lexicon of the single-word based IBM models does not take word context into account. This means unlikely alignments are being considered while training the model and this also results in additional decoding complexity. Several MT models were proposed as extensions of the IBM models which used this intuition to add additional linguistic constraints to decrease the decoding perplexity and increase the translation quality.

Wang and Waibel (1998) proposed an SMT model based on phrase-based alignments. Since their translation model reordered phrases directly, it achieved higher accuracy for translation between languages with different word orders. In (Och and Weber, 1998; Och et al., 1999), a two-level alignment model was employed to utilize shallow phrase structures: alignment between templates was used to handle phrase reordering, and word alignments within a template were used to handle phrase to phrase translation.

However, phrase level alignment cannot handle long distance reordering effectively. Parse trees have also been used in alignment models. Wu (1997) introduced constraints on alignments using a probabilistic synchronous context-free grammar restricted to Chomsky-normal form. (Wu, 1997) was an implicit or self-organizing syntax model as it did not use a Treebank. Yamada and Knight (2001) used a statistical parser trained using a Treebank in the source language to produce parse trees and proposed a tree to string model for alignment. Gildea (2003) proposed a tree to tree alignment model using output from a statistical parser in both source and target languages. The translation model involved tree alignments in which subtree cloning was used to handle cases of reordering that were not possible in earlier tree-based alignment models.

1.2 Discriminative Models for MT

Och and Ney (2002) proposed a framework for MT based on direct translation, using the conditional model $p(\mathbf{e} \mid \mathbf{f})$ estimated using a maximum entropy model. A small number of feature functions defined on the source and target sentence were used to rerank the translations generated by a baseline MT system. While the total number of feature functions was small, each feature function was a complex statistical model by itself, as for example, the alignment template feature functions used in this approach.

Och (2003) described the use of minimum error training directly optimizing the error rate on automatic MT evaluation metrics such as BLEU. The experiments showed that this approach obtains significantly better results than using the maximum mutual information criterion on parameter estimation. This approach used the same set of features as the alignment template approach in (Och and Ney, 2002).

SMT Team (2003) also used minimum error training as in Och (2003), but used a large number of feature functions. More than 450 different feature functions were used in order to improve the syntactic well-formedness of MT output. By reranking a 1000-best list generated by the baseline MT system from Och (2003), the BLEU (Papineni et al., 2001) score on the test dataset was improved from 31.6% to 32.9%.

2 Ranking and Reranking

2.1 Reranking for NLP tasks

Like machine translation, parsing is another field of natural language processing in which generative models have been widely used. In recent years, reranking techniques, especially discriminative reranking, have resulted in significant improvements in parsing. Various machine learning algorithms have been employed in parse reranking, such as Boosting (Collins, 2000), Perceptron (Collins and Duffy, 2002) and Support Vector Machines (Shen and Joshi, 2003). The reranking techniques have resulted in a 13.5% error reduction in labeled recall/precision over the previous best generative parsing models. Discriminative reranking methods for parsing typically use the notion of a margin as the distance between the best candidate parse and the rest of the parses. The reranking problem is reduced to a classification problem by using pairwise samples.

In (Shen and Joshi, 2004), we have introduced a new perceptron-like ordinal regression algorithm for parse reranking. In that algorithm, pairwise samples are used for training and margins are defined as the distance between parses of different ranks. In addition, the uneven margin technique has been used for the purpose of adapting ordinal regression to reranking tasks. In this paper, we apply this algorithm to MT reranking, and we also introduce a new perceptron-like reranking algorithm for MT.

2.2 Ranking and Ordinal Regression

In the field of machine learning, a class of tasks (called *ranking* or *ordinal regression*) are similar to the reranking tasks in NLP. One of the motivations of this paper is to apply ranking or ordinal regression algorithms to MT reranking. In the previous works on ranking or ordinal regression, the margin is defined as the distance between two consecutive ranks. Two large margin approaches have been used. One is the PRank algorithm, a variant of the perceptron algorithm, that uses multiple biases to represent the boundaries between every two consecutive ranks (Crammer and Singer, 2001; Harrington, 2003). However, as we will show in section 3.7, the PRank algorithm does not work on the reranking tasks due to the introduction of *global* ranks. The other approach is to reduce the ranking problem to a classification problem by using the method of pairwise samples (Herbrich et al., 2000). The underlying assumption is that the samples of consecutive ranks are separable. This may become a problem in the case that ranks are unreliable when ranking does not strongly distinguish between candidates. This is just what happens in reranking for machine translation.

3 Discriminative Reranking for MT

The reranking approach for MT is defined as follows: First, a baseline system generates n-best candidates. Features that can potentially discriminate between good vs. bad translations are extracted from these n-best candidates. These features are then used to determine a new ranking for the n-best list. The new top ranked candidate in this n-best list is our new best candidate translation.

3.1 Advantages of Discriminative Reranking

Discriminative reranking allows us to use global features which are unavailable for the baseline system. Second, we can use features of various kinds and need not worry about fine-grained smoothing issues. Finally, the statistical machine learning approach has been shown to be effective in many NLP tasks. Reranking enables rapid experimentation with complex feature functions, because the complex decoding steps in SMT are done once to generate the N-best list of translations.

3.2 Problems applying reranking to MT

First, we consider how to apply discriminative reranking to machine translation. We may directly use those algorithms that have been successfully used in parse reranking. However, we immediately find that those algorithms are not as appropriate for machine translation. Let $\mathbf{e}_i$ be the candidate ranked at the ith position for the source sentence, where ranking is defined on the quality of the candidates. In parse reranking, we look for parallel hyperplanes successfully separating $\mathbf{e}_1$ and $\mathbf{e}_{2\ldots n}$ for all the source sentences, but in MT, for each source sentence, we have a set of reference translations instead of a single gold standard. For this reason, it is hard to define which candidate translation is the best. Suppose we have two translations, one of which is close to reference translation ref_a while the other is close to reference translation ref_b. It is difficult to say that one candidate is better than the other.

Although we might invent metrics to define the quality of a translation, standard reranking algorithms cannot be directly applied to MT. In parse reranking, each training sentence has a ranked list of 27 candidates on average (Collins, 2000), but for machine translation, the number of candidate translations in the n-best list is much higher. (SMT Team, 2003) show that to get a reasonable improvement in the BLEU score at least 1000 candidates need to be considered in the n-best list.

In addition, the parallel hyperplanes separating $\mathbf{e}_1$ and $\mathbf{e}_{2\ldots n}$ actually are unable to distinguish good translations from bad translations, since they are not trained to distinguish any translations in $\mathbf{e}_{2\ldots n}$. Furthermore, many good translations in $\mathbf{e}_{2\ldots n}$ may differ greatly from $\mathbf{e}_1$, since there are multiple references. These facts cause problems for the applicability of reranking algorithms.

3.3 Splitting

Our first attempt to handle this problem is to redefine the notion of good translations versus bad translations. Instead of separating $\mathbf{e}_1$ and $\mathbf{e}_{2\ldots n}$, we say the top r of the n-best translations are good translations, and the bottom k of the n-best translations are bad translations, where $r + k \leq n$. Then we look for parallel hyperplanes splitting the top r translations and bottom k translations for

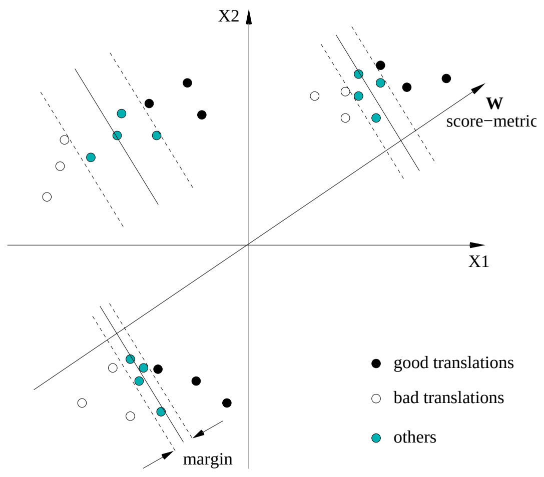

Figure 1: Splitting for MT Reranking

each sentence. Figure 1 illustrates this situation, where $n = 10, r = 3$ and $k = 3$.

3.4 Ordinal Regression

Furthermore, if we only look for the hyperplanes to separate the good and the bad translations, we, in fact, discard the order information of translations of the same class. Maybe knowing that $\mathbf{e}_{100}$ is better than $\mathbf{e}_{101}$ may be useless for training to some extent, but knowing $\mathbf{e}_2$ is better than $\mathbf{e}_{300}$ is useful, if $r = 300$. Although we cannot give an affirmative answer at this time, it is at least reasonable to use the ordering information. The problem is how to use the ordering information. In addition, we only want to maintain the order of two candidates if their ranks are far away from each other. On the other hand, we do not care the order of two translations whose ranks are very close, e.g. 100 and 101. Thus insensitive ordinal regression is more desirable and is the approach we follow in this paper.

3.5 Uneven Margins

However, reranking is not an ordinal regression problem. In reranking evaluation, we are only interested in the quality of the translation with the highest score, and we do not care the order of bad translations. Therefore we cannot simply regard a reranking problem as an ordinal regression problem, since they have different definitions for the loss function.

As far as linear classifiers are concerned, we want to maintain a larger margin in translations of high ranks and a smaller margin in translations of low ranks. For example,

$$\mathit{margin}(\mathbf{e}_1, \mathbf{e}_{30}) > \mathit{margin}(\mathbf{e}_1, \mathbf{e}_{10}) > \mathit{margin}(\mathbf{e}_{21}, \mathbf{e}_{30})$$

The reason is that the scoring function will be penalized if it can not separate $\mathbf{e}_1$ from $\mathbf{e}_{10}$, but not for the case of $\mathbf{e}_{21}$ versus $\mathbf{e}_{30}$.

3.6 Large Margin Classifiers

There are quite a few linear classifiers[1] that can separate samples with large margin, such as SVMs (Vapnik, 1998), Boosting (Schapire et al., 1997), Winnow (Zhang, 2000) and Perceptron (Krauth and Mezard, 1987). The performance of SVMs is superior to other linear classifiers because of their ability to margin maximization.

However, SVMs are extremely slow in training since they need to solve a quadratic programming search. For example, SVMs even cannot be used to train on the whole Penn Treebank in parse reranking (Shen and Joshi, 2003). Taking this into account, we use perceptron-like algorithms, since the perceptron algorithm is fast in training which allow us to do experiments on real-world data. Its large margin version is able to provide relatively good results in general.

3.7 Pairwise Samples

In previous work on the PRank algorithm, ranks are defined on the entire training and test data. Thus we can define boundaries between consecutive ranks on the entire data. But in MT reranking, ranks are defined over every single source sentence. For example, in our data set, the rank of a translation is only the rank among all the translations for the same sentence. The training data includes about 1000 sentences, each of which normally has 1000 candidate translations with the exception of short sentences that have a smaller number of candidate translations. As a result, we cannot use the PRank algorithm in the reranking task, since there are no global ranks or boundaries for all the samples.

However, the approach of using pairwise samples does work. By pairing up two samples, we compute the relative distance between these two samples in the scoring metric. In the training phase, we are only interested in whether the relative distance is positive or negative.

However, the size of generated training samples will be very large. For n samples, the total number of pairwise samples in (Herbrich et al., 2000) is roughly n^2. In the next section, we will introduce two perceptron-like algorithms that utilize pairwise samples while keeping the complexity of data space unchanged.

4 Reranking Algorithms

Considering the desiderata discussed in the last section, we present two perceptron-like algorithms for MT reranking. The first one is a splitting algorithm specially designed for MT reranking, which has similarities to a classification algorithm. We also experimented with an ordinal regression algorithm proposed in (Shen and Joshi, 2004). For the sake of completeness, we will briefly describe the algorithm here.

[1]Here we only consider linear kernels such as polynomial kernels.

4.1 Splitting

In this section, we will propose a splitting algorithm which separates translations of each sentence into two parts, the top r translations and the bottom k translations. All the separating hyperplanes are parallel by sharing the same weight vector $\mathbf{w}$. The margin is defined on the distance between the top r items and the bottom k items in each cluster, as shown in Figure 1.

Let $\mathbf{x}_{i,j}$ be the feature vector of the j^{th} translation of the i^{th} sentence, and $y_{i,j}$ be the rank for this translation among all the translations for the i^{th} sentence. Then the set of training samples is:

$$S = \{(\mathbf{x}_{i,j}, y_{i,j}) \mid 1 \le i \le m,\ 1 \le j \le n\},$$

where m is the number of clusters and n is the length of ranks for each cluster.

Let $f(\mathbf{x}) = \mathbf{w}_f \cdot \mathbf{x}$ be a linear function, where $\mathbf{x}$ is the feature vector of a translation, and $\mathbf{w}_f$ is a weight vector. We construct a hypothesis function $h_f : \mathcal{X} \to \mathcal{Y}$ with f as follows.

$$h_f(\mathbf{x}_1, ...\mathbf{x}_n) = rank(f(\mathbf{x}_1), ..., f(\mathbf{x}_n)),$$

where $rank$ is a function that takes a list of scores for the candidate translations computed according to the evaluation metric and returns the rank in that list. For example $rank(90, 40, 60) = (1, 3, 2)$.

The *splitting* algorithm searches a linear function $f(\mathbf{x}) = \mathbf{w}_f \cdot \mathbf{x}$ that successfully splits the top r-ranked and bottom k-ranked translations for each sentence, where $r + k \le n$. Formally, let $\mathbf{y}^f = (y_1^f, ..., y_n^f) = h_f(\mathbf{x}_1, ...\mathbf{x}_n)$ for any linear function f. We look for the function f such that

$$y_i^f \le r \quad \text{if} \quad y_i \le r \qquad (1)$$

$$y_i^f \ge n - k + 1 \quad \text{if} \quad y_i \ge n - k + 1, \qquad (2)$$

which means that f can successfully separate the good translations and the bad translations.

Suppose there exists a linear function f satisfying (1) and (2), we say $\{(\mathbf{x}_{i,j}, y_{i,j})\}$ is $splittable$ by f given n, r and k. Furthermore, we can define the *splitting margin* γ for the translations of the i^{th} sentence as follows.

$$\gamma(f, i) = \min_{j:y_{i,j} \le r} f(\mathbf{x}_{i,j}) - \max_{j:y_{i,j} \ge n-k+1} f(\mathbf{x}_{i,j})$$

The *minimal splitting margin*, γ^{split}, for f given n, r and k is defined as follows.

$$\begin{aligned}\gamma^{split}(f) &= \min_i \gamma(f, i) \\ &= \min_i \Big(\min_{y_{i,j} \le r} f(\mathbf{x}_{i,j}) - \max_{y_{i,j} \ge n-k+1} f(\mathbf{x}_{i,j}) \Big)\end{aligned}$$

Algorithm 1 splitting

```
Require: $r$, $k$, and a positive learning margin $\tau$.
 1: $t \leftarrow 0$, initialize $\mathbf{w}^0$;
 2: repeat
 3:   for ($i = 1, ..., m$) do
 4:     compute $\mathbf{w}^t \cdot \mathbf{x}_{i,j}$, $u_j \leftarrow 0$ for all $j$;
 5:     for ($1 \leq j < l \leq n$) do
 6:       if ($y_{i,j} \leq r$ and $y_{i,l} \geq n - k + 1$ and $\mathbf{w}^t \cdot \mathbf{x}_{i,j} < \mathbf{w}^t \cdot \mathbf{x}_{i,l} + \tau$) then
 7:         $u_j \leftarrow u_j + 1$; $u_l \leftarrow u_l - 1$;
 8:       else if ($y_{i,j} \geq n-k+1$ and $y_{i,l} \leq r$ and $\mathbf{w}^t \cdot \mathbf{x}_{i,j} > \mathbf{w}^t \cdot \mathbf{x}_{i,l} - \tau$) then
 9:         $u_j \leftarrow u_j - 1$; $u_l \leftarrow u_l + 1$;
10:       end if
11:     end for
12:     $\mathbf{w}^{t+1} \leftarrow \mathbf{w}^t + \sum_j u_j \mathbf{x}_{i,j}$; $t \leftarrow t + 1$;
13:   end for
14: until no updates made in the outer for loop
```

Algorithm 1 is a perceptron-like algorithm that looks for a function that splits the training data. The idea of the algorithm is as follows. For every two translations $\mathbf{x}_{i,j}$ and $\mathbf{x}_{i,l}$, if

- the rank of $\mathbf{x}_{i,j}$ is higher than or equal to r, $y_{i,j} \leq r$,
- the rank of $\mathbf{x}_{i,l}$ is lower than r, $y_{i,l} \geq n - k + 1$,
- the weight vector $\mathbf{w}$ can not successfully separate ($\mathbf{x}_{i,j}$ and $\mathbf{x}_{i,l}$) with a learning margin τ, $\mathbf{w} \cdot \mathbf{x}_{i,j} < \mathbf{w} \cdot \mathbf{x}_{i,l} + \tau$,

then we need to update $\mathbf{w}$ with the addition of $\mathbf{x}_{i,j} - \mathbf{x}_{i,l}$. However, the updating is not executed until all the inconsistent pairs in a sentence are found for the purpose of speeding up the algorithm. When sentence i is selected, we first compute and store $\mathbf{w}^t \cdot \mathbf{x}_{i,j}$ for all j. Thus we do not need to recompute $\mathbf{w}^t \cdot \mathbf{x}_{i,j}$ again in the inner loop. Now the complexity of a **repeat** iteration is $O(mn^2 + mnd)$, where d is the average number of active features in vector $\mathbf{x}_{i,j}$. If we updated the weight vector whenever an inconsistent pair was found, the complexity of a loop would be $O(mn^2 d)$.

The following theorem will show that Algorithm 1 will stop in finite steps, outputting a function that splits the training data with a large margin, if the training data is splittable. Due to lack of space, we omit the proof for Theorem 1 in this paper.

Theorem 1 *Suppose the training samples $\{(\mathbf{x}_{i,j}, y_{i,j})\}$ are splittable by a linear function defined on the weight vector $\mathbf{w}^*$ with a splitting margin γ, where $||\mathbf{w}^*|| = 1$. Let $R = max_{i,j}||\mathbf{x}_{i,j}||$. Then Algorithm 1 makes at most $\frac{n^2R^2+2\tau}{\gamma^2}$ mistakes on the pairwise samples during the training.*

Algorithm 2 ordinal regression with uneven margin

```
Require: a positive learning margin $\tau$.
 1: $t \leftarrow 0$, initialize $\mathbf{w}^0$;
 2: repeat
 3:   for (sentence $i = 1, ..., m$) do
 4:     compute $\mathbf{w}^t \cdot \mathbf{x}_{i,j}$ and $u_j \leftarrow 0$ for all $j$;
 5:     for ($1 \leq j < l \leq n$) do
 6:       if ($y_{i,j} < y_{i,l}$ and $dis(y_{i,j}, y_{i,l}) > \epsilon$ and $\mathbf{w}^t \cdot \mathbf{x}_{i,j} - \mathbf{w}^t \cdot \mathbf{x}_{i,l} < g(y_{i,j}, y_{i,l})\tau$) then
 7:         $u_j \leftarrow u_j + g(y_{i,j}, y_{i,l})$;
 8:         $u_l \leftarrow u_l - g(y_{i,j}, y_{i,l})$;
 9:       else if ($y_{i,j} > y_{i,l}$ and $dis(y_{i,j}, y_{i,l}) > \epsilon$ and $\mathbf{w}^t \cdot \mathbf{x}_{i,l} - \mathbf{w}^t \cdot \mathbf{x}_{i,j} < g(y_{i,l}, y_{i,j})\tau$) then
10:         $u_j \leftarrow u_j - g(y_{i,l}, y_{i,j})$;
11:         $u_l \leftarrow u_l + g(y_{i,l}, y_{i,j})$;
12:       end if
13:     end for
14:     $\mathbf{w}^{t+1} \leftarrow \mathbf{w}^t + \sum_j u_j \mathbf{x}_{i,j}$; $t \leftarrow t + 1$;
15:   end for
16: until no updates made in the outer for loop
```

4.2 Ordinal Regression

The second algorithm that we will use for MT reranking is the ϵ-insensitive ordinal regression with uneven margin, which was proposed in (Shen and Joshi, 2004), as shown in **Algorithm 2**.

In Algorithm 2, the function dis is used to control the level of insensitivity, and the function g is used to control the learning margin between pairs of translations with different ranks as described in Section 3.5. There are many candidates for g. The following definition for g is one of the simplest solutions.

$$g(p, q) \equiv \frac{1}{p} - \frac{1}{q}$$

We will use this function in our experiments on MT reranking.

5 Experiments and Analysis

We provide experimental results on the NIST 2003 Chinese-English large data track evaluation. We use the data set used in (SMT Team, 2003). The training data consists of about 170M English words, on which the baseline translation system is trained. The training data is also used to build language models which are used to define feature functions on various syntactic levels. The development data consists of 993 Chinese sentences. Each Chinese sentence is associated with 1000-best English translations generated by the baseline MT system. The development data set is used to estimate the parameters for the feature functions for the purpose of reranking. The

Table 1: BLEU scores reported in (SMT Team, 2003). Every single feature was combined with the 6 baseline features for the training and test. The minimum error training (Och, 2003) was used on the development data for parameter estimation.

Feature	BLEU%
Baseline	31.6
POS Language Model	31.7
Supertag Language Model	31.7
Wrong NN Position	31.7
Word Popularity	31.8
Aligned Template Models	31.9
Count of Missing Word	31.9
Template Right Continuity	32.0
IBM Model 1	**32.5**

test data consists of 878 Chinese sentences. Each Chinese sentence is associated with 1000-best English translations too. The test set is used to assess the quality of the reranking output.

In (SMT Team, 2003), 450 features were generated. Six features from (Och, 2003) were used as baseline features. Each of the 450 features was evaluated independently by combining it with 6 baseline features and assessing on the test data with the minimum error training. The baseline BLEU score on the test set is 31.6%. Table 1 shows some of the best performing features.

In (SMT Team, 2003), aggressive search was used to combine features. After combining about a dozen features, the BLEU score did not improve any more, and the score was 32.9%. It was also noticed that the major improvement came from the Model 1 feature. By combining the four features, Model 1, matched parentheses, matched quotation marks and POS language model, the system achieved a BLEU score of 32.6%.

In our experiments, we will use 4 different kinds of feature combinations:

- **Baseline**: The 6 baseline features used in (Och, 2003), such as cost of word penalty, cost of aligned template penalty.
- **Best Feature**: Baseline + IBM Model 1 + matched parentheses + matched quotation marks + POS language model.
- **Top Twenty**: Baseline + 14 features with individual BLEU score no less than 31.9% with the minimum error training.
- **Large Set**: Baseline + 50 features with individual BLEU score no less than 31.7% with the minimum error training. Since the baseline is 31.6% and the 95% confidence range is ± 0.9%, most of the features in this set are not individually discriminative with respect to the BLEU metric.

We apply Algorithm 1 and 2 to the four feature sets. For algorithm 1, the splitting algorithm, we set $k = 300$ in the 1000-best translations given by the baseline MT system. For algorithm 2, the ordinal regression algorithm, we set the updating condition as $y_{i,j} * 2 < y_{i,l}$ and $y_{i,j} + 20 < y_{i,l}$, which means one's rank number is at most half of the other's and there are at least 20 ranks in between. Figures 2-9 show the results of using Algorithm 1 and 2 with the four feature sets. The x-axis represents the number of iterations in the training. The left y-axis stands for the BLEU% score on the test data, and the right y-axis stands for log of the loss function on the development data.

Algorithm 1, the splitting algorithm, converges on the first three feature sets. The smaller the feature set is, the faster the algorithm converges. It achieves a BLEU score of 31.7% on the Baseline, 32.8% on the Best Feature, but only 32.6% on the Top Twenty features. However it is within the range of 95% confidence. Unfortunately on the Large Set, Algorithm 1 converges very slowly.

In the Top Twenty set there are a fewer number of individually non-discriminative feature making the pool of features "better". In addition, generalization performance in the Top Twenty set is better than the Large Set due to the smaller set of "better" features, cf. (Shen and Joshi, 2004). If the number of the non-discriminative features is large enough, the data set becomes unsplittable. We have tried using the λ trick as in (Li et al., 2002) to make data separable artificially, but the performance could not be improved with such features.

We achieve similar results with Algorithm 2, the ordinal regression with uneven margin. It converges on the first 3 feature sets too. On the Baseline, it achieves 31.4%. We notice that the model is over-trained on the development data according to the learning curve. In the Best Feature category, it achieves 32.7%, and on the Top Twenty features, it achieves 32.9%. This algorithm does not converge on the Large Set in 10000 iterations.

We compare our perceptron-like algorithms with the minimum error training used in (SMT Team, 2003) as shown in Table 2. The splitting algorithm achieves slightly better results on the Baseline and the Best Feature set, while the minimum error training and the regression algorithm tie for first place on feature combinations. However, the differences are not significant.

We notice in those separable feature sets the performance on the development data and the test data are tightly consistent. Whenever the log-loss on the development set is decreased, and BLEU score on the test set goes up, and vice versa. This tells us the merit of these two algorithms; By optimizing on the loss function for

Table 2: Comparison between the minimum error training with discriminative reranking on the test data (BLEU%)

Algorithm	Baseline	Best Feat	Feat Comb
Minimum Error	31.6	32.6	**32.9**
Splitting	**31.7**	**32.8**	32.6
Regression	31.4	32.7	**32.9**

the development data, we can improve performance on the test data. This property is guaranteed by the theoretical analysis and is borne out in the experimental results.

6 Conclusions and Future Work

In this paper, we have successfully applied the discriminative reranking to machine translation. We applied a new perceptron-like splitting algorithm and ordinal regression algorithm with uneven margin to reranking in MT. We provide a theoretical justification for the performance of the splitting algorithms. Experimental results provided in this paper show that the proposed algorithms provide state-of-the-art performance in the NIST 2003 Chinese-English large data track evaluation.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. 0121285. The first author was partially supported by JHU postworkshop fellowship and NSF Grant ITR-0205456. The second author is partially supported by NSERC, Canada (RGPIN: 264905). We thank the members of the SMT team of JHU Workshop 2003 for help on the dataset and three anonymous reviewers for useful comments.

References

P. F. Brown, J. Cocke, S. A. Della Pietra, V. J. Della Pietra, F. Jelinek, J. D. Lafferty, R. L. Mercer, and P. S. Roossin. 1990. A statistical approach to machine translation. *Computational Linguistics*, 16(2):79–85.

M. Collins and N. Duffy. 2002. New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In *Proceedings of ACL 2002*.

M. Collins. 2000. Discriminative reranking for natural language parsing. In *Proceedings of the 7th ICML*.

K. Crammer and Y. Singer. 2001. PRanking with Ranking. In *NIPS 2001*.

D. Gildea. 2003. Loosely tree-based alignment for machine translation. In *ACL 2003*.

E. F. Harrington. 2003. Online Ranking/Collaborative Filtering Using the Perceptron Algorithm. In *ICML*.

R. Herbrich, T. Graepel, and K. Obermayer. 2000. Large margin rank boundaries for ordinal regression. In A.J. Smola, P. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 115–132. MIT Press.

W. Krauth and M. Mezard. 1987. Learning algorithms with optimal stability in neural networks. *Journal of Physics A*, 20:745–752.

Y. Li, H. Zaragoza, R. Herbrich, J. Shawe-Taylor, and J. Kandola. 2002. The perceptron algorithm with uneven margins. In *Proceedings of ICML 2002*.

F. J. Och and H. Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *ACL 2002*.

F. J. Och and H. Weber. 1998. Improving statistical natural language translation with categories and rules. In *COLING-ACL 1998*.

F. J. Och, C. Tillmann, and H. Ney. 1999. Improved alignment models for statistical machine. In *EMNLP-WVLC 1999*.

F. J. Och. 2003. Minimum error rate training for statistical machine translation. In *ACL 2003*.

K. Papineni, S. Roukos, and T. Ward. 2001. Bleu: a method for automatic evaluation of machine translation. IBM Research Report, RC22176.

R. E. Schapire, Y. Freund, P. Bartlett, and W. S. Lee. 1997. Boosting the margin: a new explanation for the effectiveness of voting methods. In *Proc. 14th ICML*.

L. Shen and A. K. Joshi. 2003. An SVM based voting algorithm with application to parse reranking. In *Proc. of CoNLL 2003*.

L. Shen and A. K. Joshi. 2004. Flexible margin selection for reranking with full pairwise samples. In *Proc. of 1st IJCNLP*.

SMT Team. 2003. Final report: Syntax for statistical machine translation. JHU Summer Workshop 2003, http://www.clsp.jhu.edu/ws2003/groups/translate.

V. N. Vapnik. 1998. *Statistical Learning Theory*. John Wiley and Sons, Inc.

Y. Wang and A. Waibel. 1998. Modeling with structures in statistical machine translation. In *COLING-ACL 1998*.

D. Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–400.

K. Yamada and K. Knight. 2001. A syntax-based statistical translation model. In *ACL 2001*.

T. Zhang. 2000. Large Margin Winnow Methods for Text Categorization. In *KDD-2000 Workshop on Text Mining*.

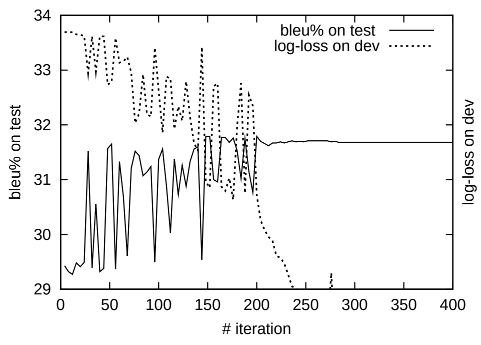

Figure 2: Splitting on Baseline

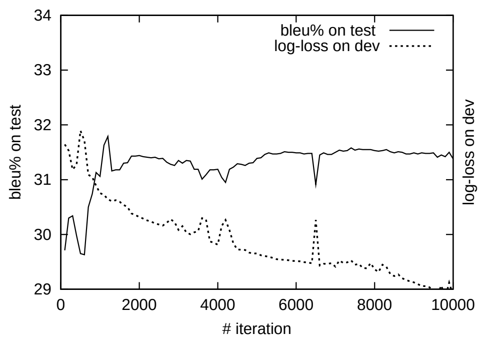

Figure 6: Ordinal Regression on Baseline

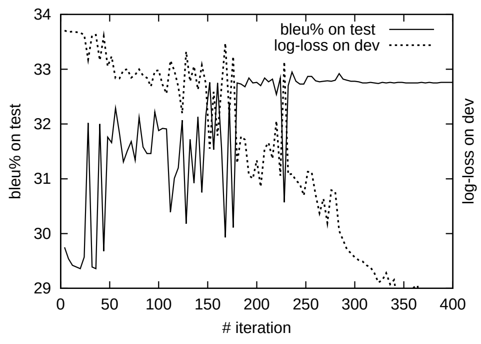

Figure 3: Splitting on Best Feature

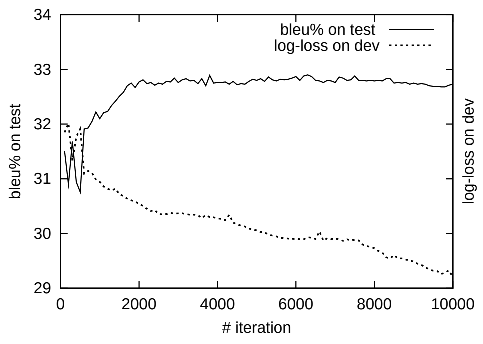

Figure 7: Ordinal Regression on Best Feature

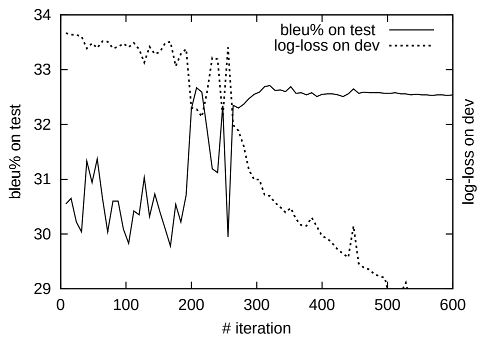

Figure 4: Splitting on Top Twenty

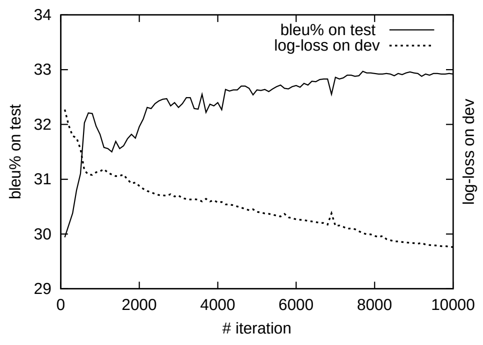

Figure 8: Ordinal Regression on Top Twenty

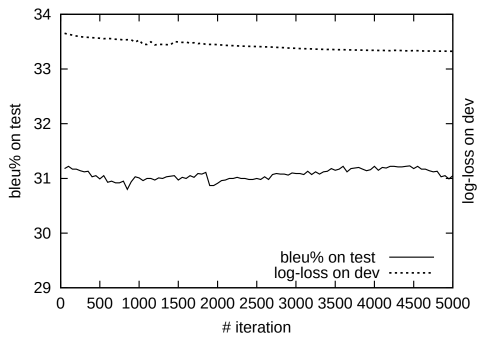

Figure 5: Splitting on Large Set

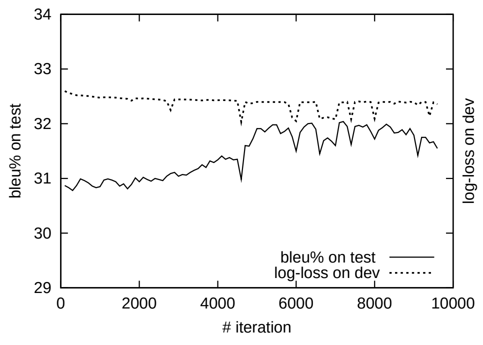

Figure 9: Ordinal Regression on Large Set