Minimum Sample Risk Methods for Language Modeling[1]

Jianfeng Gao
Microsoft Research Asia
jfgao@microsoft.com

Hao Yu, Wei Yuan
Shanghai Jiaotong Univ., China

Peng Xu
John Hopkins Univ., U.S.A.
xp@clsp.jhu.edu

Abstract

This paper proposes a new discriminative training method, called *minimum sample risk* (MSR), of estimating parameters of language models for text input. While most existing discriminative training methods use a loss function that can be optimized easily but approaches only approximately to the objective of minimum error rate, MSR minimizes the training error directly using a heuristic training procedure. Evaluations on the task of Japanese text input show that MSR can handle a large number of features and training samples; it significantly outperforms a regular trigram model trained using maximum likelihood estimation, and it also outperforms the two widely applied discriminative methods, the boosting and the perceptron algorithms, by a small but statistically significant margin.

1 Introduction

Language modeling (LM) is fundamental to a wide range of applications, such as speech recognition and Asian language text input (Jelinek 1997; Gao et al. 2002). The traditional approach uses a parametric model with *maximum likelihood estimation* (MLE), usually with smoothing methods to deal with data sparseness problems. This approach is optimal under the assumption that the true distribution of data on which the parametric model is based is known. Unfortunately, such an assumption rarely holds in realistic applications.

An alternative approach to LM is based on the framework of discriminative training, which uses a much weaker assumption that training and test data are generated from the same distribution but the form of the distribution is unknown. Unlike the traditional approach that maximizes the function (i.e. likelihood of training data) that is loosely associated with error rate, discriminative training methods aim to directly minimize the error rate on training data even if they reduce the likelihood. So, they potentially lead to better solutions. However, the error rate of a finite set of training samples is usually a step function of model parameters, and cannot be easily minimized. To address this problem, previous research has concentrated on the development of a loss function that approximates the exact error rate and can be easily optimized. Though these methods (e.g. the boosting method) have theoretically appealing properties, such as convergence and bounded generalization error, we argue that the approximated loss function may prevent them from attaining the original objective of minimizing the error rate.

In this paper we present a new estimation procedure for LM, called *minimum sample risk* (MSR). It differs from most existing discriminative training methods in that instead of searching on an approximated loss function, MSR employs a simple heuristic training algorithm that minimizes the error rate on training samples directly. MSR operates like a multidimensional function optimization algorithm: first, it selects a subset of features that are the most effective among all candidate features. The parameters of the model are then optimized iteratively: in each iteration, only the parameter of one feature is adjusted. Both feature selection and parameter optimization are based on the criterion of minimizing the error on training samples. Our evaluation on the task of Japanese text input shows that MSR achieves more than 20% error rate reduction over MLE on two newswire data sets, and it also outperforms the other two widely applied discriminative methods, the boosting method and the perceptron algorithm, by a small but statistically significant margin.

Although it has not been proved in theory that MSR is always *robust*, our experiments of cross-domain LM adaptation show that it is. MSR can effectively adapt a model trained on one domain to

[1] The work was done while the second, third and fourth authors were visiting Microsoft Research Asia. Thanks to Hisami Suzuki for her valuable comments.

different domains. It outperforms the traditional LM adaptation method significantly, and achieves at least comparable or slightly better results to the boosting method and the perceptron algorithm.

2 IME Task and LM

This paper studies LM on the task of Asian language (e.g. Chinese or Japanese) text input. This is the standard method of inputting Chinese or Japanese text by converting the input phonetic symbols into the appropriate word string. In this paper we call the task IME, which stands for *input method editor*, based on the name of the commonly used Windows-based application.

Performance on IME is measured in terms of the character error rate (CER), which is the number of characters wrongly converted from the phonetic string divided by the number of characters in the correct transcript. Current IME systems make about 5-15% CER in conversion of real data in a wide variety of domains (e.g. Gao et al. 2002).

Similar to speech recognition, IME is viewed as a Bayes decision problem. Let A be the input phonetic string. An IME system's task is to choose the most likely word string W^* among those candidates that could be converted from A:

$$W^* = \arg\max_{W \in \mathbf{GEN}(A)} P(W \mid A) = \arg\max_{W \in \mathbf{GEN}(A)} P(W)P(A \mid W) \quad (1)$$

where $\mathbf{GEN}(A)$ denotes the candidate set given A.

Unlike speech recognition, however, there is no acoustic ambiguity since the phonetic string is inputted by users. Moreover, if we do not take into account typing errors, it is reasonable to assume a unique mapping from W and A in IME, i.e. $P(A \mid W) = 1$. So the decision of Equation (1) depends solely upon $P(W)$, making IME a more *direct* evaluation test bed for LM than speech recognition. Another advantage is that it is easy to convert W to A (for Chinese and Japanese), which enables us to obtain a large number of training data for discriminative learning, as described later.

The values of $P(W)$ in Equation (1) are traditionally calculated by MLE: the optimal model parameters $\boldsymbol{\lambda}^*$ are chosen in such a way that $P(W|\boldsymbol{\lambda}^*)$ is maximized on training data. The arguments in favor of MLE are based on the assumption that the form of the underlying distributions is known, and that only the values of the parameters characterizing those distributions are unknown. In using MLE for LM, one always assumes a multinomial distribution of language. For example, a trigram model makes the assumption that the next word is predicted depending only on two preceding words. However, there are many cases in natural language where words over an arbitrary distance can be related. MLE is therefore not optimal because the assumed model form is incorrect.

What are the best estimators when the model is known to be false then? In IME, we can tackle this question empirically. Best IME systems achieve the least CER. Therefore, the best estimators are those which minimize the expected error rate on unseen test data. Since the distribution of test data is unknown, we can approximately minimize the error rate on some given training data (Vapnik 1999). Toward this end, we have developed a very simple heuristic training procedure called *minimum sample risk*, as presented in the next section.

3 Minimum Sample Risk

3.1 Problem Definition

We follow the general framework of linear discriminant models described in (Duda et al. 2001). In the rest of the paper we use the following notation, adapted from Collins (2002).

- Training data is a set of example input/output pairs. In LM for IME, training samples are represented as $\{A_i, W_i^R\}$, for $i = 1 \ldots M$, where each A_i is an input phonetic string and W_i^R is the reference transcript of A_i.
- We assume some way of generating a set of candidate word strings given A, denoted by $\mathbf{GEN}(A)$. In our experiments, $\mathbf{GEN}(A)$ consists of top N word strings converted from A using a baseline IME system that uses only a word trigram model.
- We assume a set of D+1 features $f_d(W)$, for $d = 0 \ldots D$. The features could be arbitrary functions that map W to real values. Using vector notation, we have $\mathbf{f}(W) \in \Re^{D+1}$, where $\mathbf{f}(W) = [f_0(W), f_1(W), \ldots, f_D(W)]^T$. Without loss of generality, $f_0(W)$ is called the base feature, and is defined in our case as the log probability that the word trigram model assigns to W. Other features ($f_d(W)$, for $d = 1 \ldots D$) are defined as the counts of word n-grams (n = 1 and 2 in our experiments) in W.
- Finally, the parameters of the model form a vector of D+1 dimensions, each for one feature function, $\boldsymbol{\lambda} = [\lambda_0, \lambda_1, \ldots, \lambda_D]$. The score of a word string W can be written as

$$Score(W,\boldsymbol{\lambda}) = \boldsymbol{\lambda}\mathbf{f}(W) = \sum_{d=0}^{D} \lambda_d f_d(W). \quad (2)$$

The decision rule of Equation (1) is rewritten as

$$W^*(A,\boldsymbol{\lambda}) = \arg\max_{W \in \mathbf{GEN}(A)} Score(W,\boldsymbol{\lambda}). \quad (3)$$

Equation (3) views IME as a ranking problem, where the model gives the ranking score, not probabilities. We therefore do not evaluate the model via perplexity.

Now, assume that we can measure the number of conversion errors in W by comparing it with a reference transcript W^R using an error function $\mathrm{Er}(W^R,W)$ (i.e. the string edit distance function in our case). We call the sum of error counts over the training samples *sample risk*. Our goal is to minimize the sample risk while searching for the parameters as defined in Equation (4), hence the name *minimum sample risk* (MSR). W_i^* in Equation (4) is determined by Equation (3),

$$\boldsymbol{\lambda}_{MSR} \stackrel{def}{=} \arg\min_{\boldsymbol{\lambda}} \sum_{i=1\ldots M} \mathrm{Er}(W_i^R, W_i^*(A_i,\boldsymbol{\lambda})). \quad (4)$$

We first present the basic MSR training algorithm, and then the two improvements we made.

3.2 Training Algorithm

The MSR training algorithm is cast as a multidimensional function optimization approach (Press et al. 1992): taking the feature vector as a set of directions; the first direction (i.e. feature) is selected and the objective function (i.e. sample risk) is minimized along that direction using a *line search*; then from there along the second direction to its minimum, and so on, cycling through the whole set of directions as many times as necessary, until the objective function stops decreasing.

This simple method can work properly under two assumptions. First, there exists an implementation of line search that optimizes the function along one direction efficiently. Second, the number of candidate features is not too large, and these features are not highly correlated. However, neither of the assumptions holds in our case. First of all, Er(.) in Equation (4) is a step function of $\boldsymbol{\lambda}$, thus cannot be optimized directly by regular gradient-based procedures – a grid search has to be used instead. However, there are problems with simple grid search: using a large grid could miss the optimal solution whereas using a fine-grained grid would lead to a very slow algorithm. Secondly, in the case of LM, there are millions of candidate features, some of which are highly correlated. We address these issues respectively in the next two subsections.

3.3 Grid Line Search

Our implementation of a grid search is a modified version of that proposed in (Och 2003). The modifications are made to deal with the efficiency issue due to the fact that there is a very large number of features and training samples in our task, compared to only 8 features used in (Och 2003). Unlike a simple grid search where the intervals between any two adjacent grids are equal and fixed, we determine for each feature a sequence of grids with differently sized intervals, each corresponding to a different value of sample risk.

As shown in Equation (4), the loss function (i.e. sample risk) over all training samples is the sum of the loss function (i.e. Er(.)) of each training sample. Therefore, in what follows, we begin with a discussion on minimizing Er(.) of a training sample using the line search.

Let $\boldsymbol{\lambda}$ be the current model parameter vector, and f_d be the selected feature. The line search aims to find the optimal parameter λ_d^* so as to minimize Er(.). For a training sample (A, W^R), the score of each candidate word string $W \in \mathbf{GEN}(A)$, as in Equation (2), can be decomposed into two terms:

$$Score(W,\boldsymbol{\lambda}) = \boldsymbol{\lambda}\mathbf{f}(W) = \sum_{d'=0 \vee d' \neq d}^{D} \lambda_{d'} f_{d'}(W) + \lambda_d f_d(W),$$

where the first term on the right hand side does not change with λ_d. Note that if several candidate word strings have the same feature value $f_d(W)$, their relative rank will remain the same for any λ_d. Since $f_d(W)$ takes integer values in our case ($f_d(W)$ is the count of a particular n-gram in W), we can group the candidates using $f_d(W)$ so that candidates in each group have the same value of $f_d(W)$. In each group, we define the candidate with the highest value of

$$\sum_{d'=0 \vee d' \neq d}^{D} \lambda_{d'} f_{d'}(W)$$

as the *active* candidate of the group because no matter what value λ_d takes, only this candidate could be selected according to Equation (3).

Now, we reduce $\mathbf{GEN}(A)$ to a much smaller list of active candidates. We can find a set of intervals for λ_d, within each of which a particular active candidate will be selected as W^*. We can compute the Er(.) value of that candidate as the Er(.) value for the corresponding interval. As a result, for each

training sample, we obtain a sequence of intervals and their corresponding Er(.) values. The optimal value λ_d^* can then be found by traversing the sequence and taking the midpoint of the interval with the lowest Er(.) value.

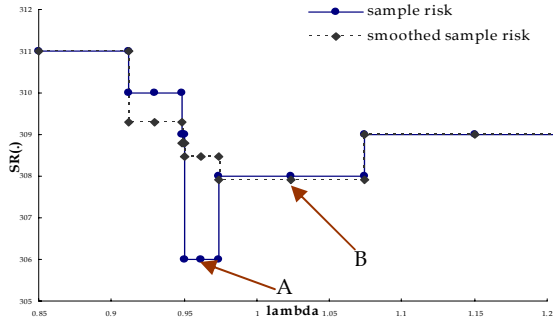

Figure 1. Examples of line search.

This process can be extended to the whole training set as follows. By merging the sequence of intervals of each training sample in the training set, we obtain a global sequence of intervals as well as their corresponding sample risk. We can then find the optimal value λ_d^* as well as the minimal sample risk by traversing the global interval sequence. An example is shown in Figure 1.

The line search can be unstable, however. In some cases when some of the intervals are very narrow (e.g. the interval A in Figure 1), moving the optimal value λ_d^* slightly can lead to much larger sample risk. Intuitively, we prefer a stable solution which is also known as a robust solution (with even slightly higher sample risk, e.g. the interval B in Figure 1). Following Quirk et al. (2004), we evaluate each interval in the sequence by its corresponding *smoothed sample risk*. Let λ be the midpoint of an interval and SR(λ) be the corresponding sample risk of the interval. The smoothed sample risk of the interval is defined as

$$\int_{\lambda-b}^{\lambda+b} \mathrm{SR}(\lambda)\, d\lambda$$

where b is a smoothing factor whose value is determined empirically (0.06 in our experiments). As shown in Figure 1, a more stable interval B is selected according to the smoothed sample risk.

In addition to reducing **GEN**(A) to an active candidate list described above, the efficiency of the line search can be further improved. We find that the line search only needs to traverse a small subset of training samples because the distribution of features among training samples are very sparse. Therefore, we built an inverted index that lists for each feature all training samples that contain it. As will be shown in Section 4.2, the line search is very efficient even for a large training set with millions of candidate features.

3.4 Feature Subset Selection

This section describes our method of selecting among millions of features a small subset of highly effective features for MSR learning. Reducing the number of features is essential for two reasons: to reduce computational complexity and to ensure the generalization property of the linear model. A large number of features lead to a large number of parameters of the resulting linear model, as described in Section 3.1. For a limited number of training samples, keeping the number of features sufficiently small should lead to a simpler model that is less likely to overfit to the training data.

The first step of our feature selection algorithm treats the features *independently*. The effectiveness of a feature is measured in terms of the reduction of the sample risk on top of the base feature f_0. Formally, let SR(f_0) be the sample risk of using the base feature only, and SR($f_0 + \lambda_d f_d$) be the sample risk of using both f_0 and f_d and the parameter λ_d that has been optimized using the line search. Then the effectiveness of f_d, denoted by $E(f_d)$, is given by

$$E(f_d) = \frac{\mathrm{SR}(f_0) - \mathrm{SR}(f_0 + \lambda_d f_d)}{\max_{f_i, i=1\ldots D} (\mathrm{SR}(f_0) - \mathrm{SR}(f_0 + \lambda_i f_i))}, \qquad (5)$$

where the denominator is a normalization term to ensure that $E(f) \in [0, 1]$.

The feature selection procedure can be stated as follows: The value of $E(.)$ is computed according to Equation (5) for each of the candidate features. Features are then ranked in the order of descending values of $E(.)$. The top l features are selected to form the feature vector in the linear model.

Treating features independently has the advantage of computational simplicity, but may not be effective for features with high correlation. For instance, although two features may carry rich discriminative information when treated separately, there may be very little gain if they are combined in a feature vector, because of the high correlation between them. Therefore, in what follows, we describe a technique of incorporating correlation information in the feature selection criterion.

Let x_{md}, $m = 1\ldots M$ and $d = 1\ldots D$, be a Boolean value: $x_{md} = 1$ if the sample risk reduction of using the d-th feature on the m-th training sample, com-

puted by Equation (5), is larger than zero, and 0 otherwise. The cross correlation coefficient between two features f_i and f_j is estimated as

$$C(i,j) = \frac{\sum_{m=1}^{M} x_{mi} x_{mj}}{\sqrt{\sum_{m=1}^{M} x_{mi}^2 \sum_{m=1}^{M} x_{mj}^2}}. \quad (6)$$

It can be shown that $C(i, j) \in [0, 1]$. Now, similar to (Theodoridis and Koutroumbas 2003), the feature selection procedure consists of the following steps, where f_i denotes any selected feature and f_j denotes any candidate feature to be selected.

Step 1. For each of the candidate features (f_d, for d = 1…D), compute the value of $E(f)$ according to Equation (5). Rank them in a descending order and choose the one with the highest $E(.)$ value. Let us denote this feature as f_1.

Step 2. To select the second feature, compute the cross correlation coefficient between the selected feature f_1 and each of the remaining M-1 features, according to Equation (6).

Step 3. Select the second feature f according to

$$j^* = \arg\max_{j=2...D} \{\alpha E(f_j) - (1-\alpha) C(1,j)\}$$

where α is the weight that determines the relative importance we give to the two terms. The value of α is optimized on held-out data (0.8 in our experiments). This means that for the selection of the second feature, we take into account not only its impact of reducing the sample risk but also the correlation with the previously selected feature. It is expected that choosing features with less correlation gives better sample risk minimization.

Step 4. Select k-th features, k = 3…K, according to

$$j^* = \arg\max_{j} \left\{ \alpha E(f_j) - \frac{1-\alpha}{k-1} \sum_{i=1}^{k-1} C(i,j) \right\} \quad (7)$$

That is, we select the next feature by taking into account its average correlation with all previously selected features. The optimal number of features, l, is determined on held-out data.

Similarly to the case of line search, we need to deal with the efficiency issue in the feature selection method. As shown in Equation (7), the estimates of $E(.)$ and $C(.)$ need to be computed. Let D and K ($K << D$) be the number of all candidate features and the number of features in the resulting model, respectively. According to the feature selection method described above, we need to estimate $E(.)$ for each of the D candidate features only once in Step 1. This is not very costly due to the efficiency of our line search algorithm. Unlike the case of $E(.)$, $O(K \times D)$ estimates of $C(.)$ are required in Step 4. This is computationally expensive even for a medium-sized K. Therefore, every time a new feature is selected (in Step 4), we only estimate the value of $C(.)$ between each of the selected features and each of the top N remaining features with the highest value of $E(.)$. This reduces the number of estimates of $C(.)$ to $O(K \times N)$. In our experiments we set N = 1000, much smaller than D. This reduces the computational cost significantly without producing any noticeable quality loss in the resulting model.

The MSR algorithm used in our experiments is summarized in Figure 2. It consists of feature selection (line 2) and optimization (lines 3 - 5) steps.

```
1  Set λ0 = 1 and λd = 0 for d=1…D
2  Rank all features and select the top K features, using
   the feature selection method described in Section 3.4
3  For t = 1…T (T= total number of iterations)
4      For each k = 1…K
5          Update the parameter of fk using line search.
```

Figure 2: The MSR algorithm

4 Evaluation

4.1 Settings

We evaluated MSR on the task of Japanese IME. Two newspaper corpora are used as training and test data: Nikkei and Yomiuri Newspapers. Both corpora have been pre-word-segmented using a lexicon containing 167,107 entries. A 5,000-sentence subset of the Yomiuri Newspaper corpus was used as held-out data (e.g. to determine learning rate, number of iterations and features etc.). We tested our models on another 5,000-sentence subset of the Yomiuri Newspaper corpus.

We used an 80,000-sentence subset of the Nikkei Newspaper corpus as the training set. For each A, we produced a word lattice using the baseline system described in (Gao et al. 2002), which uses a word trigram model trained via MLE on anther 400,000-sentence subset of the Nikkei Newspaper corpus. The two subsets do not overlap so as to simulate the case where unseen phonetic symbol strings are converted by the baseline system. For efficiency, we kept for each training sample the best 20 hypotheses in its candidate conversion set **GEN**(A) for discriminative training. The oracle best hypothesis, which gives the minimum number of errors, was used as the reference transcript of A.

	Model	CER (%)	% over MLE
1.	**MLE**	3.70	--
2.	**MSR (*K*=2000)**	**2.95**	**20.9**
3.	**Boosting**	3.06	18.0
4.	**Perceptron**	3.07	17.8

Table 1. Comparison of CER results.

4.2 Results

We used unigrams and bigrams that occurred more than once in the training set as features. We did not use trigram features because they did not result in a significant improvement in our pilot study. The total number of candidate features we used was around 860,000.

Our main experimental results are shown in Table 1. Row 1 is our baseline result using the word trigram model. Notice that the result is much better than the state-of-the-art performance currently available in the marketplace (e.g. Gao et al. 2002), presumably due to the large amount of training data we used, and to the similarity between the training and the test data. Row 2 is the result of the model trained using the MSR algorithm described in Section 3. We also compared the MSR algorithm to two of the state-of-the-art discriminative training methods: **Boosting** in Row 3 is an implementation of the improved algorithm for the boosting loss function proposed in (Collins 2000), and **Perceptron** in Row 4 is an implementation of the averaged perceptron algorithm described in (Collins 2002).

We see that all discriminative training methods outperform MLE significantly (p-value < 0.01). In particular, MSR outperforms MLE by more than 20% CER reduction. Notice that we used only unigram and bigram features that have been included in the baseline trigram model, so the improvement is solely attributed to the high performance of MSR. We also find that MSR outperforms the perceptron and boosting methods by a small but statistically significant margin.

The MSR algorithm is also very efficient: using a subset of 20,000 features, it takes less than 20 minutes to converge on an XEON(TM) MP 1.90GHz machine. It is as efficient as the perceptron algorithm and slightly faster than the boosting method.

4.3 Robustness Issues

Most theorems that justify the robustness of discriminative training algorithms concern two questions. First, is there a guarantee that a given algorithm converges even if the training samples are not linearly separable? This is called the *convergence* problem. Second, how well is the training error reduction preserved when the algorithm is applied to unseen test samples? This is called the *generalization* problem. Though we currently cannot give a theoretical justification, we present empirical evidence here for the robustness of the MSR approach.

As Vapnik (1999) pointed out, the most robust linear models are the ones that achieve the least training errors with the least number of features. Therefore, the robustness of the MSR algorithm are mainly affected by the feature selection method. To verify this, we created four different subsets of features using different settings of the feature selection method described in Section 3.4. We selected different numbers of features (i.e. 500 and 2000) with and without taking into account the correlation between features (i.e. α in Equation (7) is set to 0.8 and 1, respectively). For each of the four feature subsets, we used the MSR algorithm to generate a set of models. The CER curves of these models on training and test data sets are shown in Figures 3 and 4, respectively.

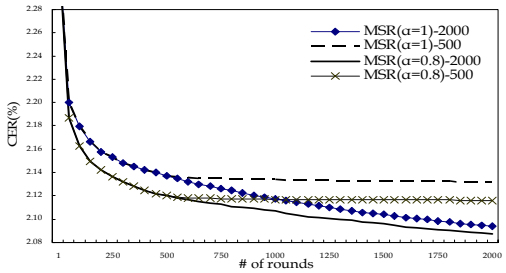

Figure 3. Training error curves of the MSR algorithm

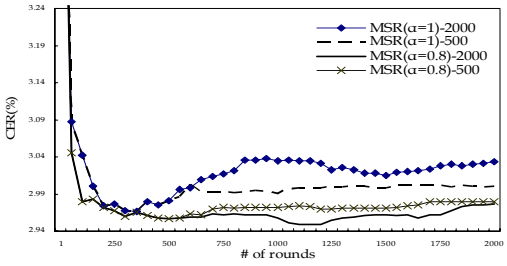

Figure 4. Test error curves of the MSR algorithm

The results reveal several facts. First, the convergence properties of MSR are shown in Figure 3 where in all cases, training errors drop consistently with more iterations. Secondly, as expected, using more features leads to overfitting, For example, MSR(α =1)-2000 makes fewer errors than MSR(α =1)-500 on training data but more errors on test data. Finally, taking into account the correlation between features (e.g. α = 0.8 in Equation (7)) re-

sults in a better subset of features that lead to not only fewer training errors, as shown in Figure 3, but also better generalization properties (fewer test errors), as shown in Figure 4.

4.4 Domain Adaptation Results

Though MSR achieves impressive performance in CER reduction over the comparison methods, as described in Section 4.2, the experiments are all performed using newspaper text for both training and testing, which is not a realistic scenario if we are to deploy the model in an application. This section reports the results of additional experiments in which we adapt a model trained on one domain to a different domain, i.e., in a so-called cross-domain LM adaptation paradigm. See (Suzuki and Gao 2005) for a detailed report.

The data sets we used stem from five distinct sources of text. The Nikkei newspaper corpus described in Section 4.1 was used as the background domain, on which the word trigram model was trained. We used four adaptation domains: Yomiuri (newspaper corpus), TuneUp (balanced corpus containing newspapers and other sources of text), Encarta (encyclopedia) and Shincho (collection of novels). For each of the four domains, we used an 72,000-sentence subset as adaptation training data, a 5,000-sentence subset as held-out data and another 5,000-sentence subset as test data. Similarly, all corpora have been word-segmented, and we kept for each training sample, in the four adaptation domains, the best 20 hypotheses in its candidate conversion set for discriminative training.

We compared MSR with three other LM adaptation methods:

Baseline is the background word trigram model, as described in Section 4.1.

MAP (maximum a *posteriori*) is a traditional LM adaptation method where the parameters of the background model are adjusted in such a way that maximizes the likelihood of the adaptation data. Our implementation takes the form of linear interpolation as $P(w_i|h) = \lambda P_b(w_i|h) + (1-\lambda)P_a(w_i|h)$, where P_b is the probability of the background model, P_a is the probability trained on adaptation data using MLE and the history h corresponds to two preceding words (i.e. P_b and P_a are trigram probabilities). λ is the interpolation weight optimized on held-out data.

Perceptron, **Boosting** and **MSR** are the three discriminative methods described in the previous sections. For each of them, the base feature was derived from the word trigram model trained on the background data, and other n-gram features (i.e. f_d, $d = 1 \ldots D$ in Equation (2)) were trained on adaptation data. That is, the parameters of the background model are adjusted in such a way that minimizes the errors on adaptation data made by background model.

Model	Yomiuri	TuneUp	Encarta	Shincho
Baseline	3.70	5.81	10.24	12.18
MAP	3.69	5.47	7.98	10.76
MSR	**2.73**	**5.15**	**7.40**	**10.16**
Boosting	2.78	5.33	7.53	10.25
Perceptron	2.78	5.20	7.44	10.18

Table 2. CER(%) results on four adaptation test sets .

Results are summarized in Table 2. First of all, in all four adaptation domains, discriminative methods outperform MAP significantly. Secondly, the improvement margins of discriminative methods over MAP correspond to the similarities between background domain and adaptation domains. When the two domains are very similar to the background domain (such as Yomiuri), discriminative methods outperform MAP by a large margin. However, the margin is smaller when the two domains are substantially different (such as Encarta and Shincho). The phenomenon is attributed to the underlying difference between the two adaptation methods: MAP aims to improve the likelihood of a distribution, so if the adaptation domain is very similar to the background domain, the difference between the two underlying distributions is so small that MAP cannot adjust the model effectively. However, discriminative methods do not have this limitation for they aim to reduce errors directly. Finally, we find that in most adaptation test sets, MSR achieves slightly better CER results than the two competing discriminative methods. Specifically, the improvements of MSR are statistically significant over the boosting method in three out of four domains, and over the perceptron algorithm in the Yomiuri domain. The results demonstrate again that MSR is robust.

5 Related Work

Discriminative models have recently been proved to be more effective than generative models in some NLP tasks, e.g., parsing (Collins 2000), POS tagging (Collins 2002) and LM for speech recognition (Roark et al. 2004). In particular, the linear models, though simple and non-probabilistic in nature, are preferred to their probabilistic coun-

terpart such as logistic regression. One of the reasons, as pointed out by Ng and Jordan (2002), is that the parameters of a discriminative model can be fit either to maximize the conditional likelihood on training data, or to minimize the training errors. Since the latter optimizes the objective function that the system is graded on, it is viewed as being more truly in the *spirit* of discriminative learning.

The MSR method shares the same motivation: to minimize the errors directly as much as possible. Because the error function on a finite data set is a step function, and cannot be optimized easily, previous research approximates the error function by loss functions that are suitable for optimization (e.g. Collins 2000; Freund et al. 1998; Juang et al. 1997; Duda et al. 2001). MSR uses an alternative approach. It is a simple heuristic training procedure to minimize training errors directly without applying any approximated loss function.

MSR shares many similarities with previous methods. The basic training algorithm described in Section 3.2 follows the general framework of multi-dimensional optimization (e.g., Press et al. 1992). The line search is an extension of that described in (Och 2003; Quirk et al. 2005. The extension lies in the way of handling large number of features and training samples. Previous algorithms were used to optimize linear models with less than 10 features. The feature selection method described in Section 3.4 is a particular implementation of the feature selection methods described in (e.g., Theodoridis and Koutroumbas 2003). The major difference between the MSR and other methods is that it estimates the effectiveness of each feature in terms of its expected training error reduction while previous methods used metrics that are loosely coupled with reducing training errors. The way of dealing with feature correlations in feature selection in Equation (7), was suggested by Finette et al. (1983).

6 Conclusion and Future Work

We show that MSR is a very successful discriminative training algorithm for LM. Our experiments suggest that it leads to significantly better conversion performance on the IME task than either the MLE method or the two widely applied discriminative methods, the boosting and perceptron methods. However, due to the lack of theoretical underpinnings, we are unable to prove that MSR will always succeed. This forms one area of our future work.

One of the most interesting properties of MSR is that it can optimize any objective function (whether its gradient is computable or not), such as error rate in IME or speech, BLEU score in MT, precision and recall in IR (Gao et al. 2005). In particular, MSR can be performed on large-scale training set with millions of candidate features. Thus, another area of our future work is to test MSR on wider varieties of NLP tasks such as parsing and tagging.

References

Collins, Michael. 2002. Discriminative training methods for Hidden Markov Models: theory and experiments with the perceptron algorithm. In *EMNLP 2002*.

Collins, Michael. 2000. Discriminative reranking for natural language parsing. In *ICML 2000*.

Duda, Richard O, Hart, Peter E. and Stork, David G. 2001. *Pattern classification*. John Wiley & Sons, Inc.

Finette S., Blerer A., Swindel W. 1983. Breast tissue classification using diagnostic ultrasound and pattern recognition techniques: I. Methods of pattern recognition. *Ultrasonic Imaging*, Vol. 5, pp. 55-70.

Freund, Y, R. Iyer, R. E. Schapire, and Y. Singer. 1998. An efficient boosting algorithm for combining preferences. In *ICML'98*.

Gao, Jianfeng, Hisami Suzuki and Yang Wen. 2002. Exploiting headword dependency and predictive clustering for language modeling. In *EMNLP 2002*.

Gao, J, H. Qin, X. Xiao and J.-Y. Nie. 2005. Linear discriminative model for information retrieval. In *SIGIR*.

Jelinek, Fred. 1997. *Statistical methods for speech recognition*. MIT Press, Cambridge, Mass.

Juang, B.-H., W.Chou and C.-H. Lee. 1997. Minimum classification error rate methods for speech recognition. *IEEE Tran. Speech and Audio Processing* 5-3: 257-265.

Ng, A. N. and M. I. Jordan. 2002. On discriminative vs. generative classifiers: a comparison of logistic regression and naïve Bayes. In *NIPS 2002:* 841-848.

Och, Franz Josef. 2003. Minimum error rate training in statistical machine translation. In *ACL 2003*

Press, W. H., S. A. Teukolsky, W. T. Vetterling and B. P. Flannery. 1992. *Numerical Recipes In C: The Art of Scientific Computing*. New York: Cambridge Univ. Press.

Quirk, Chris, Arul Menezes, and Colin Cherry. 2005. Dependency treelet translation: syntactically informed phrasal SMT. In *ACL 2005*: 271-279.

Roark, Brian, Murat Saraclar and Michael Collins. 2004. Corrective language modeling for large vocabulary ASR with the perceptron algorithm. In *ICASSP 2004*.

Suzuki, Hisami and Jianfeng Gao. 2005. A comparative study on language model adaptation using new evaluation metrics. In *HLT/EMNLP 2005*.

Theodoridis, Sergios and Konstantinos Koutroumbas. 2003. *Pattern Recognition*. Elsevier.

Vapnik, V. N. 1999. *The nature of statistical learning theory*. Springer-Verlag, New York.