An Empirical Study on Language Model Adaptation

JIANFENG GAO
Microsoft Research

HISAMI SUZUKI
Microsoft Research

WEI YUAN[1]
Shanghai Jiao Tong University

This paper presents an empirical study on four techniques of language model adaptation, including a maximum *a posteriori* (MAP) method and three discriminative training models, in the application of Japanese Kana-Kanji conversion. We compare the performance of these methods from various angles by adapting the baseline model to four adaptation domains. In particular, we attempt to interpret the results given in terms of the character error rate (CER) by correlating them with the characteristics of the adaptation domain measured using the information-theoretic notion of cross entropy. We show that such a metric correlates well with the CER performance of the adaptation methods, and also show that the discriminative methods are not only superior to a MAP-based method in terms of achieving larger CER reduction, but are also superior in having fewer side effects, and more robust against the similarity of background and adaptation domains.

Categories and Subject Descriptors: 1.2.7 [**Artificial Intelligence**]: Natural Language Processing – *Language Models*

General Terms: Algorithm, Languages, Human Factors, Theory

Additional Key Words and Phrases: statistical language modeling, discriminative training, Asian language text input, domain adaptation, entropy

---

1. INTRODUCTION

Language model (LM) adaptation attempts to adjust the parameters of a LM so that it performs well on a particular domain of data. This paper presents an empirical study of several LM adaptation methods on the task of Japanese text input. In particular, we focus on the so-called *cross-domain* LM adaptation paradigm, i.e. to adapt a LM trained on one domain (which we call the *background* domain) to a different domain (*adaptation* domain), for which only a small amount of training data is available.

The LM adaptation methods investigated in this paper can be grouped into two categories: maximum *a posteriori* (MAP) and discriminative training. Linear interpolation is representative of a MAP method [Bellagarda 2001]. The other three methods, including the boosting [Collins 2000] and perceptron [Collins 2002] algorithms and the minimum sample risk (MSR) method [Gao et al. 2005], are discriminative methods, each of which uses a different training algorithm.

We carried out experiments over many training data sizes on four distinct adaptation corpora, the characteristics of which were measured using the information-theoretic notion of cross entropy. We found that discriminative training methods outperformed the linear interpolation method in all cases, with some additional desirable properties: they were not only better in terms of character error rate, but were also superior in having fewer side effects, and were

[1] This research was conducted while the author was visiting Microsoft Research Asia.

more robust across different training sets of different domains and sizes. None of the discriminative training methods, however, was found to significantly outperform the others in our experiments.

The rest of the paper is organized as follow. Section 2 introduces the task of IME and the role of LM. In Section 3, we review related work. After a description of the LM adaptation methods that are used in our experiments in Section 4, Sections 5 and 6 present experimental results and their discussions. We conclude our paper in Section 7.

2. LANGUAGE MODEL AND THE TASK OF IME

Our study falls into the context of Asian language (Japanese in this study) text input. The standard method for doing this is that the users first input phonetic strings, which are then converted into appropriate word strings by software. The task of automatic conversion has been the subject of language modeling research in the context of *Pinyin-to-Character conversion* in Chinese [Gao et al. 2002a] and *Kana-Kanji conversion* in Japanese [Gao et al. 2002b]. In this paper, we call the task *IME* (Input Method Editor), based on the name of the commonly used Windows-based application.

The performance of IME is typically 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 commercial Japanese IME systems exhibit about 5-15% CER in conversion of real-world data in a wide variety of domains.

In many ways, IME is a similar task to speech recognition. The most obvious similarity is that IME can also be viewed as a Bayesian decision problem: let A be the input phonetic string (which corresponds to the acoustic signal in speech); the task of IME is to choose the most likely word string W^* among those candidates that could have been converted from A:

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

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

Unlike speech recognition, however, there is no acoustic ambiguity in IME, because the phonetic string is provided directly by users. Moreover, we can assume a unique mapping from W to A in IME, i.e., $P(A \mid W) = 1$. So the decision of Equation (1) depends solely upon $P(W)$, i.e., the language model probability, making IME an ideal application for evaluating LM techniques.[2] Another advantage is that it is relatively easy to convert W to A, making it possible to obtain a large amount of training data for discriminative learning, as described later.

From the perspective of LM adaptation, IME faces the same problem speech recognition does: the quality of the model depends heavily on the similarity of the training and test data. This poses a serious challenge to IME, as it is currently the most widely used method of inputting Chinese or Japanese characters, used by millions of users for inputting text of a wide variety of domains. LM adaptation in

[2] For this reason, we have made the MSR-IME corpus [Suzuki and Gao 2005b] available for research purposes, in the hope that it will facilitate further LM research.

IME is therefore an imminent requirement for improving user experience, not only as we build static domain-specific LMs, but also in making online user adaptation possible in the future.

3. RELATED WORK

One of our goals in this paper is to quantify the characteristics of different domains of text, and to correlate them with the performance of various techniques for LM adaptation to compare their effectiveness and robustness. This relates our work to the study of domain similarity calculation and to different techniques for LM adaptation.

3.1 Measuring Domain Similarity

Statistical language modeling (SLM) assumes that language is generated from underlying distributions. When we discuss different domains of text, we assume that the text from each of these domains is generated from a different underlying distribution. We therefore consider the problem of distributional similarity in this paper.

Cross entropy is a widely used measure in evaluating LM. Given a language L with its true underlying probability distribution p and another distribution q (e.g., a SLM) which attempts to model L, the cross entropy of L with respect to q is

$$H(L,q) = -\lim_{n\to\infty}\frac{1}{n}\sum_{w_1\ldots w_n} p(w_1\ldots w_n)\log q(w_1\ldots w_n) \tag{2}$$

where $w_1 \ldots w_n$ is a word string in L. However, in reality, the underlying p is never known and the corpus size is never infinite. We therefore make the assumption that L is an ergodic and stationary process [Manning and Schütze 1999], and approximate the cross entropy by calculating it for a sufficiently large n instead of calculating it for the limit.

$$H(L,q) \approx -\frac{1}{n}\log q(w_1\ldots w_n) \tag{3}$$

This measures how well a model approximates the language L.

The Kullback-Leibler (KL) divergence, or relative entropy, is another measure of distributional similarity that has been widely used in natural language processing and information retrieval [Dagan et al. 1999]. Given the two distributions p and q above, the KL divergence is defined as

$$D(p(w_1\ldots w_n)\,||\,q(w_1\ldots w_n)) = \sum_{w_1\ldots w_n} p(w_1\ldots w_n)\log\frac{p(w_1\ldots w_n)}{q(w_1\ldots w_n)} \tag{4}$$

The cross entropy and the KL divergence are related notions. Given the notations of L, p and q above, Manning and Schütze [1999] show that

$$H(L,q) = H(L) + D(p\,||\,q) \tag{5}$$

In other words, the cross entropy takes into account both the similarity between two distributions (given by KL divergence) and the entropy of the corpus in question, both of which contribute to the complexity of a LM task. In this paper we are interested in measuring the complexity of the LM adaptation task. We

therefore define the similarity between two domains using the cross entropy. We will also use the metric that approximates the entropy of the corpus to capture the in-domain diversity of a corpus, as described in Section 5.2.[3]

3.2 LM Adaptation Methods

In this paper, two major approaches to cross-domain adaptation have been investigated: maximum *a posteriori* (MAP) estimation and discriminative training methods.

In MAP estimation methods, adaptation data is used to adjust the parameters of the background model so as to maximize the likelihood of the adaptation data [Bellagarda 2001]. Discriminative training methods of LM adaptation, on the other hand, aim at using the adaptation data to directly minimize the errors on the adaptation data made by the background model. These techniques have been applied successfully to the task of language modeling in non-adaptation [Roark et al. 2004] as well as adaptation scenarios [Bacchiani et al. 2004] for speech recognition. But most of them focused on the investigation of performance of certain methods for LM adaptation, without analyzing in detail the underlying reasons of different performance achieved by different methods. In this paper we attempt to investigate the effectiveness of different discriminative methods in an IME adaptation scenario, with a particular emphasis on correlating their performance with the characteristics of adaptation domain.

4. LM ADAPTATION METHODS

We have implemented four methods in our experiments. The Linear Interpolation (LI) falls into the framework of MAP while the boosting, the perceptron and the MSR methods fall into that of discriminative training.

4.1 Linear Interpolation Method

In MAP estimation methods, adaptation data is used to adjust the parameters of the background model so as to maximize the likelihood of the adaptation data.

The linear interpolation is a special case of MAP according to Bacchiani and Roark [2003]. At first, we generate trigram models on background data and adaptation data respectively. The two models are then combined into one as:

$$P(w_i \mid h) = \lambda P_B(w_i \mid h) + (1-\lambda) P_A(w_i \mid h) \tag{6}$$

where P_B is the probability of the background model, P_A is the probability of the adaptation model and the history h corresponds to two preceding words. For simplicity, we chose a single λ for all histories and tune it on held-out data.

4.2 Discriminative Training Methods

This section describes three discriminative training methods we used in this study. For a detailed description of each algorithm, readers are referred to Collins [2000] for the boosting algorithm, Collins [2002] for the perceptron learning algorithm, and Gao et al. [2005] for the MSR method.

---

[3] There are other well-known metrics of similarity within NLP literature, such as the mutual information or cosine similarity [Lee 1999], which we do not discuss in this paper.

4.2.1 Problem Definition

All the three discriminative training methods follow the general framework of linear models [Duda et al. 2001; Collins 2002]. We use the following notation, adapted from Collins [2002], in the rest of the paper.

- 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 **GEN**(A). In our experiments, **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 $\mathbf{f}(W) = [f_0(W), f_1(W), \ldots, f_D(W)]^T$. $f_0(W)$ 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

$$Score(W, \boldsymbol{\lambda}) = \boldsymbol{\lambda}\mathbf{f}(W) = \sum_{d=0}^{D} \lambda_d f_d(W). \tag{7}$$

The decision rule of Equation (1) is rewritten as

$$W^*(A, \boldsymbol{\lambda}) = \arg\max_{W \in \mathbf{GEN}(A)} Score(W, \boldsymbol{\lambda}). \tag{8}$$

Equation (8) views IME as a ranking problem, where the model gives the ranking score, not probabilities. We therefore do not evaluate the LM obtained using discriminative training via perplexity.

Assume we can measure the number of conversion errors in W by comparing it with a reference transcript W^R using an error function Er(W^R,W), which is an edit distance function in our case. We call the sum of error counts over the training samples *sample risk* (SR). Discriminative training methods strive to minimize the SR by optimizing the model parameters, as defined in Equation (9), where W_i is determined by Equation (8),

$$\boldsymbol{\lambda}^* = \arg\min_{\boldsymbol{\lambda}} SR(\boldsymbol{\lambda}) = \arg\min_{\boldsymbol{\lambda}} \sum_{i=1 \ldots M} Er(W_i^R, W_i(A_i, \boldsymbol{\lambda})). \tag{9}$$

However, SR(.) cannot be optimized easily since Er(.) is a piecewise constant (or step) function of $\boldsymbol{\lambda}$ and its gradient is undefined. Therefore, discriminative methods apply different approaches that optimize it approximately. As we shall describe below, the boosting and perceptron algorithms approximate SR(.) by loss functions that are suitable for optimization, while MSR uses a simple heuristic training procedure to minimize SR(.) directly without resorting to an approximated loss function. We now describe each of the discriminative methods in turn.

4.2.2 The Boosting Algorithm

1 Set $\lambda_0 = 1$ and $\lambda_d = 0$ for d=1…D
2 Select a feature f_d which has largest estimated impact on reducing ExpLoss of Equation (12)
3 Update $\lambda_d = \lambda_d + \delta_d$, where δ_d is estimated by Equation (13), and return to Step 2

Figure 1: The boosting algorithm

The boosting algorithm we used is based on Collins [2000]. It uses an exponential function to approximate SR(.). We define a ranking error in a case where an incorrect candidate conversion W gets a higher score than the correct conversion W^R. The margin of the pair (W^R, W) with respect to the model $\boldsymbol{\lambda}$ is estimated as

$$M(W^R, W) = Score(W^R, \boldsymbol{\lambda}) - Score(W, \boldsymbol{\lambda}) \tag{10}$$

The rank loss function (RLoss) is then defined as

$$\text{RLoss}(\boldsymbol{\lambda}) = \sum_{i=1...M} \sum_{W_i \in \mathbf{GEN}(A_i)} I[M(W_i^R, W_i)] \tag{11}$$

where $I[\pi]$ = 1 if $\pi \leq 0$, and 0 otherwise. Note that RLoss takes into account all candidates in GEN(A). Since optimizing the RLoss in (11) is NP-complete, the boosting algorithm optimizes its upper bound, i.e., the exponential loss function (ExpLoss), as

$$\text{ExpLoss}(\boldsymbol{\lambda}) = \sum_{i=1...M} \sum_{W_i \in \mathbf{GEN}(A_i)} \exp(-M(W_i^R, W_i)) \tag{12}$$

Notice that ExpLoss is convex so there is no problem with local minima when optimizing it. It is shown in Freund et al. [1998] and Collins [2000] that there exist gradient search procedures (i.e., RankBoost and its variants) that converge to the right solution.

Figure 1 summarizes the boosting algorithm we used. After initialization, Steps 2 and 3 are repeated N times; at each iteration, a feature is chosen and its weight is updated. We used the following update for the dth feature f_d:

$$\delta_d = \frac{1}{2} \log \frac{C_d^+ + \varepsilon Z}{C_d^- + \varepsilon Z} \tag{13}$$

where C_d^+ is a value increasing exponentially with the sum of margins of (W^R, W) pairs over the set where f_d is seen in W^R but not in W; C_d^- is the value related to the sum of margins over the set where f_d is seen in W but not in W^R. ε is a smoothing factor (whose value is optimized on held-out data) and Z is a normalization constant.

In short, the boosting criterion ExpLoss approximates SR in two steps: using a ranking error loss function RLoss to approximate SR, and optimizing RLoss approximately by minimizing its upper bound function ExpLoss.

4.2.3 The Perceptron Algorithm

The perceptron algorithm can be viewed as a form of incremental training procedure (e.g., using stochastic approximation) that optimizes a *minimum square error* (MSE) loss function, which is an approximation of SR [Mitchell 1997].

The MSE loss function is defined as

$$\text{MSELoss}(\boldsymbol{\lambda}) = \frac{1}{2} \sum_{i=1...M} (Score(W_i^R, \boldsymbol{\lambda}) - Score(W_i, \boldsymbol{\lambda}))^2 \quad (14)$$

It is simply half the squared difference between the score of the correct conversion and the score of the incorrect one, summing over all training samples. It is useful to note that the MSE solution, under certain conditions, leads to approximations to maximum likelihood (ML) solution. The quality of the approximation depends upon the form of the linear discriminant functions (e.g., Equation (7)). While this property has a certain theoretical appeal, the discriminant function that best approximates the Bayes discriminant does not necessarily minimize the sample risk, as discussed in Duda et al. [2001]. In such a sense, similar to ExpLoss, MSELoss is also an approximation to SR.

Despite this property, the MSE criterion has received considerable attention in the literature, and there are many solution procedures available. Here, we consider the *delta rule*, a training algorithm of an unthresholded perceptron. Below, we discuss the derivation of the delta rule, following the description in Mitchell [1997].

The delta rule in its component form is

$$\lambda_d = \lambda_d - \eta \times G(\lambda_d), \quad (15)$$

where η is the step size, and G is the gradient of MSELoss. G can be estimated by differentiating the loss function of Equation (14) with respect to λ_d as

$$\begin{aligned} G(\lambda_d) &= \frac{\partial \text{MSELoss}(\boldsymbol{\lambda})}{\partial \lambda_d} \\ &= \sum_{i=1...M} (Score(W_i^R, \boldsymbol{\lambda}) - Score(W_i, \boldsymbol{\lambda}))(f_d(W_i^R) - f_d(W_i)) \end{aligned} \quad (16)$$

However, the objective function of Equation (14) in the context of our task has many local minima, and thus gradient descent is not guaranteed to find the global minimum. We therefore use a *stochastic approximation* to gradient descent. Whereas gradient descent computes parameter updates after summing over all training samples as shown in Equation (16), the stochastic approximation method updates parameters incrementally, following the calculation of the error for each individual training sample, as shown in Equation (17).

$$G(\lambda_d) = (Score(W_i^R, \boldsymbol{\lambda}) - Score(W_i, \boldsymbol{\lambda}))(f_d(W_i^R) - f_d(W_i)) \quad (17)$$

The stochastic approximation method can be viewed as optimizing a distinct loss function $\text{MSELoss}_i(\boldsymbol{\lambda})$ defined for each individual training sample i as follows

$$\text{MSELoss}_i(\boldsymbol{\lambda}) = \frac{1}{2}(Score(W_i^R, \boldsymbol{\lambda}) - Score(W_i, \boldsymbol{\lambda}))^2 \quad (18)$$

The optimization algorithm we used in our experiments is shown in Figure 2. It takes T passes over the training set. All parameters (except for λ_0) are initially set to be 0. Each training sample (i.e. phonetic string A) is converted using the current parameter settings. If the highest scoring word sequence under the current model is not correct, the parameters are updated in a simple additive fashion: to alter the parameters according to the gradient with respect to $\text{MSELoss}_i(\boldsymbol{\lambda})$. Empirically, the

1 Initialize all parameters in the model, i.e. $\lambda_0 = 1$ and $\lambda_d = 0$ for d=1…D
2 For t = 1…T, where T is the total number of iterations
3 For each training sample (A_i, W_i^R), i = 1…M
4 Use current model $\boldsymbol{\lambda}$ to choose some W_i from GEN(A_i) by Equation (8)
5 For d = 1 … D
6 $\lambda_d = \lambda_d + \eta(Score(W_i^R, \boldsymbol{\lambda}) - Score(W_i, \boldsymbol{\lambda}))(f_d(W_i^R) - f_d(W_i))$, where η is the size of the learning step

Figure 2: The standard perceptron algorithm with delta rule

sequence of these updates, when iterated over all training samples, provides a reasonable approximation to descending the gradient with respect to the original loss function of Equation (14). The algorithm is similar to the perceptron algorithm described in Collins [2002]. The key difference is that instead of using the delta rule of (15) (as shown in Line 6 of Figure 2), Collins [2002] updates parameters using the rule: $\lambda_d = \lambda_d + f_d(W_i^R) - f_d(W_i)$. Our pilot study shows that the delta rule leads to a slightly better performance. Following Collins [2002], we used the averaged perceptron algorithm in our experiments, a simple refinement to the algorithm in Figure 2, which has been proved to be more robust. Let $\lambda_d^{t,i}$ be the value for the dth parameter after the ith training sample has been processed in pass t over the training data. Then the "averaged parameters" are defined as in Equation (19).

$$(\lambda_d)_{avg} = (\sum_{t=1}^{T}\sum_{i=1}^{M}\lambda_d^{t,i}) / (T \cdot M) \tag{19}$$

In short, there are two approximations embedded in the perceptron algorithm. One is the use of the MSE criterion that approximates the ML criterion. The other is the stochastic approximation that is introduced for parameter optimization. Though in theory these approximations could to some degree prevent the algorithm from attaining the original objective of minimizing the SR, they turn out to be an effective compromise empirically, as we shall show in Section 5.

4.2.4 The Minimum Sample Risk Method

The minimum sample risk (MSR) [Gao et al. 2005] training algorithm is motivated by analogy with the feature selection procedure for the boosting algorithm [Freund et al. 1998]. It is a greedy procedure for selecting a small subset of the features that have the largest contribution in reducing SR in a sequential manner.

Conceptually, MSR operates like any multidimensional function optimization approach: the first direction (i.e., feature) is selected and SR is minimized along that direction using a *line search*, i.e., adjusting the parameter of the selected feature while keeping all other parameters fixed; 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 SR 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 (9) 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 with each other. Below, we will address these issues respectively.

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 (9), the sample risk over all training samples is the sum of the loss function (i.e. Er(.)) of each training sample. Therefore, we first explain how to minimize 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 (7), 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 (8).

In this way, we can 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. 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.

In addition to reducing $\mathbf{GEN}(A)$ 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 is very sparse. Therefore, we built an inverted

index that lists for each feature all training samples that contain it. Our experiments show that the line search is very efficient even for a large training set with millions of candidate features.

Feature Subset Selection. 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. 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

$$E(f_d) = \frac{\mathrm{SR}(f_0) - \mathrm{SR}(f_0 + \lambda_d f_d)}{\max_{f_i, i=1\ldots D} (\mathrm{SR}(f_0) - \mathrm{SR}(f_0 + \lambda_i f_i))}, \tag{20}$$

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 (20) 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 highly correlating features. 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 , if they are highly correlated with each other. 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, computed by Equation (20), is larger than zero, and 0 otherwise. The cross correlation coefficient between two features f_i and f_j is estimated as

$$C(i, j) = \frac{\sum_{m=1}^{M} x_{mi} x_{mj}}{\sqrt{\sum_{m=1}^{M} x_{mi}^2 \sum_{m=1}^{M} x_{mj}^2}} \tag{21}$$

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 $d = 1\ldots D$), compute the value of $E(f)$ according to Equation (20). 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 (21).

1 Set $\lambda_0 = 1$ and $\lambda_d = 0$ for d=1…D
2 Rank all features and select the top K features, using the feature subset selection method.
3 For $t = 1…T$ (T= total number of iterations)
4 For each $k = 1…K$
5 Update the parameter of f_k using line search.

Figure 3: The MSR algorithm

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

$$j^* = \arg\max_{j} \left\{ \alpha E(f_j) - \frac{1-\alpha}{k-1} \sum_{i=1}^{k-1} C(i, j) \right\} \quad (22)$$

That is, we select the next feature by taking into account its average correlation with all previously selected features.

Similarly to the case of line search, we need to deal with the efficiency issue in the feature selection method. As shown in Equation (22), 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, which is 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 3. It consists of feature selection (line 2) and optimization (lines 3 - 5) steps. Readers are referred to Gao et al. [2005] for a complete description of the MSR implementation and the empirical justification for its performance.

5. EXPERIMENTAL RESULTS

5.1 Data

The data used in our experiments stem from five distinct sources of text. A 36-million-word *Nikkei* newspaper corpus was used as the background domain. We used four adaptation domains: *Yomiuri* (newspaper corpus), *TuneUp* (balanced corpus containing newspaper and other sources of text), *Encarta* (encyclopedia) and *Shincho* (collection of novels).

For the computation of domain characteristics (Section 5.2), we extracted 1 million words from the training data of each domain respectively (corresponding to 13K to 78K sentences depending on the domain). For this experiment, we also used a lexicon consisting of the words in our baseline lexicon (167,107 words) plus all words in the corpora used for this experiment (that is, 1M words times 5 domains), which included 216,565 entries. The use of such a lexicon was motivated by the need to eliminate the effect of out-of-vocabulary (OOV) items.

For the experiment of LM adaptation (Section 5.3), we created training data consisting of 72K sentences (0.9M~1.7M words) and test data of 5K sentences (65K~120K words) from each adaptation domain. The first 800 and 8,000 sentences of each adaptation training data were also used to show how different sizes of adaptation training data affected the performances of various adaptation methods. Another 5K-sentence subset was used as held-out data for each domain. For domain adaptation experiments, we used our baseline lexicon consisting of 167,107 entries.

5.2 Computation of Domain Characteristics

The first domain characteristic we computed was the similarity between two domains for the task of LM. As discussed in Section 3, we used the cross entropy as the metric: we first trained a word trigram model using the system described in Gao et al. [2002a] on the 1-million-word corpus of domain B, and used it in the computations of the cross entropy $H(L_A, q_B)$ following Equation (3). For simplicity, we denote $H(L_A, q_B)$ as $H(A,B)$.

Table I displays the cross entropy between two domains of text. Note that the cross entropy is not symmetric, i.e., $H(A,B)$ is not necessarily the same as $H(B,A)$. In order to have a representative metric of similarity between two domains, we computed the *average cross entropy* between two domains, shown in Table II, and used this quantity as the metric for domain similarity.

Along the main diagonal in Tables I and II, we also have the cross entropy computed for $H(A,A)$, i.e., when two domains we compare are the same (in boldface). This value, which we call *self entropy* for convenience, is an approximation of the entropy of the corpus A, and measures the amount of information per word, i.e., the *diversity* of the corpus. Note that the self entropy increases in the order of Nikkei → Yomiuri → Encarta → TuneUp → Shincho. This indeed reflects the in-domain variability of text: Nikkei, Yomiuri and Encarta are highly edited text, following style guidelines; they also tend to have repetitious content. In contrast, Shincho is a collection of novels, on which no style or content restriction is imposed. We expect that the LM task to be more difficult as the corpus is more diverse; we will further discuss the effect of diversity in Section 6.[4]

[4] Another derivative notion from Table I is the notion of balanced corpus. In Table I, the smallest cross entropy for each text domain (rows) is the self entropy (in boldface), as expected. Note, however, that the second smallest cross entropy (underlined) is always obtained from the TuneUp model (except for Nikkei, for which Yomiuri provides the second smallest cross entropy). This reflects the fact that the

Table I. Cross entropy (rows: corpora; column: models)

	Nikkei	Yomiuri	TuneUp	Encarta	Shincho
Nikkei	**3.94**	7.46	7.65	9.81	10.10
Yomiuri	7.93	**4.09**	7.62	9.26	9.97
TuneUp	8.25	8.03	**4.41**	9.04	9.06
Encarta	8.79	8.66	8.60	**4.40**	9.30
Shincho	8.70	8.61	8.07	9.10	**4.61**

Table II. Average cross entropy

	Nikkei	Yomiuri	TuneUp	Encarta	Shincho
Nikkei	**3.94**	7.69	7.95	9.30	9.40
Yomiuri		**4.09**	7.82	8.96	9.29
TuneUp			**4.41**	8.82	8.56
Encarta				**4.40**	9.20
Shincho					**4.61**

5.3 Results of LM Adaptation

We trained our baseline trigram model on our background (Nikkei) corpus using the system described in Gao et al. [2002a]. The CER (%) of this model on each adaptation domain is in the second column of Table III. For the LI adaptation method (the third column of Table III), we trained a word trigram model on the adaptation data, and linearly combined it with the background model, as described in Equation (6).

For the discriminative methods (the last three columns in Table III), we produced a candidate word lattice for each input phonetic string in the adaptation training set using the background trigram model mentioned above. For efficiency purposes, we kept only the best 20 hypotheses from the lattice as the candidate conversion set for discriminative training. The lowest CER hypothesis in the lattice, rather than the reference transcript, was used as the gold standard[5].

To compare the performances of different discriminative methods, we fixed the following parameter settings: we set the number of iterations N to be 2,000 for the boosting and MSR methods (i.e., at most 2,000 features in the final models); for the perceptron algorithm, we set T = 40 (in Figure 1). These settings might lead to an unfair comparison, as the perceptron algorithm will select far more features than the boosting and MSR algorithm. However, we used these settings as they all converged under these settings. All other parameters were tuned empirically on held-out data.

In evaluating both MAP and discriminative methods, we used an N-best rescoring approach. That is, we created N-best hypotheses using the background trigram model (N=100 in our experiments) for each sentence in test data, and used domain-adapted models to rescore the N-best list. The oracle CERs (i.e., the

TuneUp corpus was created by collecting sentences from various sources of text, in order to create a representative test corpus. Using the notion of cross entropy, such a characteristic of a test corpus can also be quantified.

[5] This is an empirical decision which is also applied by other researchers e.g., Roark et al. [2004].

Table III. CER (%) and CER reduction (%) over Baseline
(Y=Yomiuri; T=TuneUp; E=Encarta; S=-Shincho)

Domain	Baseline	LI	Boosting	Perceptron	MSR
Y (800)	3.70	3.70 (0.00)	3.13 (15.41)	3.18 (14.05)	3.17 (14.32)
Y (8K)	3.70	3.69 (0.27)	2.88 (22.16)	2.85 (22.97)	2.88 (22.16)
Y (72K)	3.70	3.69 (0.27)	2.78 (24.86)	2.78 (24.86)	2.73 (26.22)
T (800)	5.81	5.81 (0.00)	5.69 (2.07)	5.69 (2.07)	5.70 (1.89)
T (8K)	5.81	5.70 (1.89)	5.48 (5.85)	5.47 (5.85)	5.47 (5.85)
T (72K)	5.81	5.47 (5.85)	5.33 (8.26)	5.20 (10.50)	5.15 (11.36)
E (800)	10.24	9.60 (6.25)	9.82 (4.10)	9.43 (7.91)	9.44 (7.81)
E (8K)	10.24	8.64 (15.63)	8.54 (16.60)	8.34 (18.55)	8.42 (17.77)
E (72K)	10.24	7.98 (22.07)	7.53 (26.46)	7.44 (27.34)	7.40 (27.73)
S (800)	12.18	11.86 (2.63)	11.91 (2.22)	11.90 (2.30)	11.89 (2.38)
S (8K)	12.18	11.15 (8.46)	11.09 (8.95)	11.20 (8.05)	11.04 (9.36)
S (72K)	12.18	10.76 (11.66)	10.25 (15.85)	10.18 (16.42)	10.16 (16.58)

minimal possible CER given the hypotheses in the list) ranged from 1.45% to 5.09% depending on the adaptation domain. Table III summarizes the results of various adaptation methods in terms of CER (%) and CER reduction (in parentheses) over the baseline model. In the first column, the numbers in parentheses next to the domain name indicates the number of training sentences used for adaptation.

6. DISCUSSION

6.1 Domain Similarity and CER

The first row of Table II shows that the average cross entropy with respect to the background domain (Nikkei) increases in the following order: Yomiuri → TuneUp → Encarta → Shincho. This indicates that among the adaptation domains, Yomiuri is the most similar to Nikkei, closely followed by TuneUp; Shincho and Encarta are the least similar to Nikkei. This is consistent with our intuition, since Nikkei and Yomiuri are both newspaper corpora, and TuneUp, which is a manually constructed corpus from various representative domains of text, contains newspaper articles.

This metric of similarity correlates very well with the CER. Figure 4 plots the domain similarity, indicated by cross entropy, with Nikkei (the line graph, scaled on the right axis) along with CER when 8,000 training sentences were used in adaptation experiments (expressed in the bar graph, scaled on the left axis). The correlation between the domain similarity and the CER is directly observed from the graph: the correlation coefficient was r=0.94 using the Pearson product moment correlation coefficient using data from all sizes. In other words, the more similar the adaptation domain is to the background domain, the better the CER results are.

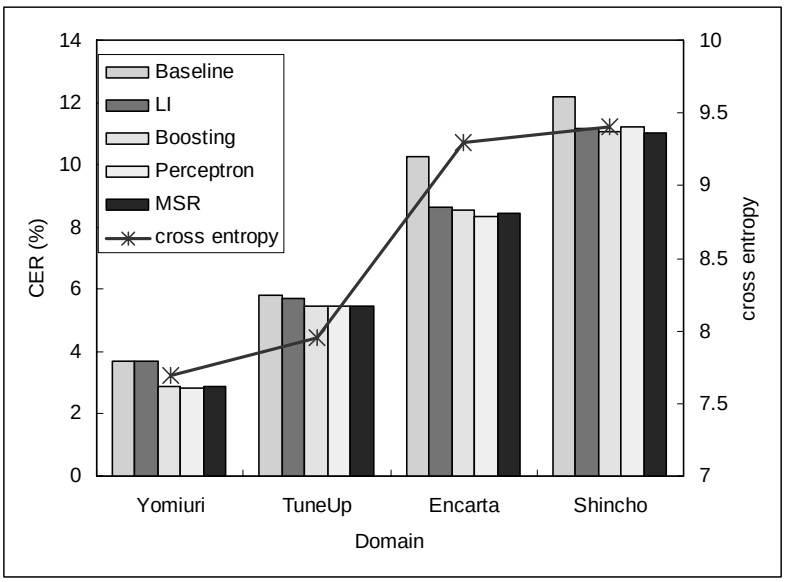

Figure 4: CER and domain similarity

6.2 Domain Similarity and the Robustness of Adaptation Methods

The effectiveness of a LM adaptation method is measured by the relative CER reduction over the baseline model. Figure 5 shows the CER reduction of various methods for each domain when the training data size was 8K.[6]

In Figure 5, the X-axis is arranged in the order of domain similarity with the background domain, i.e., Yomiuri → TuneUp → Encarta → Shincho. The first thing we notice is that the discriminative methods outperform LI in most cases: in fact, for all rows in Table III, MSR outperforms LI in a statistically significant manner ($p < 0.01$ using t-test);[7] the differences among the three discriminative methods, on the other hand, are not statistically significant in most cases.

We also note that the performance of LI is greatly influenced by domain similarity. More specifically, when the adaptation domain is similar to the background domain (i.e., for Yomiuri and TuneUp corpora), the contribution of the LI model is extremely limited. This can be explained as follows: if the adaptation data is too similar to the background, the difference between the two underlying distributions is so slight that adding adaptation data leads to no or very small improvements.

Such a limitation is not observed with the discriminative methods. For example, all discriminative methods are quite effective on Yomiuri, achieving more than 20% CER reduction. We therefore conclude that discriminative methods, unlike LI, are robust against the similarity between background and adaptations domains.

It is worth noting that our results differ from Bacchiani et al. [2004] in that in our system, the perceptron algorithm alone achieved better results than MAP estimation. However, the difference may only be apparent, given different

[6] Essentially the same trend is observed with other training data sizes.
[7] The only exception to this was Shincho (800).

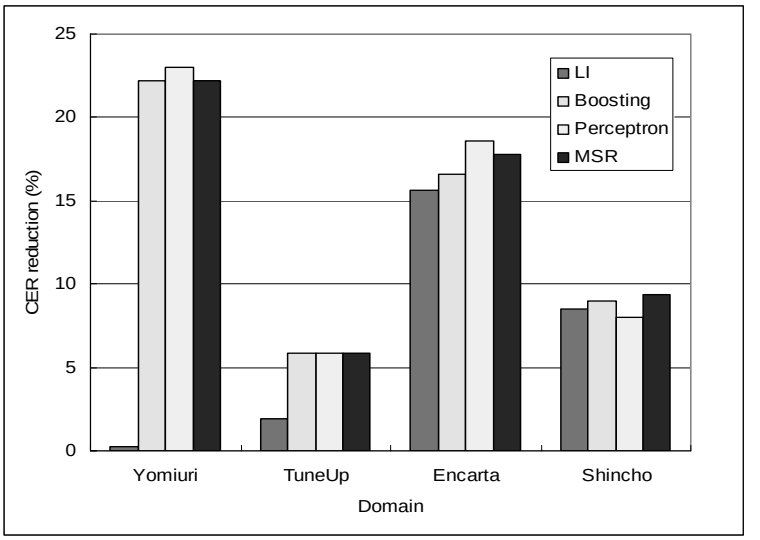

Figure 5: CER reduction by different adaptation methods

experimental settings for the two studies. We used the N-best reranking approach with the same N-best list for both MAP estimation and discriminative training, while in Bacchiani et al. [2004], two different lattices were used: the perceptron model was applied to rerank the lattice created by the background model, while the MAP adaptation model was used to produce the lattice itself. The fact that the combination of these models (i.e., first use the MAP estimation to create hypotheses and then use the perceptron algorithm to rerank them) produced the best results [Bacchiani et al. 2004] indicates that given a candidate lattice, the perceptron algorithm is effective in candidate reranking, thus making our results compatible with theirs.

6.3 Adaptation Data Size and CER Reduction

Among the discriminative methods, an interesting characteristic regarding the CER reduction and the data size is observed. Figure 6 displays the self entropy of four adaptation corpora along the X-axis, and the improvement in CER reduction when 72K-sentence adaptation data is used over when 800 sentences are used along the Y-axis. In other words, for each adaptation method, each point in the figure corresponds to the CER reduction ratio on a domain (corresponding to Yomiuri, Encarta, TuneUp, Shincho from left to right) when 90 times more adaptation data was available.

From this figure, we can see that there is a positive correlation between the diversity of the adaptation corpus and the benefit of having more training data available (r=0.9~0.94 depending on the training method).[8] This has an intuitive explanation: the less diverse the adaptation data is, the less distinct training examples it will include for discriminative training. This result is useful in guiding the process of adaptation data collection.

[8] The correlation is weaker in the boosting method than the other two discriminative methods for the reasons that are not clear to us at this moment.

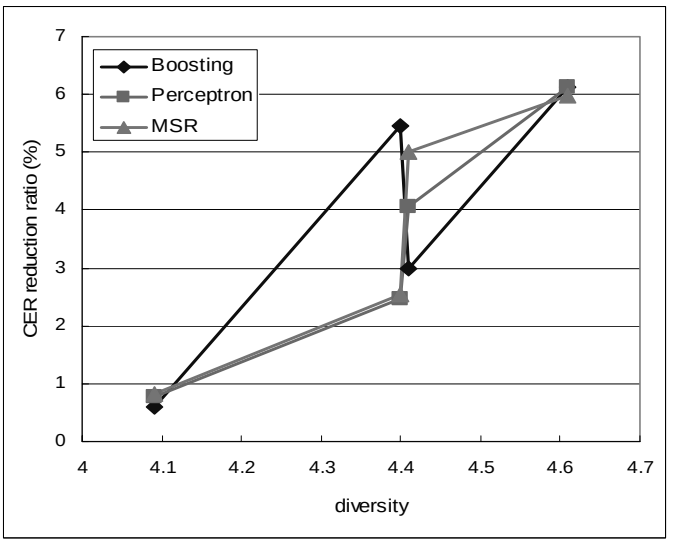

Figure 6: Improvement in CER reduction for discriminative methods by increasing the adaptation data size from 800 to 72K sentences

6.4 Domain Characteristics and Error Ratio

The results presented so far measure the performance of various adaptation techniques in terms of CER. However, CER is not the only metric that provides meaningful comparison among systems. Suzuki and Gao [2005a] discusses the metric of *error ratio* (ER), which measures the side effects of a new model, i.e., the number of newly introduced errors relative to the number of errors that were corrected by the new model.[9] Such a metric should be useful in an actual software deployment scenario: if there are two new models with the same CER performance, the model with smaller error ratio should be desirable, as the software users are more intolerant to newly introduced errors than seeing errors that have always existed. In this section, we compare the adaptation methods using ER.

According to Suzuki and Gao [2005a], error ratio is defined as

$$ER = \frac{|E_A|}{|E_B|},$$

where $|E_A|$ is the number of errors found *only* in the new (adaptation) model, and $|E_B|$ the number of errors corrected by the new model. Intuitively, this quantity captures the *cost* of improvement in the adaptation model, corresponding to the number of newly introduced errors per each improvement. The smaller the ratio is, the better the model is at the same CER: ER=0 if the adapted model introduces no new errors, ER<1 if the adapted model makes CER improvements, ER=1 if the CER improvement is zero (i.e., the adapted model makes as many new mistakes as it corrects old mistakes), and ER>1 when the adapted model has worse CER performance than the baseline model.

[9] Gillick and Cox [1989] use these numbers to perform McNemar's test to obtain the statistical significance of the difference in performance of two speech recognition algorithms.

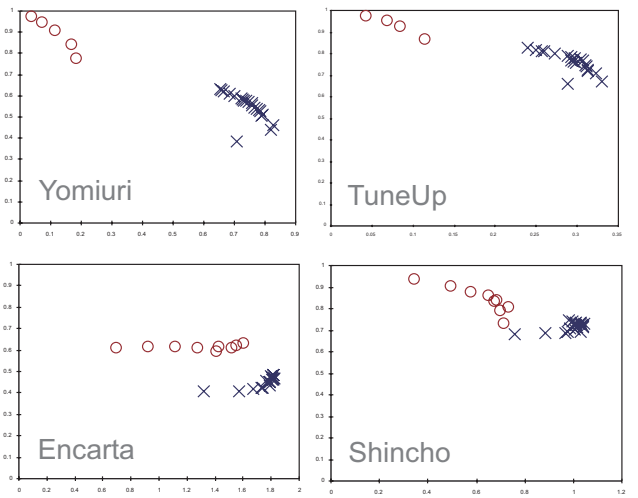

Figure 7: Relative error reduction/ER plot for all four domains
x-axes: RER (%); y-axes: ER. ○ : linear interpolation models; ×:MSR models

Figure 7 compares the performance of various MSR models at different iterations with linear interpolation models at various lambda values in four adaptation domains using the metric of error ratio. In each graph, the x-axis plots the *relative error rate reduction* (RER, i.e., the CER difference between the background and adapted models in %), and the y-axis the error ratio (max=1; min=0). We can see that MSR models are better than linear interpolation models in all domains, as they achieve larger CER reduction (larger values on the x-axis) at smaller ER (smaller values on the y-axis). When the models achieve similar CER reduction, as they happen with Encarta and Shincho domains, the MSR models have smaller ER values. We can therefore conclude that a discriminative method (in this case MSR) is superior to linear interpolation not only in terms of CER reduction, but also of having fewer side effects. This desirable result is attributed to the nature of discriminative training, which works specifically to adjust feature weights so as to minimize errors.

Figure 8 compares the three discriminative models with respect to RER/ER by plotting the best models for 8,000 training samples (i.e., models used to produce the results in Table III for 8,000 training samples) for each algorithm. Though they perform similarly in most cases, we can see some small differences: even though the boosting and perceptron algorithms have the same CER for Yomiuri and TuneUp from Table III, the perceptron is better in terms of ER; this may be due to the use of exponential loss function in the boosting algorithm which is less robust against noisy data (Hastie et al., 2001). We also observe that Yomiuri and Encarta do better in terms of side effects than TuneUp and Shincho for all algorithms, which can be explained by corpus diversity, as the former two sets are less stylistically diverse and thus more consistent within the domain.

7. CONCLUSION AND FUTURE WORK

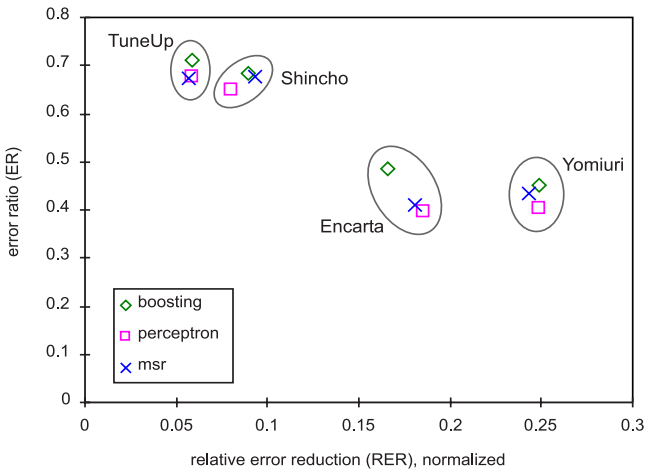

Figure 8: RER/ER plot for MSR, boosting and perceptron models (X-axis is normalized to represent relative error *rate* reduction)

In this paper, we have examined the performance of various LM adaptation methods in terms of domain similarity and diversity. We have found that (1) the notion of cross-domain similarity, measured by the cross entropy, correlates with the CER of all models (Section 6.1), and (2) the notion of in-domain diversity, measured by the self entropy, correlates with the utility of more adaptation training data for discriminative training methods (Section 6.3). In comparing discriminative methods with a MAP-based method, we have also found that (1) the former uniformly achieve better performance than the latter, not only in terms of CER reduction but also in having fewer side effects (Section 6.4), and (2) are more robust against the similarity of background and adaptation data (Section 6.2).

One important direction of future research in language modeling is an online learning scenario, i.e., to incrementally build models using incoming data for adaptation, taking all previously available data as background corpus. Such a scenario is easily conceivable in the context of adapting to a user or to a newly introduced topic. We hope that the results obtained in this paper serve as a starting point for this direction of research.

REFERENCES

Bacchiani, M. and Roark, B. 2003. Unsupervised language model adaptation. In *ICASSP 2003*. 224-227

Bacchiani, M., Roark, B., and Saraçlar, M. 2004. Language model adaptation with MAP estimation and the perceptron algorithm. In *HLT-NAACL 2004*. 21-24.

Bellagarda, J. 2001. An overview of statistical language model adaptation. In *ITRW on Adaptation Methods for Speech Recognition 2001*. 165-174.

Collins, M. 2000. Discriminative reranking for natural language parsing. In *ICML 2000*. 175-182.

Collins, M. 2002. Discriminative training methods for Hidden Markov Models: theory and experiments with perceptron algorithms. In *EMNLP 2002*. 1-8.

Dagan, I., Lee, L., and Pereira, F. 1999. Similarity-based models of co-occurrence probabilities. *Machine Learning*, 34(1-3). 43-69.

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

Freund, Y, Iyer, R., Schapire, R. E., and Singer, Y. 1998. An efficient boosting algorithm for combining preferences. In *ICML'98*. 170-178.

Gao, J., Goodman, J., Li, M., and Lee. K.-F. 2002a. Toward a unified approach to statistical language modeling for Chinese. *ACM Transactions on Asian Language Information Processing*, 1-1: 3-33.

Gao, J., Suzuki, H., and Wen, Y. 2002b. Using headword dependency and predictive clustering for language modeling. In *EMNLP 2002*. 248-256.

Gao. J., Yu, H., Yuan, W., and Xu, P. 2005. Minimum sample risk methods for language modeling. In *HLT/EMNLP 2005*. 209-216.

Gillick, L. and Cox, J. 1989. Some statistical issues in the comparison of speech recognition algorithms. In *IEEE Conference on Acoustics, Speech and Signal Processing*. 532-535.

Hastie, T., Tibshirani, R., and Friedman, J. 2001. *The Elements of Statistical Learning*. Springer-Verlag, New York.

Lee, L. 1999. Measures of distributional similarity. In *ACL 1999*. 25-32.

Manning, C.D., and Schütze, H. 1999. *Foundations of Statistical Natural Language Processing*. The MIT Press.

Mitchell, T. M. 1997. *Machine Learning*. The McGraw-Hill Companies, Inc.

Och, F.J. 2003. Minimum error rate training in statistical machine translation. In *ACL 2003*. 160-167.

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

Roark, B., Saraclar, M., and Collins, M. 2004. Corrective language modeling for large vocabulary ASR with the perceptron algorithm. In *ICASSP 2004*. 749-752.

Suzuki, H. and Gao, J. 2005a. A comparative study on language model adaptation techniques using new evaluation metrics. In *HLT/EMNLP 2005*. 265-272.

Suzuki, H. and Gao, J. 2005b. Microsoft Research IME corpus. *Microsoft Research Technical Report*, TR-2005-168.

Theodoridis, S. and Koutroumbas, K. 2003. *Pattern Recognition*. Elsevier.

Authors' addresses: Jianfeng Gao, Microsoft Research, One Microsoft Way, Redmond. WA. 98052, U.S.A. Email: jfgao@microsoft.com. Hisami Suzuki, Microsoft Research, One Microsoft Way, Redmond. WA. 98052, U.S.A. Email: hisamis@microsoft.com. Wei Yuan, Shanghai Jiao Tong University, 1954 Huashan Road, Shanghai 200230, China. Email: sunnyuanovo@sjtu.edu.cn