A Comparative Study on Language Model Adaptation Techniques Using New Evaluation Metrics

Hisami Suzuki
Microsoft Research
One Microsoft Way
Redmond WA 98052 USA
hisamis@microsoft.com

Jianfeng Gao
Microsoft Research Asia
49 Zhichun Road, Haidian District
Beijing 100080 China
jfgao@microsoft.com

Abstract

This paper presents comparative experimental results on four techniques of language model adaptation, including a maximum *a posteriori* (MAP) method and three discriminative training methods, the boosting algorithm, the average perceptron and the minimum sample risk method, on the task of Japanese Kana-Kanji conversion. We evaluate these techniques beyond simply using the character error rate (CER): the CER results are interpreted using a metric of domain similarity between background and adaptation domains, and are further evaluated by correlating them with a novel metric for measuring the side effects of adapted models. Using these metrics, we show that the discriminative methods are superior to a MAP-based method not only in terms of achieving larger CER reduction, but also of being more robust against the similarity of background and adaptation domains, and achieve larger CER reduction with fewer side effects.

1 Introduction

Language model (LM) adaptation attempts to adjust the parameters of a LM so that it performs well on a particular (sub-)domain of data. Currently, most LMs are based on the Markov assumption that the prediction of a word depends only on the preceding n–1 words, but such n-gram statistics are known to be extremely susceptible to the characteristics of training samples. This is true even when the data sources are supposedly similar: for example, Rosenfeld (1996) showed that perplexity doubled when a LM trained on the Wall Street Journal (1987-1989) was tested on the AP newswire stories of the same period. This observation, coupled with the fact that training data is available in large quantities only in selected domains, facilitates the need for LM adaptation.

There have been two formulations of the LM adaptation problem. One is the *within-domain adaptation*, in which adapted LMs are created for different topics in a single domain (e.g., Seymore and Rosenfeld, 1997; Clarkson and Robinson, 1997; Chen et al., 1998). In these studies, a domain is defined as a body of text originating from a single source, and the main goal of LM adaptation is to fine-tune the model parameters so as to improve the LM performance on a specific sub-domain (or topic) using the training data at hand.

The other formulation, which is the focus of the current study, is to adapt a LM to a novel domain, for which only a very small amount of training data is available. This is referred to as *cross-domain adaptation*. Following Bellegarda (2001), we call the domain used to train the original model the *background* domain, and the novel domain with a small amount of training data as the *adaptation* domain. 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. Count merging and linear interpolation of models are the two MAP estimation methods investigated in speech recognition experiments (Iyer et al., 1997; Bacchiani and Roark, 2003), with count merging reported to slightly outperform linear interpolation.

Discriminative approaches to 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 (Bacchiani et al., 2004) scenarios.

In this paper, we present comparative experimental results on four language model adaptation techniques and evaluate them from various angles, attempting to elucidate the characteristics of these models. The four models we compare are a maximum *a posteriori* (MAP) method and three discriminative training methods, namely the boosting algorithm (Collins, 2000), the average perceptron (Collins, 2002) and the minimum sample risk method (Gao et al., 2005). Our evaluation of these techniques is unique in that we go beyond simply comparing them in terms of character error rate (CER): we use a metric of distributional similarity to measure the distance between background and adaptation domains, and attempt to correlate it with the CER of each adaptation method. We also propose a novel metric for measuring the side effects of adapted models using the notion of *backward compatibility*, which is very important from a software deployment perspective.

Our experiments are conducted in the setting of Japanese Kana-Kanji conversion, as we believe this task is excellently suited for evaluating LMs. We begin with the description of this task in the following section.

2 Language Modeling in the Task of IME

This paper studies language modeling in the context of Asian language (e.g., Chinese or Japanese) text input. The standard method for doing this is that the users first input the phonetic strings, which are then converted into the appropriate word string 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 by 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 exhibit about 5-15% CER on 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:

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

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

Unlike speech recognition, however, there is no acoustic ambiguity 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., $P(A|W) = 1$. So the decision of Equation (1) depends solely on $P(W)$, which makes IME ideal for testing language modeling techniques. Another advantage of using IME for language modeling research is that it is relatively easy to convert W to A, which facilitates the creation of training data for discriminative learning, as described later.

From the perspective of LM adaptation, IME faces the same problem speech recognition faces: 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 *any* domain. LM adaptation in 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 Discriminative Algorithms for LM Adaptation

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 perceptron learning, and Gao et al. (2005) for the minimum sample risk method.

3.1 Definition

The following set-up, adapted from Collins (2002), was used for all three discriminative training methods:

- Training data is a set of input-output pairs. In the task of IME, we have training samples $\{A_i, W_i^R\}$, for $i = 1\ldots M$, where each A_i is an input phonetic string and each W_i^R is the reference transcript of A_i.
- 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 $\mathbf{f}(W) = \{f_0(W), f_1(W), \ldots, f_D(W)\}$. The feature $f_0(W)$ is called the *base model* feature, and is defined as the log probability that the word trigram model assigns to W. The features $f_d(W)$ for $d = 1\ldots D$ are defined as the word n-gram counts (n = 1 and 2 in our experiments) in W.
- The parameters of the model form a vector of $D + 1$ dimensions, one for each feature function, $\boldsymbol{\lambda}= \{\lambda_0, \lambda_1, \ldots, \lambda_D\}$. The likelihood score of a word string W can then be written as

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

Given a model $\boldsymbol{\lambda}$ and an input A, the decision rule of Equation (1) can then be rewritten as

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

We can obtain the number of conversion errors in W by comparing it with the reference transcript W^R using an error function $\mathrm{Er}(W^R,W)$, which is an edit distance in our case. We call the sum of error counts over the training set the *sample risk* (SR). Discriminative training methods strive to optimize the parameters of a model by minimizing SR, as in Equation (4).

$$\boldsymbol{\lambda}^* = \arg\min_{\lambda} SR(\boldsymbol{\lambda}) = \arg\min_{\lambda} \sum_{i=1\ldots M} \mathrm{Er}(W_i^R, W_i(A_i, \boldsymbol{\lambda})) \tag{4}$$

However, (4) cannot be optimized directly by regular gradient-based procedures as it is a piecewise constant function of $\boldsymbol{\lambda}$ and its gradient is undefined. The discriminative training methods described below differ in how they achieve the optimization: the boosting and perceptron algorithms approximate SR by loss functions that are suitable for optimization; the minimum sample risk method, on the other hand, uses a simple heuristic training procedure to minimize SR directly without resorting to an approximated loss function.

3.2 The boosting algorithm

The boosting algorithm we used is based on Collins (2000). Instead of measuring the number of conversion errors directly, it uses a loss function that measures the number of ranking errors, i.e., cases where an incorrect candidate W receives a higher score than the correct conversion W^R. The *margin* of the pair (W^R, W) with respect to the model λ is given by

$$M(W^R, W) = Score(W^R, \boldsymbol{\lambda}) - Score(W, \boldsymbol{\lambda}) \tag{5}$$

The loss function is then defined as

$$\mathrm{RLoss}(\boldsymbol{\lambda}) = \sum_{i=1\ldots M} \sum_{W_i\in\mathbf{GEN}(A_i)} I[M(W_i^R, W_i)] \tag{6}$$

where $I[\pi] = 1$ if $\pi \le 0$, and 0 otherwise. Note that RLoss takes into account all candidates in $\mathbf{GEN}(A)$.

Since optimizing (6) is NP-complete, the boosting algorithm optimizes its upper bound:

$$\mathrm{ExpLoss}(\boldsymbol{\lambda}) = \sum_{i=1\ldots M} \sum_{W_i\in\mathbf{GEN}(A_i)} \exp(-M(W_i^R, W_i)) \tag{7}$$

Figure 1 summarizes the boosting algorithm we used. After initialization, Step 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:

$$\delta_d = \frac{1}{2}\log\frac{C_d^+ + \varepsilon Z}{C_d^- + \varepsilon Z} \tag{8}$$

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.

```
1  Set $\lambda_0 = 1$ and $\lambda_d = 0$ for $d=1\ldots D$
2  Select a feature $f_d$ which has largest estimated im-
   pact on reducing ExpLoss of Equation (7)
3  Update $\lambda_d$ by Equation (8), and return to Step 2
```

Figure 1: The boosting algorithm

3.3 The perceptron algorithm

The perceptron algorithm can be viewed as a form of incremental training procedure that optimizes a *minimum square error* (MSE) loss function, which is an approximation of SR (Mitchell, 1997). As shown in Figure 2**,** it starts with an initial parameter setting and updates it for each training sample. We used the average perceptron algorithm of Collins (2002) in our experiments, a variation that has been proven to be more effective than the standard algorithm shown in Figure 2. Let $\lambda_d^{t,i}$ be the

```
1 Set $\lambda_0$ = 1 and $\lambda_d$ = 0 for $d$=1…$D$
2 For $t$ = 1…$T$ ($T$ = the total number of iterations)
3   For each training sample ($A_i$, $W_i^R$), $i$ = 1…$M$
4     Choose the best candidate $W_i$ from GEN($A_i$)
      according to Equation (3)
5     For each $\lambda_d$ ($\eta$ = size of learning step)
6       $\lambda_d = \lambda_d + \eta(f_d(W_i^R) - f_d(W_i))$
```

Figure 2: The perceptron algorithm

value for the *d*th parameter after the *i*th training sample has been processed in pass *t* over the training data. The average parameters are defined as

$$(\lambda_d)_{avg} = (\sum_{t=1}^{T}\sum_{i=1}^{M}\lambda_d^{t,i})/(T \cdot M). \quad (9)$$

3.4 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: a 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. This is repeated until SR stops decreasing.

Regular numerical line search algorithms cannot be applied directly because, as described above, the value of a feature parameter versus SR is not smooth and there are many local minima. MSR thus adopts the method proposed by Och (2003). Let **GEN**(A) be the set of n-best candidate word strings that could be converted from A. By adjusting λ_d for a selected feature f_d, we can find a set of intervals for λ_d within which a particular candidate word string is selected. We can compute *Er*(.) for the candidate and use it as the *Er*(.) value for the corresponding interval. As a result, we obtain an ordered sequence of *Er*(.) values and a corresponding sequence of λ intervals for each training sample. By summing *Er*(.) values over all training samples, we obtain a global sequence of SR and the corresponding global sequence of λ_d intervals. We can then find the optimal λ_d as well as its corresponding SR by traversing the sequence.

Figure 3 summarizes the MSR algorithm. See Gao et al. (2005) for a complete description of the MSR implementation and the empirical justification for its performance.

```
1 Set $\lambda_0$ = 1 and $\lambda_d$ = 0 for $d$=1…$D$
2 Rank all features by its expected impact on reduc-
  ing SR and select the top $N$ features
3 For each $n$ = 1…$N$
4   Update the parameter of $f$ using line search
```

Figure 3: The MSR algorithm

4 Experimental Results

4.1 Data

The data used in our experiments come 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). The characteristics of these domains are measured using the information theoretic notion of cross entropy, which is described in the next subsection.

For the experiment of LM adaptation, we used the training data consisting of 8,000 sentences and test data of 5,000 sentences from each adaptation domain. Another 5,000-sentence subset was used as held-out data for each domain, which was used to determine the values of tunable parameters. All the corpora used in our experiments are pre-segmented into words using a baseline lexicon consisting of 167,107 entries.

4.2 Computation of domain characteristics

Yuan et al. (2005) introduces two notions of domain characteristics: a within-domain notion of *diversity*, and a cross-domain concept of *similarity*. Diversity is measured by the entropy of the corpus and indicates the inherent variability within the domain. Similarity, on the other hand, is intended to capture the difficulty of a given adaptation task, and is measured by the cross entropy.

For the computation of these metrics, we extracted 1 million words from the training data of each domain respectively, and created a lexicon consisting of the words in our baseline lexicon plus all words in the corpora used for this experiment (resulting in 216,565 entries) to avoid the effect of out-of-vocabulary items. Given two domains *A* and

	N	Y	T	E	S
Nikkei	**3.94**	7.46	7.65	9.81	10.10
Yomiuri		**4.09**	7.82	8.96	9.29
TuneUp			**4.41**	8.82	8.56
Encarta				**4.40**	9.20
Shincho					**4.61**

Table 1: Cross entropy

Domain	Base	LI	MSR	Boost	Percep
Yomiuri	3.70	3.69	2.89	2.88	**2.85**
TuneUp	5.81	5.70	5.48	**5.47**	**5.47**
Encarta	10.24	8.64	8.39	8.54	**8.34**
Shincho	12.18	11.47	**11.05**	11.09	11.20

Table 2: CER results (%) (Base=baseline model; LI=linear interpolation)

B, we then trained a word trigram model for each domain *B*, and used the resulting model in computing the cross entropy of domain *A*. For simplicity, we denote this as *H*(*A*,*B*).

Table 1 summarizes our corpora along this dimension. Note that the cross entropy is not symmetric, i.e., H(*A*,*B*) is not necessarily the same as H(*B*,*A*), so we only present the *average cross entropy* in Table 1. We can observe that Yomiuri and TuneUp are much more *similar* to the background Nikkei corpus than Encarta and Shincho.

H(*A*,*A*) along the diagonal of Table 1 (in boldface) is the entropy of the corpus, indicating the corpus *diversity*. This quantity indeed reflects the in-domain variability of text: newspaper and encyclopedia articles are highly edited text, following style guidelines and often with repetitious content. In contrast, Shincho is a collection of novels, on which no style or content restriction is imposed. We use these metrics in the interpretation of CER results in Section 5.

4.3 Results of LM adaptation

The discriminative training procedure was carried out as follows: for each input phonetic string *A* in the adaptation training set, we produced a word lattice using the baseline trigram models described in Gao et al. (2002b). We kept the top 20 hypotheses from this lattice as the candidate conversion set **GEN**(*A*). The lowest CER hypothesis in the lattice rather than the reference transcript was used as W^R. We used unigram and bigram features that occurred more than once in the training set.

We compared the performance of discriminative methods against a MAP estimation method as the baseline, in this case the linear interpolation method. Specifically, we created a word trigram model using the adaptation data for each domain, which was then linearly interpolated at the word level with the baseline model. The probability according to the combined model is given by

$$p(w_i \mid h) = \lambda P_B(w_i \mid h) + (1-\lambda) P_A(w_i \mid h) \,,$$

where P_B is the probability of the background model, P_A the probability of the adaptation model, and the history *h* corresponds to two preceding words. λ was tuned using the held-out data.

In evaluating both MAP estimation and discriminative models, we used an *N*-best rescoring approach. That is, we created *N* best hypotheses using the baseline trigram model (*N*=100 in our experiments) for each sentence in the test data, and used adapted models to rescore the *N*-best list. The oracle CERs (i.e., the minimal possible CER given the available hypotheses) ranged from 1.45% to 5.09% depending on the adaptation domain.

The results of the experiments are shown in Table 2. We can make some observations from the table. First, all discriminative methods significantly outperform the linear interpolation (statistically significant according to the *t*-test at $p < 0.01$). In contrast, the differences among three discriminative methods are very subtle and most of them are not statistically significant. Secondly, the CER results correlate well with the metric of domain similarity in Table 1 (*r*=0.94 using the Pearson product moment correlation coefficient). This is consistent with our intuition that the closer the adaptation domain is to the background domain, the easier the adaptation task.

Regarding the similarity of the adaptation domain to the background, we also observe that the CER reduction of the linear interpolation model is particularly limited when the adaptation domain is similar to the background domain: the CER reduction of the linear interpolation model for Yomiuri and TuneUp over the baseline is 0% and 1.89% respectively, in contrast to ~22% and ~5.8% improvements achieved by the discriminative models. The discriminative methods are therefore more robust against the similarity of the adaptation and background data than the linear interpolation.

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 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 indicates that given a candidate lattice, the perceptron algorithm is effective in candidate reranking, thus making our results compatible with theirs.

5 Discussion

The results in Section 4 demonstrate that discriminative training methods for adaptation are overall superior to MAP adaptation methods. In this section, we show additional advantages of discriminative methods beyond simple CER improvements.

5.1 Using metrics for side effects

In the actual deployment of LM adaptation, one issue that bears particular importance is the number of side effects that are introduced by an adapted model. For example, consider an adapted model which achieves 10% CER improvements over the baseline. Such a model can be obtained by improving 10%, or by improving 20% *and* by introducing 10% of new errors. Clearly, the former model is preferred, particularly if the models before and after adaptation are both to be exposed to users. This concept is more widely acknowledged within the software industry as *backward compatibility* – a requirement that an updated version of software supports all features of its earlier versions. In IME, it means that all phonetic strings that can be converted correctly by the earlier versions of the system should also be converted correctly by the new system as much as possible. Users are typically more intolerant to seeing errors on the strings that used to be converted correctly than seeing errors that also existed in the previous version. Therefore, it is crucial that when we adapt to a new domain, we do so by introducing the smallest number of side effects, particularly in the case of an incremental adaptation to the domain of a particular *user*, i.e., to building a model with incremental learning capabilities.

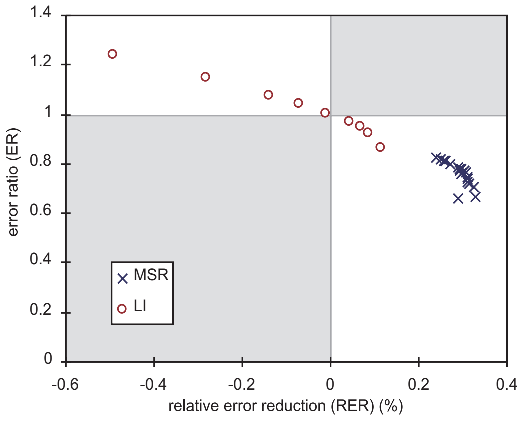

Figure 4: RER/ER plot for MSR and LI models for TuneUp domain

5.2 Error ratio

In order to measure side effects, we introduce the notion of *error ratio* (ER), which 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.

Given the notion of CER and ER, a model can be plotted on a graph as in Figure 4: the *relative error reduction* (RER, i.e., the CER difference between the background and adapted models) is plotted along the x-axis, and ER along the y-axis. Figure 4 plots the models obtained after various numbers of iterations for MSR training and at various interpolation weights for linear interpolation for the TuneUp domain. The points in the upper-left quadrant, ER>1 and RER<0, are the models that performed worse than the baseline model (some of the interpolated models fall into this category); the shaded areas (upper-right and lower-left quadrants) are by definition empty. The lower-right quadrant is the area of interest to us, as they

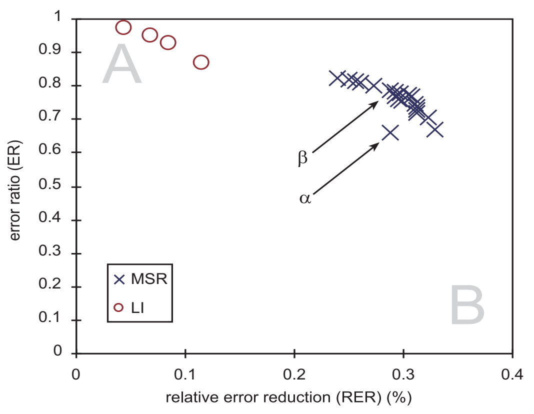

Figure 5: RER/ER plot for the models with ER<1 and RER>0 for TuneUp domain. See Figure 8 for the description of *α* and *β*

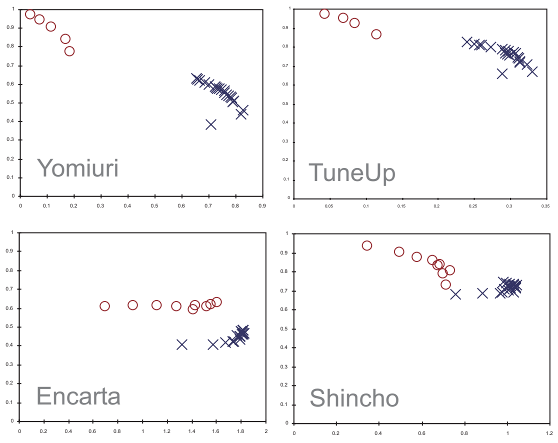

Figure 6: RER/ER plot for all four domains
x-axes: RER (%); y-axes: ER
○: linear interpolation models; ×:MSR models

represent the models that led to CER improvements; we will focus only on this area now in Figure 5.

In this figure, a model is considered to have fewer side effects when the ER is smaller at the same RER (i.e., smaller value of *y* for a fixed value of *x*), or when the RER is larger at the same ER (i.e., larger value of *x* at the fixed *y*). That is, the closer a model is plotted to the corner B of the graph, the better the model is; the closer it is to the corner A, the worse the model is.

5.3 Model comparison using RER/ER

From Figure 5, we can clearly see that MSR models have better RER/ER-performance than linear interpolation models, as they are plotted closer to the corner B. Figure 6 displays the same plot for all four domains: the same trend is clear from all graphs. 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 error.

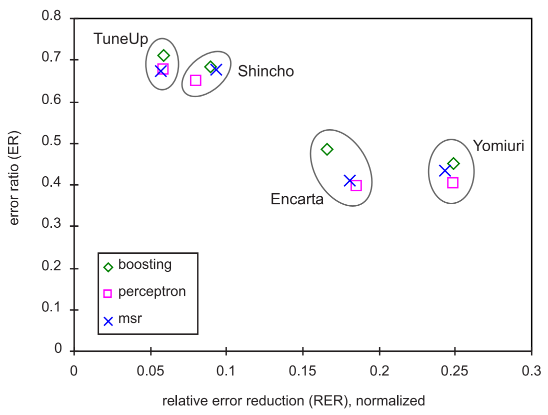

Figure 7: RER/ER plot for MSR, boosting and perceptron models (X-axis is normalized to represent relative error *rate* reduction)

Figure 7 compares the three discriminative models with respect to RER/ER by plotting the best models (i.e., models used to produce the results in Table 1) for each algorithm. We can see that even though the boosting and perceptron algorithms have the same CER for Yomiuri and TuneUp from Table 2, 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 set is less stylistically diverse and thus more consistent within the domain.

5.4 Overfitting and side effects

The RER/ER graph also casts the problem of *overfitting* in an interesting perspective. Figure 8 is derived from running MSR on the TuneUp test corpus, which depicts a typical case of overfitting: the CER drops in the beginning, but after a certain number of iterations, it goes up again. The models indicated by *α* and *β* in the graph are of the same CER, and as such, these models are equivalent. When plotted on the RER/ER graph in Figure 5,

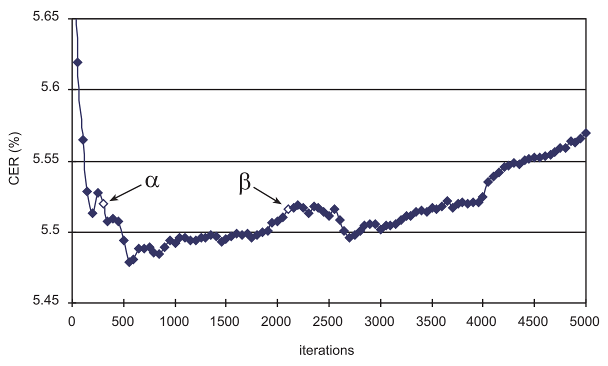

Figure 8: MSR test error curve for TuneUp

however, it is clear that the overfit model β has the worse ER than the non-overfit counterpart α. In other words, models α and β have the same CER, but they are not equivalent: model β is not only worse in light of containing more features, but also in terms of causing more side effects.

6 Conclusion and Future Work

We have presented a comparison of three discriminative learning approaches with a MAP estimation method in the task of LM adaptation for IME. We have shown that all discriminative models are significantly better than the linear interpolation method, in that they achieve larger CER reduction with fewer side effects across different domains.

One direction of future research is to apply this technique to an incremental learning scenario, i.e., to incrementally build models using incoming data for adaptation, taking all previously available data as background corpus. The new metric for backward compatibility we proposed in the paper will play a particularly important role in such a scenario.

Acknowledgements

We would like to thank Kevin Duh, Gary Kacmarcik, Eric Ringger, Yoshiharu Sato and Wei Yuan for their help at various stages of this research.

References

Bacchiani, M. and B. Roark. 2003. Unsupervised Language Model Adaptation. *Proceedings of ICASSP*, pp.224-227.

Bacchiani, M., B. Roark and M. Saraclar. 2004. Language Model Adaptation with MAP Estimation and the Perceptron Algorithm. *Proceedings of HLT-NAACL*, pp.21-24.

Bellegarda, J.R. 2001. An Overview of Statistical Language Model Adaptation. *ITRW on Adaptation Methods for Speech Recognition*, pp. 165-174.

Chen, S.F., K. Seymore and R. Rosenfeld. 1998. Topic Adaptation for Language Modeling Using Unnormalized Exponential Models. *Proceedings of ICASSP*.

Clarkson, P.R., and A.J. Robinson. 1997. Language Model Adaptation Using Mixtures and an Exponentially Decaying Cache. *Proceedings of ICASSP*.

Collins, M. 2000. Discriminative Reranking for Natural Language Parsing. *ICML 2000*.

Collins, M. 2002. Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithm. *Proceedings of EMNLP*, pp.1-8.

Freund, Y., R. Iyer, R.E. Shapire and Y. Singer. 1998 An Efficient Boosting Algorithm for Combining Preferences. *ICML'98*.

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

Gao, J, H. Suzuki and Y. Wen. 2002b. Using Headword Dependency and Predictive Clustering for Language Modeling. *Proceedings of EMNLP*: 248-256.

Gao, J., H. Yu, P. Xu and W. Yuan. 2005. Minimum Sample Risk Methods for Language Modeling. *Proceedings of EMNLP 2005*.

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

Iyer, R., M. Ostendorf and H. Gish. 1997. Using Out-of-Domain Data to Improve In-Domain Language Models. *IEEE Signal Processing Letters*, 4-8: 221-223.

Mitchell, Tom M. 1997. *Machine learning*. The McGraw-Hill Companies, Inc.

Och, F. J. 2003. Minimum Error Rate Training in Statistical Machine Translation. *Proceedings of ACL*: 160-167.

Roark, B., M. Saraclar and M. Collins. 2004. Corrective Language Modeling for Large Vocabulary ASR with the Perceptron Algorithm. *Proceedings of ICASSP*: 749-752.

Rosenfeld, R. 1996. A Maximum Entropy Approach to Adaptive Statistical Language Modeling. *Computer, Speech and Language*, 10: 187-228.

Seymore, K. and R. Rosenfeld. 1997. Using Story Topics For Language Model Adaptation. *Proceedings of Eurospeech '97*.

Yuan, W., J. Gao and H. Suzuki. 2005. An Empirical Study on Language Model Adaptation Using a Metric of Domain Similarity. *Proceedings of IJCNLP 05*.