Data Sampling and Dimensionality Reduction Approaches for Reranking ASR Outputs Using Discriminative Language Models

Eriç Dikici¹, Murat Semerci², Murat Saraclar¹, Ethem Alpaydın²

¹Department of Electrical and Electronics Engineering, Boğaziçi University, Istanbul, Turkey
²Department of Computer Engineering, Boğaziçi University, Istanbul, Turkey
{erinc.dikici,murat.semerci,murat.saraclar,alpaydin}@boun.edu.tr

Abstract
This paper investigates various approaches to data sampling and dimensionality reduction for discriminative language models (DLM). Being a feature based language modeling approach, the aim of DLM is to rerank the ASR output with discriminatively trained feature parameters. Using a Turkish morphology based feature set, we examine the use of online Principal Component Analysis (PCA) as a dimensionality reduction method. We exploit ranking perceptron and ranking SVM as two alternative discriminative modeling techniques, and apply data sampling to improve their efficiency. We obtain a reduction in word error rate (WER) of 0.4%, significant at $p < 0.001$ over the baseline perceptron result.

Index Terms: discriminative language modeling, dimensionality reduction, data sampling, ranking perceptron, ranking SVM

1. Introduction
An ASR system outputs multiple transcription hypotheses for a given acoustic speech signal. These are usually ordered with respect to their recognition scores, but the hypothesis having the highest score is not necessarily the most accurate transcription. Classical DLM approaches view the problem as discrimination of better transcriptions from worse in a binary classification setting, while trying to optimize an objective function that is directly related to the word error rate (WER). A better alternative would be to specify the DLM task as a reranking problem, so that by reordering these outputs, the hypothesis with the least number of errors is at the top.

The perceptron algorithm and its variants approaching the problem from both classification and reranking points of view are the most widely used techniques in discriminative training of language models [1, 2].

Another generalized linear classifier which has been proven useful in binary classification tasks is the Support Vector Machine (SVM). The SVM is optimal in the sense of finding a maximal margin that separates the two classes. However, SVM-based techniques can be computationally demanding when the number of training examples is large and the feature dimension is high. Versions of SVMs adapted for ranking and reranking have also been proposed [3].

Zhou et al. [4] compare the accuracies and training times of four discriminative algorithms, namely perceptron, boosting, ranking SVMs and minimum sample risk. Because of the long training durations, the authors had to reduce the length of $N$-best lists from 100 to 20 for their ranking SVM implementation. Despite this limitation, they show that ranking SVM is able to obtain a more generalizable model.

Oba et al. [5] investigate different hypothesis selection schemes from an $N$-best list, and argue that the selection of hypotheses based on WER should be preferred over the recognition score. Using the perceptron algorithm, they achieve an accurate and compact corrective model.

Crammer and Singer [6] propose a perceptron ranking (PRanking) algorithm that divides the space into regions bounded by threshold levels. The study by Shen and Joshi [2] includes a review of reranking and two perceptron based algorithms to decrease data complexity and training time.

In this study we propose several data sampling and feature selection methods in order to increase the accuracy, robustness and computational efficiency of the discriminative model. We attempt dimensionality reduction to seek for an appropriate feature subspace projection. We compare the behavior and performance of averaged perceptron, ranking perceptron and ranking SVM algorithms. To make algorithmically efficient use of the ranking procedure, we put forward different data sampling and grouping approaches.

This paper is organized as follows: In Section 2, we give brief information on the methods used in the study. The experimental setup and results are given in Section 3. Section 4 concludes the paper with discussions and remarks on future work.

2. Methods

2.1. Discriminative language modeling using linear models
DLM, which is a complementary method to baseline generative language modeling, is used to estimate language model parameters in a discriminative setting.

The training inputs to a DLM are examples extracted from $N$-best lists, i.e., the score-wise ordered outputs of an ASR system using a generative language model. In the linear model, each example of the training set is represented by a feature vector of the acoustic input ($x$) and the candidate hypothesis ($y$), denoted by $\Phi(x, y)$. We would like to learn $w$, the vector of weights associated with these features. In testing, the highest scoring hypothesis that maximizes the inner product $\langle w, \Phi(x, y) \rangle$ is chosen. We use three discriminative training algorithms, which will be explained in the following sections.

2.1.1. Averaged perceptron
We use a variant of the perceptron algorithm as outlined in [1]. The idea of the perceptron algorithm is to penalize features associated with the current best hypothesis and to reward features associated with the gold-standard hypothesis. We define the gold standard of the $i$-th utterance ($y_i$) as the hypothesis having the lowest WER, i.e. the oracle, and the current best as $z_i = \arg\max_{z \in H_i} \langle w, \Phi(x_i, z) \rangle$. 

Copyright © 2011 ISCA

1461

28 – 31 August 2011, Florence, Italy
where $x_i$ is the input of the i-th utterance and $H_i$ is the list of all possible hypotheses for this utterance. For each training example $(x_i, y_i)$, if $z_i$ differs from $y_i$, the j-th feature weight is increased by $\Phi_j(x_i, y_i)$ and decreased by $\Phi_j(x_i, z_i)$:

$$w = w + \Phi(x_i, y_i) - \Phi(x_i, z_i)$$

The algorithm requires several epochs (passes) over the training set and it has been shown that three epochs are adequate for practical purposes [1]. Once the training is completed, the weights are averaged over the number of utterances ($I$) and epochs ($T$) to increase model robustness: $w_{avg} = \frac{1}{IT} \sum_{i,t} w_i$.

### 2.1.2. Ranking Perceptron

The canonical perceptron and SVM algorithms are for binary classification. However, it is more natural to regard the DLM task as reranking of hypotheses generated by the ASR output, since the WER provides a target ranking for each possible transcription. This problem is similar to ordinal regression in that all examples of the training set $\Phi(x, y)$ are given a rank ($r$) instead of a class label. However, unlike the ordinal regression case, in reranking, the ranks are comparable only between the examples of the same utterance ($N$-best list). The uneven margin perceptron [2] separates examples of the same utterance with respect to their pairwise difference. If $\Phi(x_a, y_a)$ and $\Phi(x_b, y_b)$ are in the same $N$-best list and have ranks $r_a$ and $r_b$, the uneven margins are determined by the margin function $g(r_a, r_b)$ which depends on the ranks of the pair:

$$g(r_a, r_b) = \frac{1}{r_a} - \frac{1}{r_b}$$

The pairwise distance between the examples must be greater than the margin function times a margin multiplier $\tau$, and the margin-rank relations must be preserved:

$$r_a > r_b \iff \langle w, \Phi(x_a, y_a) - \Phi(x_b, y_b) \rangle \geq \tau g(r_a, r_b)$$

$$r_a > r_b \Rightarrow g(r_a, r_c) > g(r_a, r_b) > g(r_b, r_c)$$

In this study, we apply the weight updates by introducing a learning rate, $\eta$, which is decreased by multiplying with a decay rate of $\gamma < 1$ at the end of each epoch.

$$w = w + \eta g(r_a, r_b)(\Phi(x_a, y_a) - \Phi(x_b, y_b))$$

The weights are averaged as in Section 2.1.1.

### 2.1.3. Ranking SVM

Similar to the ranking perceptron algorithm, the goal here is to estimate a weight vector $w$ such that for any two hypotheses $a$ and $b$ belonging to the same $N$-best list $H_i$,

$$r_a > r_b \iff \langle w, \Phi(x_a, y_a) \rangle > \langle w, \Phi(x_b, y_b) \rangle$$

The ranking SVM algorithm is a modification of the classical SVM setup to handle the reranking problem, defined as

$$\min_{w} \frac{1}{2} \langle w, w \rangle + \frac{C}{|P|} \sum_{(a,b) \in P} \xi_{ab}$$

s.t. $\forall (a,b) \in P$, $\langle w, \Phi(x_a, y_a) \rangle - \langle w, \Phi(x_b, y_b) \rangle \geq 1 - \xi_{ab}$

(1)

Here, $C$ is the trade-off value and $P$ is the set of $(a, b)$ pairs for which $r_a > r_b$. The constraint in Equation 1 implies that the ranking algorithm can also be viewed as an SVM classification problem on pairwise difference vectors. In that sense, the algorithm tries to find a large margin linear function which minimizes the number of pairs of training sample set to be swapped with respect to the desired ranking [3].

### 2.2. Data Selection

Since the ranking SVM formulation brings constraints of pairwise examples having different ranks, the problem becomes more complex as the sample size and the number of unique ranks are increased. In order to relax some of these constraints and to decrease the CPU times, we can sample the input data. The most straightforward sampling scheme would be to decrease the size of the $N$-best list, as in [4]. Sampling randomly from the list also provides a viable reference.

In this study we propose three different data selection approaches. All approaches start by sorting $N$-best lists in ascending number of word errors. If two hypotheses have the same error, the one having a higher recognition score gets the lower order (index). In the first approach, we select $n = \{2, 3, 5, 9\}$ instances in uniform intervals from this ordered list. We call this approach uniform sampling, and denote it with US-$n$. For instance in US-5 with a 50-best list, the hypotheses with the indices 1, 13, 25, 37 and 50 are selected (The best and the worst hypotheses are always in the shortlist).

In the second approach, we group the hypotheses having the same unique WER and select the one having the highest score as the representative example of that group. We name this approach rank grouping (RG). For the first and second approaches, the rank of the example is the same as its WER.

Finally, in the third approach, the rank clustering (RC), we generate artificial rank clusters. We try RC-3×3 and RC-3×10, where we select 3 and 10 examples, respectively, from the top, middle and bottom of the ordered list.

Each uniform sampling scheme with increasing $n$ also contains the instances from the previous ones. In the US scheme, we aim to decrease the number of instances. In RG, we keep one representative instance from all available ranks. In RC, we decrease both the number of instances and ranks.

A simplified example of these sampling schemes is presented in Table 1. The first column denotes the hypotheses ordered with respect to their number of word errors (WE), shown in the second column. In the columns that follow, the rank values associated with these hypotheses are given.

<table>
<thead>
<tr>
<th>Order</th>
<th>WE</th>
<th>US-2</th>
<th>US-3</th>
<th>US-5</th>
<th>RG-1</th>
<th>RC-3×2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>8</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>9</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
</tbody>
</table>

### 2.3. Dimensionality reduction via Online PCA

Reducing the number of dimensions eases the curse of dimensionality and drastically decreases the required resources. It also provides a better generalization of the data by eliminating the effects of noise. In our problem, where the feature vector is very high dimensional, applying a dimensionality reduction technique can provide a better characterization for the linear classification of ASR output hypotheses.

Principal Component Analysis (PCA) is a linear transformation method such that the data is mapped into a space where the mapped data is uncorrelated and the maximal amount of variance is preserved by using as small a number of coordinates as possible. It is an offline algorithm where the dataset...
is processed in a single batch. A number of online PCA algorithms have also been proposed in the literature [7]. The online PCA network consists of the input and output nodes, the weights between the input and output layers and the lateral weights between the output nodes. The network has \( d \) input units and \( m \) output units, where \( m \leq d \). The weight vector for the output node \( k \) is \( \mathbf{v}_k = (v_{1k}, v_{2k}, \ldots, v_{dk})^T \). The lateral weights \( \mathbf{u}_{lk} \) are organized such that there is a connection between the output units \( l \) to \( k \) if \( l < k \). Let \( \phi \) be a \( d \)-dimensional input vector with zero mean. Then the output value \( (\mathbf{o}_k) \) can be evaluated as

\[
\mathbf{o}_k = \mathbf{v}_k \cdot \phi + \sum_{l<k} \mathbf{u}_{lk} \mathbf{o}_l
\]

Given a learning rate \( \eta \), the input and lateral weights are updated by Hebbian and anti-Hebbian rules, respectively [7].

\[
\Delta \mathbf{v}_{jk} = \eta \phi_j \mathbf{o}_k \hspace{1cm} j = 1, 2, \ldots, d, \hspace{0.5cm} k = 1, 2, \ldots, m
\]

\[
\Delta \mathbf{u}_{lk} = -\eta \phi_k \mathbf{o}_l \hspace{1cm} l < k
\]

### 3. Experiments

#### 3.1. Experimental setup

In this study, we apply discriminative language modeling techniques on a Turkish broadcast news transcription task. The dataset we use in our experiments includes approximately 194 hours of speech recorded from news channels. We divide the data into disjoint training (188 hours), held-out (3.1 hours) and test (3.3 hours) subsets. These subsets contain a total of 105355, 1947 and 1784 utterances, respectively.

The acoustic model and the baseline ASR system were developed with the AT&T tools\(^1\) and the generative language models were built using the SRILM toolkit\(^2\). These setups are the same as mentioned in [8]. We use a statistical morph based ASR setup, as morphs obtained via the Morfessor algorithm\(^3\) were shown to be more suitable and effective for the agglutinative nature of Turkish [9].

We use 50-best lists of hypotheses extracted from the ASR lattice output. The first element of the feature vector, \( \Phi_0(x, y) \), is the log-probability of \( x \) in the lattice obtained from the baseline recognizer, and shows the contribution of baseline acoustic and language models. The rest of the feature vector consists of counts showing the number of times a particular morph occurs in the hypothesis. Having a total of 45889 morphs, we have highly sparse feature vectors.

Using this setup, the generative baseline and oracle word error rates on the held-out set are 22.93% and 14.18%, and on the test set are 22.37% and 13.92% respectively.

#### 3.2. Perceptron baseline

We train the language model using the averaged perceptron algorithm as mentioned in Section 2.1.1. The weight \( \alpha_0 \) associated with the zero-th feature \( \Phi_0(x, y) \) is fixed during the weight updates. For the optimal choice of \( \alpha_0 \) and the number of epochs, the error rate on the held-out set is 22.14%, implying a reduction of 0.8% over the generative baseline. We observe that in the final model only 20.1K of the original 45.9K features are kept (have nonzero weights). The same model yields 21.84% error on the test data.

### 3.3. Dimensionality reduction

In our experiments, we use a modified version of the online PCA algorithm. First, we apply a count based threshold on the data to eliminate uncommon features. Thus, any possible noise due to these features is ignored and the evaluation time and memory storage are saved. Then we apply the averaged perceptron on this transformed feature set.

With one epoch of averaged perceptron training, the held-out word error rates obtained via changing the threshold \( \theta \in \{50, 100\} \) and setting the output dimensions \( m \in \{1000, 500\} \) are all 22.93%, a result comparable to the generative baseline. We also merge online PCA output features with the input features and feed them together to a linear model. This yields again the same result. We see that the PCA features dominate the sparse input features and the linear model is incapable of making use of the extra information from the input features.

To investigate whether count based thresholding has an adverse effect on these results, we repeat the experiments without applying online PCA. Using four different thresholds (shown in Table 2) and eliminating the features which occur less than these levels, we see that the performance of the perceptron is not degraded drastically. In some cases, such as with the threshold of 500, we even obtain slightly better WER.

<table>
<thead>
<tr>
<th>Threshold (( \theta ))</th>
<th>Input Dims (( d ))</th>
<th>WER (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>45889</td>
<td>22.35</td>
</tr>
<tr>
<td>50</td>
<td>23396</td>
<td>22.34</td>
</tr>
<tr>
<td>100</td>
<td>17949</td>
<td>22.35</td>
</tr>
<tr>
<td>500</td>
<td>9680</td>
<td>22.28</td>
</tr>
<tr>
<td>1000</td>
<td>6875</td>
<td>22.36</td>
</tr>
</tbody>
</table>

### 3.4. Ranking Perceptron

As mentioned in Section 1, the DLM task is more suitable to be viewed as a reranking problem. In our ranking experiments, we choose the rank of each hypothesis to be equal to the number of errors. Thus, any hypothesis having less errors becomes a lower rank (higher priority) example in the feature space.

In training of the averaged ranking perceptron, the weight \( \alpha_0 \) is fixed during the weight updates. Grid search is applied over \( \eta, \gamma, \) and \( \tau \). The model is trained over 10 epochs. The optimal parameter set which gives the lowest held-out set error is used as the final model. The lowest error rate obtained on the held-out set is 21.70%.

### 3.5. Ranking SVM

We use the SVM\(^{rank} \) tool\(^4\), which provides a fast implementation of the ranking SVM algorithm. Table 3 shows the WERs obtained from the held-out set, with varying trade-off (C) values. It suggests that ranking SVM gives comparable results with the averaged perceptron algorithm.

<table>
<thead>
<tr>
<th>C=1</th>
<th>C=100</th>
<th>C=1000</th>
<th>C=10000</th>
<th>C=200000</th>
</tr>
</thead>
<tbody>
<tr>
<td>22.34</td>
<td>22.26</td>
<td>22.13</td>
<td>22.14</td>
<td>22.11</td>
</tr>
</tbody>
</table>

---

1\(^{\text{http://www2.research.att.com/~fsmt/tools/}}\)
2\(^{\text{http://www.speech.sri.com/projects/srilm/}}\)
3\(^{\text{http://www.cs.hut.fi/projects/morpho/}}\)
4\(^{\text{http://www.cs.cornell.edu/people/tj/svm_light/svm_rank.html}}\)
3.6. Data selection

In Table 4, we present held-out WERs of the three data sampling approaches for the three algorithms, with an optimal parameter selection that yields the lowest error rates on the same set. Results of the 50-best setup is also given for comparison.

Table 4: Sampling schemes held-out WER (%)  
<table>
<thead>
<tr>
<th>Method</th>
<th>Avg. Per.</th>
<th>SVMrank</th>
<th>PerRank</th>
</tr>
</thead>
<tbody>
<tr>
<td>All (50)</td>
<td>22.14</td>
<td>22.11</td>
<td>21.70</td>
</tr>
<tr>
<td>US-2</td>
<td>22.10</td>
<td>22.66</td>
<td>22.07</td>
</tr>
<tr>
<td>US-3</td>
<td>22.14</td>
<td>22.21</td>
<td>21.98</td>
</tr>
<tr>
<td>US-5</td>
<td>22.22</td>
<td>21.86</td>
<td>21.87</td>
</tr>
<tr>
<td>US-9</td>
<td>22.18</td>
<td>21.89</td>
<td>21.97</td>
</tr>
<tr>
<td>RG-1</td>
<td>22.26</td>
<td>22.11</td>
<td>21.99</td>
</tr>
<tr>
<td>RC-3×3</td>
<td>22.18</td>
<td>22.12</td>
<td>21.82</td>
</tr>
<tr>
<td>RC-3×10</td>
<td>22.28</td>
<td>21.91</td>
<td>21.73</td>
</tr>
</tbody>
</table>

The first observation is that in the averaged perceptron, using only two examples (the best and worst in terms of WER) is as good as using all the examples. This finding corroborates the result presented in [5]. Adding more examples to the training set does not change the error rate. Unlike the perceptron, SVMrank benefits from increasing number of examples. The best result obtained here is 21.86% with the US-5 sampling scheme. This value is better than using 5-best lists (22.11%), or choosing 5 examples randomly (22.24%). The RG and RC schemes also provide comparable or better results than the baseline. Furthermore, the ranking SVM is able to learn a more generalizable model than the perceptron baseline, suggested by the fact that it performs better on an unseen test set (21.56% vs 21.84%), as shown in Table 5. This result is significant at $p = 0.027 < 0.05$ as measured by the NIST MAPSSWE test.

The averaged ranking perceptron outperforms the other ranking algorithms almost in all cases. As in SVMrank, the accuracy improves by around 0.4% as the number of selected hypotheses increases. The test error improvement for 50-best over the averaged perceptron is also significant at $p < 0.001$.

Table 5: Test set WER (%)  
<table>
<thead>
<tr>
<th>Method</th>
<th>Avg. Per.</th>
<th>SVMrank</th>
<th>PerRank</th>
</tr>
</thead>
<tbody>
<tr>
<td>All (50)</td>
<td>21.84</td>
<td>21.57</td>
<td>21.46</td>
</tr>
<tr>
<td>US-5</td>
<td>22.01</td>
<td>21.56</td>
<td>21.78</td>
</tr>
</tbody>
</table>

4. Summary and Discussion

In this paper, we presented some approaches to enhance discriminative language modeling performance for Turkish broadcast news transcription. The results of three training methods, averaged perceptron, ranking perceptron and ranking SVM have been compared. We apply online PCA as a dimensionality reduction technique on the sparse feature set, and some sample selection strategies to decrease the complexity of ranking SVM. Overall, we obtained a WER reduction of 0.3-0.4% on the held-out and test sets.

We see that though online PCA does not degrade accuracy, it does not improve it either. Actually in terms of processing time, the system becomes slower because though we decrease dimensionality, we also lose sparsity. We are currently working on (possibly nonlinear) dimensionality reduction methods which will generate sparse representations.

The performance of the ranking SVM model is comparable to and sometimes better than that of the averaged perceptron. We also see that with the perceptron, sampling approaches do help decrease the error rates.

The ranking perceptron has the lowest error rates, but it needs more hypotheses. However, it takes less time to train the ranking perceptron than to train the ranking SVM. Therefore, it is preferable than the other methods.

Even though the SVMrank toolkit provides a fast and efficient implementation of the algorithm, it is much slower than the perceptron for training, which takes about one minute. Table 6 shows side information about some of the experiments using the SVMrank setup, with a fixed $C=1000$. We see that, by an intelligent sampling technique from the $N$-best list, it is possible to decrease the number of training instances and thus the CPU times, while keeping the WERs still in a tolerable region.

Table 6: Training CPU times for fixed $C=1000$  
<table>
<thead>
<tr>
<th>Method</th>
<th>Number of instances</th>
<th>CPU hours</th>
<th>Held-out WER (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>4939368</td>
<td>25:00</td>
<td>22.26</td>
</tr>
<tr>
<td>US-5</td>
<td>518234</td>
<td>1:42</td>
<td>22.05</td>
</tr>
<tr>
<td>RG-1</td>
<td>460277</td>
<td>1:49</td>
<td>22.01</td>
</tr>
<tr>
<td>RC-3×3</td>
<td>896102</td>
<td>0:43</td>
<td>22.18</td>
</tr>
</tbody>
</table>

We are currently investigating the effect of using larger $N$-best and feature sets. Our future directions include using the perceptron and ranking SVM for feature selection and data summarization.

5. Acknowledgements

This research is supported in part by TUBITAK under Project number 109E142 and by the Turkish State Planning Organization (DPT) under the TAM Project number 2007K120610. Murat Saraçlar is supported by the TUBA GEBIP Award. The authors would like to thank Ebru Arısoy for the baseline DLM setup. The numerical calculations reported in this paper were performed at TUBITAK ULAKBIM, High Performance and Grid Computing Center (TR-Grid e-Infrastructure).

6. References