Discriminative Rescoring Based on Minimization of Word Errors for Transcribing Broadcast News

Akio Kobayashi, Takahiro Oku, Shinichi Homma, Shohei Sato, Toru Imai and Tohru Takagi

NHK (Japan Broadcasting Corporation)
Science and Technical Research Laboratories
{kobayashi.a-fs,oku.t-le,homma.s-fc,satou.s-gu,imai.t-mq,takagi.t-fo}@nhk.or.jp

Abstract
This paper describes a novel method of rescoring that reflects tendencies of errors in word hypotheses in speech recognition for transcribing broadcast news, including ill-trained spontaneous speech. The proposed rescoring assigns penalties to sentence hypotheses according to the recognition error tendencies in the training lattices themselves using a set of weighting factors for feature functions activated by a variety of linguistic contexts. Word hypotheses with low possibilities of correct words are penalized while those with high possibilities are rewarded by the weighting factors. We introduce two types of training techniques to obtain the factors. The first is based on conditional random fields (CRFs), and the second is based on the minimization of word errors, which implicitly reduces expected word errors. The results of transcribing Japanese broadcast news achieved a word error rate (WER) of 10.38%, which was a 6.06% reduction relative to conventional lattice rescoring.

1. Introduction
The recent progress being made in the field of corpus-based spoken-language processing has led to significantly successful applications in the real world. NHK (Japan Broadcasting Corp.) has developed a real-time large vocabulary continuous speech recognition (LVCSR) system for closed-captioning broadcast news [1]. Although the system has definite benefits in accuracy when used under spontaneous speech conditions, it has not been sufficiently precise in practical applications. From the perspective of language modeling, degraded accuracy results from the mismatch between actual speech and the training corpora, which consists mostly of written manuscripts. For example, the expressions commonly observed in actual speech with small frequencies in the training corpora tend to be errors. Meanwhile, the errors in transcribed lattices have useful information for eliminating mismatches and reducing word error rates (WERs) without depending on word frequencies. Then, accuracy should be improved when the system penalizes word hypotheses with low possibilities of correct words and rewards those with high possibilities. Hence, the behaviors of word hypotheses as either errors or correct words provide a clue to error correction, and discriminative training could capture such characteristics or error tendencies in word hypotheses. Two major approaches have thus far been proposed for handling errors. One typical approach is based on discriminative training as seen in statistical acoustic or language modeling [2, 3]. The other is based on the so-called reranking [4] method, which calculates penalties for sentence hypotheses in n-best lists using particular feature functions [5, 6].

We propose a novel rescoring method in manner of discriminative training, which explicitly minimizes the expected word errors in the lattices, and the method reflects information about erroneous words by using a set of linguistic features. It is different to the previous work in that reranking with the method tries to maximize the scores of references for sentence hypotheses in the n-best lists without explicitly regarding word errors. Our proposed rescoring scheme assigns penalties to the sentence hypotheses using a set of feature functions and their corresponding weighting factors. The feature functions are activated by a variety of linguistic contexts such as word n-grams and class n-grams. We introduce two types of training techniques to obtain the weighting factors. The first is based on a logistic regression such as conditional random fields (CRFs) [7], and the second is based on a minimum word error (MWE) criterion used in discriminative acoustic modeling [2].

We will present the results obtained using our proposed method of discriminative rescoring on a Japanese broadcast news transcription, and compare them with conventional rescoring and a perceptron algorithm [5], which is commonly used in reranking tasks.

2. Discriminative Rescoring
The method of discriminative rescoring and the techniques to train it on the lattices are described in this section. This approach is characterized by penalizing sentence hypotheses with feature functions and their weighting factors.

2.1. Discriminative Rescoring
The discriminative function that generally yields the score of a sentence hypothesis, \( w \), from an input, \( x \), which is a portion of speech, is described as

\[
g(w|x) = \lambda_0 f_0(w|x) + \lambda_1 f_1(w),
\]

where \( f_0(w|x) \) is the logarithmic acoustic score of \( x \), \( f_1(w) \) is the logarithmic language score, and \( \lambda_0 \) and \( \lambda_1 \) correspond to the weighting factors for these scores. The best hypothesis is given by

\[
w^* = \arg \max_{w \in \mathcal{L}} g(w|x),
\]

where \( \mathcal{L} \) is a word lattice.

A set of feature functions is introduced as an alternative discriminative function for Eq. (1) to reflect error information in the sentence hypotheses.

\[
g(w|x) = \lambda_0 f_0(w|x) + \lambda_1 f_1(w) + \sum_{i=2}^{T} \lambda_i f_i(w),
\]

where \( f_0(w|x) \) is the logarithmic acoustic score of \( x \), \( f_1(w) \) is the logarithmic language score, and \( \lambda_0 \) and \( \lambda_1 \) correspond to the weighting factors for these scores. The best hypothesis is given by

\[
w^* = \arg \max_{w \in \mathcal{L}} g(w|x),
\]

where \( \mathcal{L} \) is a word lattice.

A set of feature functions is introduced as an alternative discriminative function for Eq. (1) to reflect error information in the sentence hypotheses.

\[
g(w|x) = \lambda_0 f_0(w|x) + \lambda_1 f_1(w) + \sum_{i=2}^{T} \lambda_i f_i(w),
\]
where \( f_i \) \((i = 2 \ldots I)\) denotes a feature function, and \( \lambda_i \) is a corresponding weighting factor. It should be noted that the feature functions only depend on the linguistic contexts of the sentence hypotheses. Since the computational cost involved in calculating the number of insertion errors from that of the correct word hypotheses. The discriminative function derived by Eq. (3) is expected to reduce WERs by penalizing the word hypotheses with low possibilities of correct words while rewarding those with high possibilities by using the weighting factors.

The weighting factors are acquired as a result of minimizing the loss function by using a quasi-Newton method such as the L-BFGS algorithm [8]. The performance of discriminative rescoring depends on the design of the loss function, and we examined several types of loss functions that will be described in the following section.

### 2.2. Training of Weighting Factors

#### 2.2.1. Training Based on Logistic Regression

Let \( w_{m,n} \) \((m = 1, \ldots, M, n = 1, \ldots, N)\) denote the n-th sentence hypothesis in the lattice, \( L_m \), of the m-th training utterance, \( x_m \), and \( w_{m,0} \) be a corresponding reference. The conditional probability of the reference, \( w_{m,0} \), is given by

\[
q_\lambda(w_{m,0}|x_m) = \frac{\exp \sum_{x_{m,n} \in L_m} \lambda_i f_i(w_{m,0})}{\sum_{x_{m,n} \in L_m} \exp \sum_{x_{m,n} \in L_m} \sum_{j} \lambda_j f_j(w_{m,n})}, \tag{4}
\]

A simple loss function is defined on the basis of the calculation of the log likelihood of the references in the training data.

\[
\text{Loss}_1(\Lambda) = - \sum_{m=1}^{M} \sum_{n \in L_m} \log q_\lambda(w_{m,0}|x_m), \tag{5}
\]

where \( \text{Loss}_1 \) is considered as a log likelihood between the score of the reference and those of all the hypotheses, i.e., CRFs [7]. The weighting factors are obtained by maximizing the likelihood of the reference against the other hypotheses on the lattices according to the L-BFGS algorithm.

#### 2.2.2. Training Based on Minimization of Word Errors

The set of weighting factors, \( \Lambda \), given by Eq. (5) would not always be the best parameters for the hypotheses in terms of WERs, since the function does not explicitly reflect the WERs of the sentence hypotheses.

Therefore, let us define a loss function, which reflects the WERs of all the sentence hypotheses in the lattices.

\[
\text{Loss}_2(\Lambda) = - \sum_{m=1}^{M} \sum_{n \in L_m} \text{Acc}(w_{m,k}), \tag{6}
\]

where \( \text{Acc}(w_{m,k}) \) is the raw accuracy calculated by subtracting the number of insertion errors from that of the correct word hypotheses. Since the computational cost involved in calculating the raw accuracies for all hypotheses is expensive, Povey and Woodland [2] defined a simple function to approximate accuracy. Let \( t \) and \( t' \) denote the nodes on the lattice, \( L \), and \( e_{t,t} \) be an edge from \( t' \) to \( t \) \((t' < t)\). The accuracy of the edge, \( \text{acc}(e_{t,t}) \), is obtained by

\[
\text{acc}(e_{t,t}) = \begin{cases} 
-1 + 2l & \text{if same word} \\
-1 + l & \text{if different word}, \tag{7}
\end{cases}
\]

where \( l \) is the ratio of the overlapping frame length between the word hypothesis of \( e_{t,t} \) and the corresponding reference word.

The expected raw accuracy of the lattice is obtained by visiting the nodes in the lattice topologically. For example, the forward raw accuracy, \( \text{acc}(t) \), at the node, \( t \), is given by

\[
\text{acc}(t) = \sum_{t'} \text{acc}(t') \times s(e_{t,t'}) \times \alpha(t') \times \alpha(t'), \tag{8}
\]

where \( \alpha(t) \) is the forward probability at \( t \) and the score, \( s(e_{t,t'}) \), at \( e_{t,t'} \) is given by

\[
s(e_{t,t'}) = \exp(\lambda_0 f_0(e_{t,t'}) + \lambda_1 f_1(e_{t,t'}) + \sum_{i=2}^{l} f_i(e_{t,t'})). \tag{9}
\]

\( \text{Loss}_2 \) is minimized by the L-BFGS algorithm as \( \text{Loss}_1 \), and in this paper the first derivatives, or the gradients of the loss function, are numerically computed by finite difference approximation. The loss function is generally used in discriminative acoustic modeling as an MWE criterion [2, 9], and we propose that the MWE criterion be applied to lattice rescoring.

### 2.3. Feature Functions

We used a set of feature functions that were based on n-grams. The word n-gram features were simply defined, and a trigram feature function for a word tuple \((u_1, u_2, u_3)\), for example, returns the number of the tuple that occurs in the hypothesis, \( w \), or zero when it is not observed. We defined a set of word bigram feature functions and that of word trigram feature functions.

The class n-gram feature functions were derived from the results of word clustering, which was carried out on the training corpora of language modeling similar to the algorithm described by Ney et al. [10]. In our experiments, the total number of classes was set to 1,000, and all entries in the vocabulary were mapped to a corresponding class to obtain class bigram and class trigram features.

Semantic class n-gram feature functions were also used as supplementary information, because we assumed that the similarities between words obtained from a thesaurus would also effectively avoid the data sparseness problem as well as the class n-gram feature functions. In our experiments, 27-k entries from a 60-k lexicon were mapped to 838 unique semantic classes according to a Japanese thesaurus [11], and we similarly defined bigram and trigram feature functions.

### 2.4. Related Work

We compared our proposed methods with the perceptron-based training algorithm [3, 12], which is a work related to discriminative rescoring. According to the averaged perceptron algorithm [12], the weighting factors are sequentially updated utterance by utterance as:

\[
\lambda^{m+1,t} = \lambda^{m,t} + \eta \{ f_i(w_{m,0}) - f_i(w_{m,\text{best}}) \}, \tag{10}
\]

where \( \eta \) is the learning rate, which is experimentally determined, and \( w_{m,\text{best}} \) is the sentence hypothesis with the highest score given by Eq. (3) for utterance \( m \) at the \( t \)-th round of the iterations. After the iterations, the weighting factors are averaged by

\[
\hat{\lambda}^{avg} = \frac{\sum_{t=1}^{T} \sum_{m=1}^{M} \lambda^{m,t}}{M \times T}, \tag{11}
\]

where \( M \) is the number of training utterances, and \( T \) is the total number of iterations.
3. Experiments

3.1. Experimental Setup

The LVCSR system decodes input acoustic feature vectors to a bigram lattice using a tree lexicon and a bigram language model in the first pass. The bigram lattice is expanded to a trigram lattice using the conventional lattice-expansion algorithm. In the second pass, it rescores the hypotheses in the lattices using a trigram language model and discriminative scores. We obtained the acoustic model from 118 hours of male speech obtained from broadcast news. The acoustic inputs were parameterized into 39 dimensional vectors: 12 mel frequency cepstral coefficients (MFCCs) with log-power and their first- and second-order differentials, using RASTA processing [13]. We trained the trigram language model using 1.5-M sentences (42.9-M words) from Japanese broadcast news manuscripts and transcriptions, and adapted it to the latest news programs [14].

The language model was smoothed by Good-Turing estimation, and the cut-offs were set to one for the bigrams and four for the trigrams. The lexicon for the model had 60-k words. Table 1 lists the training data for obtaining the weighting factors and the evaluation data for the experiments. The numbers of feature functions used in discriminative rescoring are listed in Table 2.

<table>
<thead>
<tr>
<th>Number of Feature Functions</th>
<th>Number of Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word bigram</td>
<td>$0.63 \times 10^6$</td>
</tr>
<tr>
<td>Word trigram</td>
<td>$4.02 \times 10^6$</td>
</tr>
<tr>
<td>Class bigram</td>
<td>$0.52 \times 10^6$</td>
</tr>
<tr>
<td>Class trigram</td>
<td>$3.60 \times 10^6$</td>
</tr>
<tr>
<td>Semantic class bigram</td>
<td>$0.47 \times 10^6$</td>
</tr>
<tr>
<td>Semantic class trigram</td>
<td>$0.95 \times 10^6$</td>
</tr>
<tr>
<td>Total</td>
<td>$10.20 \times 10^6$</td>
</tr>
</tbody>
</table>

3.2. Experimental Results

We compared the following rescoring methods on the trigram lattices of the evaluation data.

**Baseline**: Conventional trigram rescoring on lattices.

**Perceptron**: Rescoring with weighting factors obtained by the perceptron algorithm.

**Loss1**: Rescoring with weighting factors obtained by training based on the logistic regression.

**Loss2**: Rescoring with weighting factors obtained by training based on the minimization of word errors.

Table 3 lists the minimum WERs during iterations on all rescoring methods, because the speed of convergence differs from one training scheme to another. The rescoring results for **Loss2** using all the feature functions achieved the lowest WER (10.38%) and a 6.06% reduction in the WER relative to **Baseline**. According to a matched pair test [15], **Loss1** and **Loss2** decreased the WER at a significance level 0.05, while the results from **Perceptron** did not significantly decrease the WER compared with **Baseline**. Although **Loss1** and **Loss2** with word n-gram features accompanied by the class n-gram and the semantic class n-gram features provided gains on the rescoring results, the supplementary features did not significantly reduce the WERs compared with the loss functions exclusively using the word n-gram feature functions. However, **Loss1** and **Loss2** using only the class n-gram features provided slightly different results from those obtained by the word n-gram features. We found that **Loss1** and **Loss2** reduced substitution and insertion errors, whereas they did not restore the results from deletion errors compared with **Baseline**.

### Table 1: Experimental Data

<table>
<thead>
<tr>
<th>#sentences</th>
<th>#words</th>
<th>PP</th>
<th>%OOV</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training</td>
<td>53.1k</td>
<td>1.09M</td>
<td>49.0</td>
</tr>
<tr>
<td>Evaluation</td>
<td>2.073</td>
<td>31,425</td>
<td>21.8</td>
</tr>
</tbody>
</table>

### Table 2: Number of Feature Functions

<table>
<thead>
<tr>
<th>Feature Function</th>
<th>Number of Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word bigram</td>
<td>$0.63 \times 10^6$</td>
</tr>
<tr>
<td>Word trigram</td>
<td>$4.02 \times 10^6$</td>
</tr>
<tr>
<td>Class bigram</td>
<td>$0.52 \times 10^6$</td>
</tr>
<tr>
<td>Class trigram</td>
<td>$3.60 \times 10^6$</td>
</tr>
<tr>
<td>Semantic class bigram</td>
<td>$0.47 \times 10^6$</td>
</tr>
<tr>
<td>Semantic class trigram</td>
<td>$0.95 \times 10^6$</td>
</tr>
<tr>
<td>Total</td>
<td>$10.20 \times 10^6$</td>
</tr>
</tbody>
</table>

### Table 3: Rescoring Results (%)

<table>
<thead>
<tr>
<th>Method</th>
<th>Word WER</th>
<th>Sub WER</th>
<th>Del WER</th>
<th>Ins WER</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Baseline</strong></td>
<td>11.05</td>
<td>6.22</td>
<td>2.26</td>
<td>2.56</td>
</tr>
<tr>
<td><strong>Perceptron</strong></td>
<td>10.92</td>
<td>6.19</td>
<td>2.26</td>
<td>2.43</td>
</tr>
<tr>
<td>+ Class</td>
<td>11.08</td>
<td>6.27</td>
<td>2.23</td>
<td>2.58</td>
</tr>
<tr>
<td>+ Semantic</td>
<td>11.01</td>
<td>6.21</td>
<td>2.28</td>
<td>2.53</td>
</tr>
<tr>
<td>+ Class, Semantic</td>
<td>10.98</td>
<td>6.19</td>
<td>2.26</td>
<td>2.53</td>
</tr>
<tr>
<td><strong>Loss1</strong></td>
<td>11.08</td>
<td>6.27</td>
<td>2.23</td>
<td>2.58</td>
</tr>
<tr>
<td>+ Class</td>
<td>11.02</td>
<td>6.21</td>
<td>2.28</td>
<td>2.53</td>
</tr>
<tr>
<td><strong>Loss2</strong></td>
<td>10.77</td>
<td>6.06</td>
<td>2.29</td>
<td>2.42</td>
</tr>
<tr>
<td>+ Class</td>
<td>10.73</td>
<td>6.05</td>
<td>2.31</td>
<td>2.36</td>
</tr>
<tr>
<td>+ Semantic</td>
<td>10.73</td>
<td>6.04</td>
<td>2.30</td>
<td>2.40</td>
</tr>
<tr>
<td>+ Class, Semantic</td>
<td>10.70</td>
<td>6.06</td>
<td>2.28</td>
<td>2.36</td>
</tr>
<tr>
<td>Class</td>
<td>10.75</td>
<td>6.06</td>
<td>2.30</td>
<td>2.40</td>
</tr>
<tr>
<td>Semantic</td>
<td>10.90</td>
<td>6.21</td>
<td>2.23</td>
<td>2.46</td>
</tr>
</tbody>
</table>

![Figure 1: Comparison of WERs and Iterations](image-url)
ter rescoring results than the other methods at small iteration counts, it only provided slight improvements during iterations. The methods based on loss functions, i.e., Loss_1 and Loss_2, gradually reduced the WERs compared with the perceptron algorithm, and Loss_2 yielded the smallest WERs at the 36th round of iterations for all methods. Consequently, it is clear that Loss_2 was effective for discriminative rescoring.

Supplementary feature functions such as class n-gram and semantic class features did not significantly gain on the rescoring results compared with results only using word n-gram features when these were incorporated in Loss_1 and Loss_2. It would appear that these feature functions had the potential to provide smoothing effects to avoid the data sparseness problem as class-based language models. In the Perceptron case, the results were degraded when additional class n-gram features and semantic class n-gram features were employed. This is because the convergence of weighting factors derived from the perceptron algorithm depends on the learning rate, and the appropriate learning rate would be changed according to the combinations of feature functions.

4.2. Reduction in Number of Features

Finally, we investigated how many feature functions significantly affected the rescoring results. The feature functions would have a strong effect on discriminative rescoring if their corresponding word tuples were frequently observed as errors in the training data; therefore, the weighting factors were expected to have large magnitudes or absolute values.

We then carried out experiments using a reduced set of feature functions according to a criterion where some of the feature functions were removed from the set in each iteration step when the magnitudes of the factors were less than a threshold, which was given in advance. Table 4 lists the rescoring results for Loss_1 and Loss_2 with the reduced sets of feature functions. In all cases, all the feature functions were involved in the loss functions in the first round of iterations. The rescoring results from Loss_1 did not degrade the WER when the threshold was set to $1.0 \times 10^{-4}$, although only 1.62% of the feature functions remained, and the number of remaining feature functions was 165.3 k. Loss_2 also retained the WER when the threshold was set to $1.0 \times 10^{-2}$, and the number of remaining features was 534.0 k. In both training methods, a small fraction of the set of features dominated the reduction in WERs. Mainly, most of the remaining word n-gram features with large magnitudes for the weighting factors were tuples that consisted of functional words, such as auxiliary verbs and particles, and some of the tuples were parts of colloquial expressions as used in spontaneous speech. However, features containing topic-sensitive content words such as proper nouns were removed from the sets. Since tuples that contained functional words were frequently observed either as errors or correct words in the training lattices, they probably had a strong influence on minimizing the loss functions. Consequently, the focal point of future work will be how to recover content words from errors.

5. Conclusions

We proposed a method of discriminative rescoring for Japanese broadcast news transcriptions. The training scheme for scoring employed a criterion based on either the logistic regression or the minimization of word errors on the lattices. Discriminative rescoring based on the minimization of word errors produced a relative reduction of 6.06% when compared to conventional trigram lattice rescoring. The advantage of using discriminative rescoring was that it reflected information about erroneous words, and the new method significantly reduced the WERs especially when a loss function was employed, which explicitly minimized the expected WERs of the lattices. It is important to correct the errors in content words for broadcast news transcriptions, hence we intend to take topic-related information into consideration in future studies.

6. References