MINIMUM WORD ERROR TRAINING OF LONG SHORT-TERM MEMORY RECURRENT NEURAL NETWORK LANGUAGE MODELS FOR SPEECH RECOGNITION

Takaaki Hori, Chiori Hori, Shinji Watanabe, John R. Hershey

Mitsubishi Electric Research Laboratories (MERL)
201 Broadway, Cambridge, MA 02139, USA
{thori, chori, watanabe, hershey}@merl.com

ABSTRACT

This paper describes minimum word error (MWE) training of recurrent neural network language models (RNNLMs) for speech recognition. RNNLMs are usually trained to minimize a cross entropy of estimated word probabilities against the correct word sequence, which corresponds to maximum likelihood criterion. However, this training does not necessarily maximize a performance measure in a target task, i.e. it does not minimize word error rate (WER) explicitly in speech recognition. To solve such a problem, several discriminative training methods have already been proposed for n-gram language models, but those for RNNLMs have not sufficiently investigated. In this paper, we propose a MWE training method for RNNLMs, and report significant WER reductions when we applied the MWE method to a standard Elman-type RNNLM and a more advanced model, a Long Short-Term Memory (LSTM) RNNLM. We also present efficient MWE training with N-best lists on Graphics Processing Units (GPUs).

***Index Terms*—** Speech recognition, Recurrent neural network language model, Long short-term memory, Minimum word error training

1. INTRODUCTION

Language models are indispensable for large-vocabulary continuous-speech recognition. These models, which are usually built based on n-gram statistics, provide prior probabilities of hypothesized sentences to disambiguate their acoustical similarities. To build an n-gram model, text corpora are used to estimate the probability of a word's occurrence conditional on the preceding n-1 words, where n is typically 3 or 4.

On the other hand, continuous space language models based on neural networks have attracted increased attention in recent years [1–7]. With this approach, word indexes are mapped to a continuous space and word probability distributions are estimated as smooth functions in that space. Consequently, the approach makes it possible to provide better generalization for unseen n-grams [2]. A recurrent neural network language model (RNNLM) is a promising instance of such continuous space language models [3–7] An RNNLM has a hidden layer with re-entrant connections to itself with one word delay. Hence, the activations of the hidden units play a role of *memory* keeping a history from the beginning of the speech. Accordingly, the RNNLM can robustly estimate word probability distributions by taking long-distance inter-word dependencies into account. In addition, more advanced RNNLMs, Long Short-Term Memory (LSTM) RNNs [8] are introduced in language modeling for speech recognition [9], which can capture longer contextual information than the standard RNNLMs by handling the memory with several gating functions, and improves the recognition accuracy.

In most cases, RNNLMs are trained to minimize a cross entropy of estimated word probabilities against the correct word sequence given history, which corresponds to maximizing the likelihood for given training data. However, this training does not necessarily maximize a performance measure in a target task, i.e. it does not minimize word error rate (WER) explicitly in speech recognition. For n-gram-based language models, several types of discriminative language models (DLMs) and their training methods have been proposed [10–13] to solve this problem, but those for continuous space LMs have not sufficiently investigated except a few methods [14–16]. In [14], continuous feature representations were used for DLMs. In [15], a hidden activation vector of RNNLM is added to the feature vector for a log-linear LM. In [16], the cross entropy criterion is modified based on word confidence measure.

In this paper, we incorporate a minimum word error (MWE) criterion in a back-propagation through time (BPTT) algorithm for RNNLMs, which minimizes the expected word error rate using a set of N-best lists generated by a speech recognizer. Although this new method much increases the training computation in proportion to the size of N-best list, it can be performed in realistic time by parallelizing the BPTT over multiple word sequences using graphic processors. We evaluate our method with class-based Elman-type RNNLMs and LSTM RNNLMs with a meeting transcription task of AMI corpus [17] and a lecture transcription task of the Corpus of Spontaneous Japanese (CSJ) [18].

This paper is organized as follows. Section 2 describes prior work related to this paper. Section 3 explains basic structure of an RNNLM and a LSTM. Section 4 presents minimum word error training for RNNLMs and implementation techniques. Section 5 shows our experimental results on meeting and lecture transcription tasks, and Section 6 concludes this paper.

2. RELATED WORK

Discriminative training methods are widely used in speech recognition, where acoustic or language models are trained to optimize their parameters based on a discriminative criterion [19, 20]. Unlike the maximum likelihood approach, those methods can improve discriminative performance of models by taking a set of competing hypotheses for each training sample into account.

In language modeling, n-gram probabilities are directly optimized with a minimum classification error criterion [10], and log-linear language models with n-gram features are trained with a perceptron algorithm [12], reranking boosting [21], and minimum word error rate training [22, 23]. Since these methods were basically de-

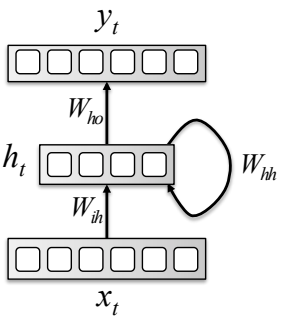

Fig. 1. RNNLM

signed for n-gram models or n-gram-feature-based models, they cannot be used directly for neural network-based language models including RNNLMs. Although a method in [15] employs a hidden activation vector of an RNNLM as additional features for a log-linear language model, the RNNLM itself is not trained discriminatively.

On the other hand, a discriminative training method has recently been proposed for RNNLMs [16]. In this method, the likelihood ratio of each reference word to the corresponding hypothesized word is used instead of the cross entropy. However, this method does not sufficiently exploit the potential ability of discriminative training with regards the following three reasons; (1) It considers only one competitor for each reference word, where the competitor is a hypothesized word in the 1-best ASR result. In general, it is better to consider multiple competitors in discriminative training. (2) It is not a sequence training since word-to-word alignment is fixed during the training. This means that inter-dependence of word errors is ignored. (3) It does not directly minimize word error rate that is the ASR performance measure. Our proposed method can handle all these problems by using expected word error rate as the objective function, which is calculated over N-best hypotheses generated by a speech recognizer.

Furthermore, we apply the proposed method to LSTM language models in addition to standard Elman-type RNNLMs. To the best of our knowledge, discriminative training of LSTM RNNLMs has not been reported.

3. RECURRENT NEURAL NETWORK LANGUAGE MODELS

In this work, we discriminatively train a class-based RNNLM [4]. and an LSTM RNNLMs based on a minimum word error criterion.

For simplicity, we start from a standard RNNLM depicted in Fig. 1. Given a word sequence $w_1, \ldots, w_t, \ldots, w_T$ with vocabulary $\mathcal{V}$, the input vector $x_t \in \{0,1\}^{|\mathcal{V}|}$ for time index t is represented as

$$x_t = \text{OneHot}(w_{t-1}), \tag{1}$$

where $\text{OneHot}(w)$ denotes the 1-of-N coding of word w, which converts a word index to a one hot vector representation.

The D dimensional activation vector $h_t \in [0,1]^D$ in the current hidden layer can be computed as

$$h_t = \sigma(W_{ih} x_t + W_{hh} h_{t-1}), \tag{2}$$

where $W_{ih} \in \mathbb{R}^{D \times |\mathcal{V}|}$ and $W_{hh} \in \mathbb{R}^{D \times D}$ are the input-hidden and hidden-hidden weight matrices. $\sigma(\cdot)$ is an element-wise sigmoid function.

Output vector $y_t \in [0,1]^{|\mathcal{V}|}$, which corresponds to the predicted word distribution, is obtained as

$$y_t = \zeta(W_{ho} h_t), \tag{3}$$

where W_{ho} is the weight matrix to the output layer. $\zeta(\cdot)$ denotes a softmax function that computes the softmax over the elements in a given vector. Finally, the following is the word occurrence probability of w_t in context h_t,

$$P(w_t|h_t) \equiv y_t[w_t], \tag{4}$$

where $y_t[w_t]$ indicates the w_t-th element of vector y_t. Hereafter, we use $[\cdot]$ to specify an element in the vector.

For class-based RNNLMs, the output vector $y_t \in [0,1]^{|\mathcal{V}|+|\mathcal{C}|}$ ($|\mathcal{C}|$: number of classes) consists of word and class outputs

$$y_t = \begin{bmatrix} y_t^{(w)} \\ y_t^{(c)} \end{bmatrix}, \tag{5}$$

which can be obtained as

$$y_{t,m}^{(w)} = \zeta(W_{ho,m}^{(w)} h_t) \tag{6}$$

$$y_t^{(c)} = \zeta(W_{ho}^{(c)} h_t), \tag{7}$$

where $y_{t,m}^{(w)}$ and $W_{ho,m}^{(w)}$ are the sub-vector of $y_t^{(w)}$ and sub-matrix of W_{ho} corresponding to the words in the m-th class, respectively. $W_{ho}^{(c)}$ is the sub-matrix of W_{ho} for the class output.

The word occurrence probability is computed as

$$P(w_t|h_t) \equiv y_{t,C(w_t)}^{(w)}[w_t] \times y_t^{(c)}[C(w_t)] \tag{8}$$

where $C(w)$ denotes the index of the class the word w belongs to.

With the class-based architecture, the computation for propagating activations from the hidden layer to the output layer can be reduced, since we need to handle only the words in the class of the current word to compute the softmax function rather than all the words in the vocabulary.

As an extension of RNNs, Long Short-Term Memory (LSTM) RNNs were proposed [8] and applied to language modeling [9]. It is well known that the standard RNNs cannot hold the hidden activation information for long time because the activation pattern at a certain time is exponentially decaying according as iterative propagation through time, and it is difficult to train interdependence between distant events [24]. To solve this problem, the LSTM has memory cells instead of regular network units. An LSTM cell can remember a value for an arbitrary length of time, which contains input, forget, and output gates that determine when the input is significant enough to remember, when it should continue to remember or forget the value, and when it should output the value. An example of LSTM cell is depicted in Fig. 2.

4. MINIMUM WORD ERROR TRAINING FOR RNNLMS

The loss function of minimum word error training can be written as

$$\mathcal{L}(\Lambda) = \sum_{k=1}^{K} \sum_{W \in \mathcal{V}^*} \mathcal{E}(W_k^{(R)}, W) P_\Lambda(W|O_k), \tag{9}$$

where Λ is the set of model parameters, K is the number of utterances in training data, O_k is the k-th acoustic observation sequence,

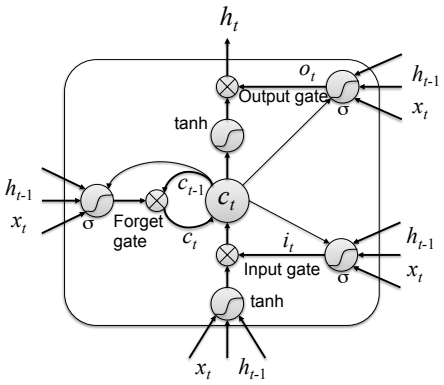

Fig. 2. Long short-term memory cell

and $W_k^{(R)} = \{w_{k,1}^{(R)}, \cdots, w_{k,T_k}^{(R)}\}$ is the k-th reference word sequence. $\mathcal{E}(W', W)$ represents the edit distance between two word sequences W' and W. $P_\Lambda(W|O)$ is the posterior probability of W computed with the model parameter set Λ.

In this work, since we use a set of N-best lists, we obtain the loss function as

$$\mathcal{L}(\Lambda) = \sum_{k=1}^{K} \sum_{n=1}^{N} \mathcal{E}(W_k^{(R)}, W_{k,n}) P_\Lambda(W_{k,n}|O_k), \tag{10}$$

where $W_{k,n} = \{w_{k,n,1}, \cdots, w_{k,n,T_{k,n}}\}$ is a word sequence of n-th hypothesis in the N-best list for k-th utterance. $T_{k,n}$ denotes the number of words in hypothesis $W_{k,n}$. The posterior probability of $W_{k,n}$ can be computed as

$$P_\Lambda(W_{k,n}|O_k) = \frac{\exp(g_{k,n})}{\sum_{m=1}^{N} \exp(g_{k,m})} \tag{11}$$

where $g_{k,n}$ is the log-likelihood score of hypothesis $W_{k,n}$ obtained by

$$g_{k,n} = \alpha \log P_{\Lambda_L}(W_{k,n}) + \log P_{\Lambda_A}(O_k|W_{k,n}), \tag{12}$$

and Λ_L and Λ_A are the sets of language and acoustic model parameters, respectively. We assume that Λ_A is fixed in language model training. α is a scaling factor to balance the acoustic and language scores.

The language log-probability is obtained by the RNNLM as

$$\begin{aligned} \log P_{\Lambda_L}(W_{k,n}) &= \sum_{t=1}^{T_{k,n}} \log P_{\Lambda_L}(w_{k,n,t}|h_{k,n,t}) \\ &= \begin{cases} \sum_{t=1}^{T_{k,n}} \log y_{k,n,t}[w_{k,n,t}], \\ \sum_{t=1}^{T_{k,n}} \log y^{(w)}_{k,n,t,C(w_{k,n,t})}[w_{k,n,t}] \times y^{(c)}_{k,n,t}[C(w_{k,n,t})] \end{cases} \end{aligned} \tag{13}$$

where $y_{k,n,t}[w_{k,n,t}]$ corresponds to the output of the RNNLM for the t-th word in $W_{k,n}$. Each word probability can be computed using a word-based model (the upper in the curly brace) or class-based model (the lower in the curly brace) according to Eq. (8). Hereafter, we describe the optimization procedure only for the word-based models, but it can be easily extended for the class-based models.

We obtain partial derivatives of loss function $L(\Lambda)$ with respect to Λ_L for the back propagation through time (BPTT) algorithm. For simplicity, here we only show the derivative with respect to each RNNLM's output $o_{k,n,t}$ before applying the softmax function, i.e. $\partial\mathcal{L}(\Lambda)/\partial o_{k,n,t}[i]$, where

$$y_{k,n,t}[w_{k,n,t}] = \frac{\exp(o_{k,n,t}[w_{k,n,t}])}{\sum_{i=1}^{|\mathcal{V}|} \exp(o_{k,n,t}[i])}. \tag{14}$$

The derivative can be factorized into two derivatives using the chain rule as

$$\frac{\partial\mathcal{L}(\Lambda)}{\partial o_{k,n,t}[i]} = \frac{\partial\mathcal{L}(\Lambda)}{\partial g_{k,n}} \frac{\partial g_{k,n}}{\partial o_{k,n,t}[i]}. \tag{15}$$

The first factor corresponds to the differences with respect to the N-best hypothesis scores, and the second factor corresponds to those of RNN's output from the target. Accordingly, if we obtained the first factor for each N-best hypothesis, the original BPTT algorithm can be performed over N-best hypotheses using the multiplication of these two factors as the error signal for the RNNLM.

By substituting Eqs. (11) and (12) into Eq. (10), the first factor can be obtained as

$$\begin{aligned} &\frac{\partial\mathcal{L}(\Lambda)}{\partial g_{k,n}} \\ &= \sum_{n'=1}^{N} \mathcal{E}(W_k^{(R)}, W_{k,n'}) P_\Lambda(W_{k,n}|O_k)\{\delta_{n,n'} - P_\Lambda(W_{k,n'}|O_k)\}, \\ &= P_\Lambda(W_{k,n}|O_k) \left\{ \mathcal{E}(W_k^{(R)}, W_{k,n}) - \bar{\mathcal{E}}^{(k)} \right\} \end{aligned} \tag{16}$$

where $\bar{\mathcal{E}}^{(k)}$ stands for the expectation of the number of word errors, which corresponds to

$$\bar{\mathcal{E}}^{(k)} = \sum_{n'=1}^{N} \mathcal{E}(W_k^{(R)}, W_{k,n'}) P_\Lambda(W_{k,n'}|O_k). \tag{17}$$

The second factor is obtained with the same way as the case of cross entropy criterion by using Eqs. (13) and (14).

$$\frac{\partial g_{k,n}}{\partial o_{k,n,t}[i]} = \frac{\partial y_{k,n,t}[w_{k,n,t}]}{\partial o_{k,n,t}[i]} = (\delta_{i,w_{k,n,t}} - y_{k,n,t}[i]). \tag{18}$$

As shown in the above equations, the first factor has an effect that if the number of errors is larger than its mean value, the error signal of Eq. (18), i.e. the second factor, is emphasized toward the positive direction, and if the number of errors is smaller, the error signal is emphasized toward the negative direction.

In the training iterations, we apply the stochastic gradient descent method on utterance-by-utterance basis, i.e. the gradients are accumulated over N-best hypotheses in the list. For each hypothesis, BPTT is performed with the error vector obtained by Eqs. (15)-(18). After processing each N-best list, the parameters are updated with the sum of gradients.

However, the proposed method needs more computations than cross-entropy-based training since the number of sentences increases by N times for N-best lists. We solve this problem by parallelization with Graphics Processing Units (GPUs). Actually, gradient accumulation can be performed in parallel over multiple hypotheses in each N-best list. According to the technique in [2], we input multiple words at the same time to the RNNLM, where all the words are located at the same position in the different sentences of the N-best list. Since the set of words, hidden activations, and output probabilities can be represented as a matrix, most steps in training can be performed by matrix-matrix operations on a GPU.

Table 1. Data sets used for experiments

		Train	Dev.	Test
AMI	Speech [hours]	77.9	8.9	8.7
	Transcript [#word]	108,502	94,914	89,635
CSJ	Speech [hours]	236.5	1.9	2.0
	Transcript [#word]	3,764,025	28,790	28,253

5. EXPERIMENTS

We evaluated our discriminative training approach with a meeting transcription task of AMI corpus [17] and a lecture transcription task of the Corpus of Spontaneous Japanese (CSJ) [18]. The data sizes of training, development and test sets are summarized in Table 1.

The baseline ASR systems for AMI and CSJ were prepared according to Kaldi's recipes [25]. Mel-Frequency Cepstral Coefficient (MFCC) features were extracted from 16-kHz-sampled speech data, which were then transformed using feature-space maximum likelihood linear regression (fMLLR) [26] using the speaker labels. Each fMLLR transform was estimated using a Gaussian Mixture Model (GMM) based ASR system by iteratively maximizing the likelihood of the data given the transcription alignments for the training data, and the one-best hypothesis alignment obtained by the system for the test data.

Standard (11 frame context) Deep Neural Network (DNN) acoustic models with six layers were trained for each task, where the AMI model had 2048 units in each hidden layer and 3987 units in the output layer, and the CSJ model had 1905 units in each hidden layer and 9380 units in the output layer [27]. The both acoustic models were trained based on Cross Entropy (CE) and retrained with a state-level Minimum Bayes Risk (sMBR) criterion [28].

Kneser-Ney smoothed 3-gram language models were prepared for the baseline systems. For AMI task, the 3-gram model was made by linear interpolation of two models trained with transcripts of AMI and Fisher corpora, respectively. The Fisher corpus [29] consists of telephone conversations, which is approximately 13 times larger than AMI with regard to the transcripts. For CSJ task, the baseline 3-gram model was made only with CSJ transcripts. The vocabulary sizes for AMI and CSJ were 49,413 and 75,568, respectively.

Standard RNN and LSTM language models were first trained with the cross entropy, and then retrained using the proposed MWE training method. The vocabulary sizes of AMI and CSJ models were 7,479 and 24,970, respectively, which were reduced from their baseline vocabularies by substituting less frequent words with a specific symbol. This vocabulary reduction is effective in terms of computation, and did not have any negative impact on the recognition accuracy in our preliminary experiments. Furthermore, we did not use the Fisher corpus for training AMI RNNLMs, which resulted in almost the same accuracy as that when including the corpus. Each standard RNNLM was constructed as a class-based model, which had one hidden layer of 300 units, while each LSTM model was made as a word-based model, which had two hidden LSTM layers of 300 units.

In RNNLM training, the development set was used for validation to control the learning rate, i.e. the rate was reduced if cross entropy or word error rate for the development set increased after each iteration. Finally, the training process was stopped when the learning rate reached a certain threshold.

We generated a 100-best list for each utterance in training, development, and test sets using the baseline ASR systems, and used those lists for MWE training and its evaluation. We measured the performance of language models in word error rate (WER) [%] of 1-best hypotheses after reranking each 100-best list by RNNLM scores. In all experiments with RNNLMs, the 3-gram probabilities were used together with RNNLM probabilities by linearly interpolating them in log domain, since the interpolation always yields better performance than single use of RNNLMs. This interpolated language scores were also used in MWE training to calculate the hypothesis score of Eq. (12).

Table 2. Word error rate on AMI corpus

	Dev. set	Test set
Baseline 3-gram	24.4	24.7
RNNLM CE	23.1	23.3
RNNLM MWE	**22.7**	**22.8**
LSTM-LM CE	22.9	23.0
LSTM-LM MWE	**22.5**	**22.6**
N-best oracle	12.5	12.0

Table 3. Word error rate on CSJ corpus

	Dev. set	Test set
Baseline 3-gram	9.1	10.9
RNNLM CE	8.7	10.3
LSTM-LM CE	8.2	9.9
LSTM-LM MWE	**8.0**	**9.7**
N-best oracle	4.5	5.8

Tables 2 and 3 show word error rates when using different language models in AMI and CSJ tasks. In the tables, "N-best oracle" means the average WER of the hypothesis with the minimum word errors in each N-best list. This indicates the lower-bound of WER that can be reached by N-best rescoring.

As shown in the tables, RNNLMs always outperform the baseline 3-gram models. For standard and LSTM RNNLMs, we obtained certain WER reductions by MWE training, in which all the WER differences are statistically significant with the level of 5% . In addition, the LSTM LMs yielded lower WERs than those of standard RNNLMs. Although it is not a strict comparison between RNN and LSTM LMs because their structures are not the same, we can see that our MWE training method are effective for both types of RNNLMs.

6. CONCLUSION

In this paper, we proposed a minimum word error (MWE) training method for recurrent neural network language models, which explicitly minimizes word error rate (WER) using multiple hypotheses generated by a speech recognizer. From the experimental results on the AMI and CSJ corpora, we showed that the proposed method yielded standard RNN and LSTM language models with lower WERs than those trained with a conventional cross entropy criterion, where the WER reductions were statistically significant. Finally we achieved 22.6% WER in the AMI task and 9.7% WER in the CSJ task by using MWE-trained LSTM language models, which correspond to 8.5% and 11% error reductions from the 3-gram baseline, respectively. Future work will include comparison and combination with other discriminative approaches such as the method of [16].

7. REFERENCES

[1] Yoshua Bengio, Rejean Ducharme, and Pascal Vincent, "A neural probabilistic language model," *Journal of Machine Learning Research*, vol. 3, pp. 1137–1155, 2003.

[2] H. Schwenk, "Continuous space language models," *Computer Speech and Language*, vol. 21, no. 3, pp. 492–518, 2007.

[3] T. Mikolov, M. Karafiat, L. Burget, J. Cernocky, , and S. Khudanpur, "Recurrent neural network based language model," in *Proc. Interspeech*, 2010, pp. 1045–1048.

[4] T. Mikolov, S. Kombrink, L. Burget, J. Cernocky, and S. Khudanpur, "Extensions of recurrent neural network language model," in *Proc. ICASSP*, 2011, pp. 5528–5531.

[5] S. Kombrink, T. Mikolov, M. Karafiat, and L. Burget, "Recurrent neural network based language modeling in meeting recognition," in *Proc. Interspeech*, 2011, pp. 2877–2880.

[6] Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson, "One billion word benchmark for measuring progress in statistical language modeling," Tech. Rep., Google, 2013.

[7] Xie Chen, Yongqiang Wang, Xunying Liu, Mark JF Gales, and Philip C Woodland, "Efficient GPU-based training of recurrent neural network language models using spliced sentence bunch," *Interspeech*, 2014.

[8] Sepp Hochreiter and Jürgen Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[9] Martin Sundermeyer, Ralf Schlüter, and Hermann Ney, "LSTM neural networks for language modeling.," in *INTERSPEECH*, 2012.

[10] Hong-Kwang Jeff Kuo, Eric Fosler-Lussier, Hui Jiang, and Chin-Hui Lee, "Discriminative training of language models for speech recognition," in *Acoustics, Speech, and Signal Processing (ICASSP), 2002 IEEE International Conference on*. IEEE, 2002, vol. 1, pp. I–325.

[11] Zheng Chen, Kai-Fu Lee, and Ming-jing Li, "Discriminative training on language model," *Optimization*, vol. 1, pp. 1, 2000.

[12] Brian Roark, Murat Saraclar, Michael Collins, and Mark Johnson, "Discriminative language modeling with conditional random fields and the perceptron algorithm," in *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, 2004, p. 47.

[13] Takanobul Oba, Takaaki Hori, Atsushi Nakamura, and Akinori Ito, "Round-robin duel discriminative language models," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 20, no. 4, pp. 1244–1255, 2012.

[14] Puyang Xu, Sanjeev Khudanpur, Maider Lehr, E Prud'hommeaux, Nathan Glenn, Damianos Karakos, Brian Roark, Kenji Sagae, Murat Saraçlar, Izhak Shafran, et al., "Continuous space discriminative language modeling," in *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*. IEEE, 2012, pp. 2129–2132.

[15] A. Kobayashi, M. Ichiki, Y. Fujita, T. Oku, and S. Sato, "Discriminative language modeling based on recurrent neural networks," in *Proc. of 2014 Autumn Meeting of the Acoustical Society of Japan (in Japanese)*, 2014.

[16] Y. Tachioka and S. Watanabe, "A discriminative method for recurrent neural network language models," in *Acoustics, Speech, and Signal Processing (ICASSP), 2015 IEEE International Conference on*, 2015.

[17] Steve Renals, Thomas Hain, and Hervé Bourlard, "Recognition and understanding of meetings the ami and amida projects," in *Proc. of the IEEE Workshop on Automatic Speech Recognition and Understanding, ASRU'07*, 12 2007.

[18] Kikuo Maekawa, "Corpus of spontaneous japanese: Its design and evaluation," in *ISCA & IEEE Workshop on Spontaneous Speech Processing and Recognition*, 2003.

[19] Biing-Hwang Juang, Wu Hou, and Chin-Hui Lee, "Minimum classification error rate methods for speech recognition," *Speech and Audio Processing, IEEE Transactions on*, vol. 5, no. 3, pp. 257–265, 1997.

[20] Brian Roark, "A survey of discriminative language modeling approaches for large vocabulary continuous speech recognition," *Automatic Speech and Speaker Recognition: Large Margin and Kernel Methods*, pp. 117–137, 2009.

[21] Zhengyu Zhou, Jianfeng Gao, Frank K Soong, and Helen Meng, "A comparative study of discriminative methods for reranking lvcsr n-best hypotheses in domain adaptation and generalization," in *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*. IEEE, 2006, vol. 1, pp. I–I.

[22] Franz Josef Och, "Minimum error rate training in statistical machine translation," in *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*. Association for Computational Linguistics, 2003, pp. 160–167.

[23] Jen-Wei Kuo and Berlin Chen, "Minimum word error based discriminative training of language models.," in *INTERSPEECH*, 2005, pp. 1277–1280.

[24] Yoshua Bengio, Patrice Simard, and Paolo Frasconi, "Learning long-term dependencies with gradient descent is difficult," *Neural Networks, IEEE Transactions on*, vol. 5, no. 2, pp. 157–166, 1994.

[25] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukáš Burget, Ondřej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlíček, Yanmin Qian, Petr Schwarz, et al., "The kaldi speech recognition toolkit," in *Proc. of the IEEE Workshop on Automatic Speech Recognition and Understanding, ASRU'11*, 2011.

[26] Mark JF Gales, "Maximum likelihood linear transformations for hmm-based speech recognition," *Computer speech & language*, vol. 12, no. 2, pp. 75–98, 1998.

[27] Takafumi Moriya, Tomohiro Tanaka, Takahiro Shinozaki, Shinji Watanabe, and Kevin Duh, "Automation of system building for state-of-the-art large vocabulary speech recognition using evolution strategy," in *Proc. of the IEEE Workshop on Automatic Speech Recognition and Understanding, ASRU'15*, 2015.

[28] Karel Veselỳ, Arnab Ghoshal, Lukás Burget, and Daniel Povey, "Sequence-discriminative training of deep neural networks.," in *INTERSPEECH*, 2013, pp. 2345–2349.

[29] Christopher Cieri, David Miller, and Kevin Walker, "The fisher corpus: a resource for the next generations of speech-to-text.," in *Fourth International Conference on Language Resources and Evaluation*, 2004.