THE EFFECT OF LATTICE PRUNING ON MMIE TRAINING

Long Qin and Alexander Rudnicky

School of Computer Science, Carnegie Mellon University
5000 Forbes Avenue, Pittsburgh, PA 15213, USA

ABSTRACT

In discriminative training, such as Maximum Mutual Information Estimation (MMIE) training, a word lattice is usually used as a compact representation of many different sentence hypotheses and hence provides an efficient representation of the confusion data. However, in a large vocabulary continuous speech recognition (LVCSR) system trained from hundreds or thousands hours training data, the extended Baum-Welch (EBW) computation on the word lattice is still very expensive. In this paper, we investigated the effect of lattice pruning on MMIE training, where we tested the MMIE performance trained with different lattice complexity. A beam pruning and a posterior probability pruning method were applied to generate different sizes of word lattices. The experimental results show that using the posterior probability lattice pruning algorithm, we can save about 40% of the total computation and get the same or more improvement compared to the baseline MMIE result.

Index Terms— MMIE training, word lattice, lattice pruning

1. INTRODUCTION

Discriminative training schemes, such as Maximum Mutual Information Estimation (MMIE) training [1], have been developed to improve the accuracy of speech recognition systems trained using maximum likelihood estimation (MLE). To eliminate computation problems, the word lattice is used as a compact representation of competing hypotheses [2], and parameter optimization is conducted using the extended Baum-Welch (EBW) algorithm [3].

The word lattice is usually a by-product of the recognition process. It contains a set of word hypotheses with boundary times and transitions between different hypotheses [4]. If no pruning is done, the lattice can be highly accurate but also exorbitantly large. One of the most effective lattice pruning methods is the beam pruning during decoding [5]. Another very common method is to further compress the lattice after it has been generated, where a word lattice is usually converted to a word graph and all the time alignment information is discarded, such as the finite state automata determination and minimization [6] and the confusion network approach [7].

The above lattice pruning methods are not appropriate for the purpose of lattice pruning for discriminative training. The word lattice used in MMIE training is quite different from the lattice used for multi-pass decoding: 1) normally a unigram language model is used for calculating the language model score for word hypotheses [8][9], 2) the time alignment information needs to be kept for the EBW computation 3) the Gaussian occupation count is weighted by the posterior probability of the word hypothesis. In this paper, we proposed a new lattice pruning method which keeps the boundary times of word hypotheses in the lattice and prunes unnecessary word hypotheses by considering their posterior probabilities. The posterior probability of a word hypothesis is calculated in the same way as in MMIE training by doing the forward-backward computation on the lattice. Using this method, word hypotheses which do not contribute much to Gaussian count accumulation are removed from the lattice. Therefore, we can save much EBW computation on the lattice and still have the same performance as training with the original lattice. Actually in [10], we found MMIE training with the posterior probability lattice pruning can even yield more improvement than the baseline MMIE results. While it is possible to dispense with pruning [11], however, from the experiment results, doing so does not appear to result in better adaptation. Moreover, practical applications of discriminative training will still benefit from better managed computational costs.

The rest of the paper is organized as follows. In section 2, an introduction of MMIE training is given. Then Section 3 describes the beam pruning method. The details of the posterior probability lattice pruning algorithm will be presented in Section 4. And in Section 5 and 6 the experiment setup and results are provided. Concluding remarks are provided in Section 7.

2. MAXIMUM MUTUAL INFORMATION ESTIMATION

As an alternative of the MLE training of HMMs, MMIE training attempts to optimize the correctness of a model by formulating an objective function that penalizes the confusable model to the true model,

$$F(\lambda) = \sum_{r=1}^{R} \log \frac{P_\lambda(O_r|M_{s_r})P(S_r)}{\sum_s P_\lambda(O_r|M_s)P(S)}, \tag{1}$$

where λ represents the acoustic model parameters, O_r are the training utterances, M_s is the model corresponding to a word sequence s, and s_r is the correct transcription for the rth utterance, while $P(S)$ is a weakened language model such as a unigram language model. Thus, MMIE tries to maximize the likelihood of the correct transcription and simultaneously minimize the likelihood of the competing word hypotheses.

The MMIE objective function can be optimized using the EBW algorithm, and the mean and variance of a particular dimension of the Gaussian mixture component m for state j can be re-estimated as follows

$$\hat{\mu}_{j,m} = \frac{\{\theta^{num}_{j,m}(O) - \theta^{den}_{j,m}(O)\} + D_{j,m}\mu_{j,m}}{\{\gamma^{num}_{j,m} - \gamma^{den}_{j,m}\} + D_{j,m}} \tag{2}$$

$$\hat{\sigma}^2_{j,m} = \frac{\{\theta^{num}_{j,m}(O^2) - \theta^{den}_{j,m}(O^2)\} + D_{j,m}(\sigma^2_{j,m} + \mu^2_{j,m})}{\{\gamma^{num}_{j,m} - \gamma^{den}_{j,m}\} + D_{j,m}} - \hat{\mu}^2_{j,m}, \tag{3}$$

where $r_{j,m}$ is the occupation count of the Gaussian mixture component, and $\theta_{j,m}(O)$ and $\theta_{j,m}(O^2)$ are the standard weighted sums over feature $x(t)$ and $x(t)^2$ respectively. The Gaussian specific learning rate constant $D_{j,m}$ is used to control the speed of the

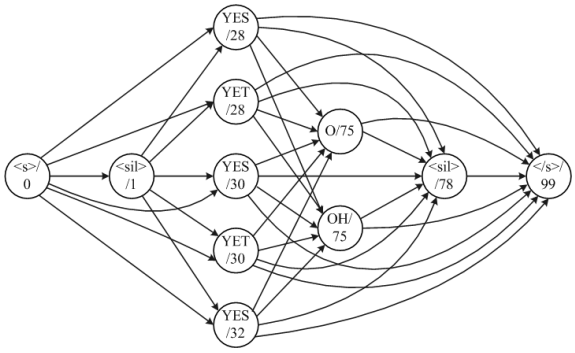

Fig. 1. A lattice corresponding to "<s> YES </s>"

parameter update and is set to be large enough to ensure that all Gaussian variances remain positive. The above update equations require accumulating statistics for the numerator and denominator HMMs by doing the forward-backward computation on the numerator and denominator lattices. The numerator lattice can be generated by aligning the acoustic data against a network of HMMs built according to the correct transcription. The denominator lattice corresponds to all possible competing word sequences and is a by-product of the recognition process of the training utterances.

Fig. 1 gives an example of a lattice generated from the Sphinx III decoder [12]. In this lattice, each node is associated with a word and its entry time. An arc is the transition from one node to another. We can find that even a sentence only has one word, there are many nodes and arcs in the lattice. For a longer sentence in a LVCSR system, a lattice may contain thousands of nodes and arcs. The EBW computation on such lattice can still be very time consuming. In our experiment, to simplify the EBW implementation, we converted the numerator lattices and denominator lattices into a simpler format in which each element contains a word hypothesis and the indexes for preceding and succeeding hypotheses. In this simplified format, word hypotheses are sorted by their entry time.

3. BEAM PRUNING

Beam pruning is one of the most effective ways to do lattice pruning. It is applied during the recognition process of the training data. During decoding, the decoder only expands the HMM states which have accumulated likelihoods that are greater than a beam width multiplied by the best accumulated likelihood so far. By doing this, we can eliminate those states which have very low likelihoods and get a very small lattice. But in MMIE training, where the Gaussian mixture occupation count is weighted by the posterior probabilities of word hypotheses, pruning only by accumulated likelihood is not appropriate. Also in beam pruning, when a state is pruned, all word sequences ending with that state are pruned out. By doing this, we may lose many competing word hypotheses which have high posterior probabilities. In fact, the beam pruning method has been reported to reduce the benefit of MMIE training [11]. Here we just consider it as a baseline result.

4. POSTERIOR PROBABILITY PRUNING

In MMIE training, the Gaussian occupation count from each competing word hypothesis is weighed by the word posterior probability. Word hypotheses with low posterior probabilities do not contribute much to the statistics accumulation, so pruning them out should not affect the MMIE performance. As shown in Fig. 1, nodes and arcs are two basic elements of a word lattice, and we can thus perform lattice pruning on those two levels respectively.

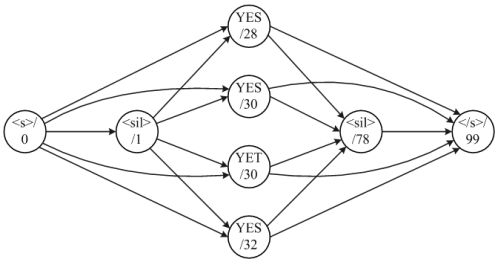

Fig. 2. A lattice corresponding to "<s> YES </s> after the posterior probability arc pruning"

4.1. Arc Pruning

Each node in the lattice has a number of outgoing arcs which are transitions from the current node to succeeding nodes. We can find an arc which has the highest posterior probability within all those transitions. Then arcs (transitions) whose posterior probability is lower than a beam width multiplied by the best one will be pruned. This can be done in a forward pass from the beginning to the end of the lattice. Similarly, each node also has many incoming arcs which are transitions from preceding nodes to the current node. We can therefore do a backward pass of pruning on those incoming arcs.

The posterior probability of each arc is calculated in the same way as doing the forward-backward computation on the lattice in MMIE training. Assuming in a word lattice, we have $q = 1...Q$ arcs sorted in time. And we have already calculated the pre-scaled language model score $lm(q)$ and acoustic likelihood $ac(q)$ for each arc. In this paper, the language model score $lm(q)$ comes from a unigram language model, while the acoustic likelihood $ac(q)$ is estimated using Viterbi. Then the posterior probability of each arc can be computed as:

1. Initialize $\alpha(1) = \beta(Q) = 1.0$
2. for arc $q = 1...Q$, for arc r preceding q,
   $\alpha(q) += \alpha(r) * ac(q) * lm(q)$
3. for arc $q = Q...1$, for arc r following q,
   $\beta(q) += \beta(r) * ac(q) * lm(q)$
4. for arc $q = 1...Q$,
   $\gamma(q) = \alpha(q) * \beta(q)/(ac(q) * lm(q) * \beta(1))$

Fig. 2 shows the posterior probability arc pruning result of the Fig. 1 lattice. It can be observed that the lattice now contains much fewer arcs. And some nodes in the lattice have also been removed as a result of the pruning of all its incoming or outgoing arcs.

4.2. Node Pruning

Aside from the large number of arcs in the word lattice, we can also find many duplicate nodes with the same word hypothesis but different entry times, such as node "YES/28", node "YES/30" and node "YES/32" as well as node "YET/28" and node "YET/30" in Fig. 1. In some lattices, the number of such duplicate nodes might be hundreds. So in addition to the posterior probability arc pruning, we can also apply the posterior probability pruning on the node level. First,

the posterior probability of each node is calculated as the sum of the posterior probability of all its outgoing arcs. Then from the beginning to the end, the search of the duplicate nodes is done in the range of a 20-frame window, which is actually +/-10 frames around a node. Among those duplicate nodes, the node whose posterior probability is lower than a beam width multiplied by the best one will be pruned.

Clearly, these two levels of posterior probability pruning can be combined together to get the best result. To do this, we fixed the beam width for the node pruning and then tried different beam widths for the arc pruning.

5. EXPERIMENT SETUP

To investigate the effect of above lattice pruning methods on MMIE training, we worked with the WSJ-SI84 data set, which contains about 15 hours speech from 84 different speakers. The input data was 13-order mel-scale frequency cepstral coefficients (MFCC) including c_0, and then the delta and delta-delta coefficients were calculated for training. The initial MLE-trained HMMs had 2932 states with 8 mixture components per state. The testing data was the Nov. '92 5k-word task development and evaluation set. And the recognition used the Lincoln Labs 5k-word closed vocabulary trigram language model from WSJ0. The numerator lattices were generated by doing forced alignment between the correct transcription and the corresponding HMM network. The denominator lattices were generated by decoding the training utterances using a 64k-word vocabulary unigram language model trained from the WSJ0 language model data. The same unigram language model was also used for lattice probability computation. The numerator lattice was always incorporated into the denominator lattice to ensure that the denominator lattice contain the correct transcription of the utterance. In this paper, the lattices were only generated once and then used in each iteration of the EBW computation.

All experiments were performed in a Linux cluster environment, where each machine has a 3.0 GHz 4-core Xeon CPU and 16 GB memory. And we used 20 CPUs of it to run our experiments.

6. EXPERIMENT RESULTS

6.1. Computation Analysis

The computation of MMIE training mainly comes from three parts: the lattice generation, the lattice format conversion and the EBW computation. In our experiments, the lattice format conversion usually costs much more than the other two parts. Furthermore, the computation from the latter two parts is proportional to the number of word hypotheses in the lattice. For example, if we pruned the lattice to half of its original size, then we can save about 50% of the running time for the lattice format conversion and the parameter update using EBW. Table 1 shows the comparison of running time for MMIE training on WSJ-SI84 data set. The size of the lattice generated from beam pruning or the posterior probability pruning is about half of the lattice used in the baseline MMIE training. Step 1, 2, 3 corresponds to the lattice generation, the lattice format conversion and 4-iteration EBW run, respectively. And step 4 is the posterior probability pruning. In this paper, the reported execution times are the CPU time averaged across all 20 processors and should be considered approximate. It can be found, we can save about 40% to 50% of the overall MMIE running time when performing MMIE training with beam pruning or the posterior probability pruning. Generally, the smaller the lattices, the less computation we would need.

Table 1. Comparison of running time (minutes) for MMIE training

	Step 1	Step 2	Step 3	Step 4	Total
baseline	28.3	141.9	78.0	-	258.2
beam	19.6	72.3	36.8	-	128.7
posterior	28.3	72.8	37.0	19.8	157.9

6.2. Baseline MMIE Results

Table 2 shows the recognition results for MMIE training compared to the initial MLE results. We can find that by applying MMIE training, the relative improvement of the word error rate (WER) on the evaluation set and development set was 6.2% and 3.0%, respectively. The different performances between evaluation set and development set had been noted in [9]. This might because the training did not generalize well to the development data.

Table 2. Baseline MLE and MMIE results

Training Criterion	Word Error Rate (WER) %	
	Eval	Dev
MLE	6.63	6.99
MMIE	6.22	6.78

6.3. Lattice Pruning Results

The lattice pruning was only applied to the denominator lattices before they were merged with the numerator lattices. To perform beam pruning, different beam widths, such as 1E-80, 1E-70, 1E-60, etc. were used during the decoding of the training utterances. The lattices generated from the beam width of 1E-80 were selected as the baseline lattices to be further pruned using the posterior probability lattice pruning method. Beam widths of 1E-70, 1E-50, etc. were tried for the arc pruning and beam widths of 1E-10 and 1E-5, etc. were tried for the node pruning. When combining the arc and the node pruning, beam width of 1E-10 was fixed for the node pruning and beam widths of 1E-70, 1E-50, etc, were tried for the arc pruning.

We used the following measurements to evaluate the effect of lattice pruning:

- Lattice density: the number of nodes in the lattice divided by the actual number of words in the correct transcription. It is an estimate of the lattice complexity.
- Lattice WER: the best WER of any path in the lattice. It is a lower bound on WER which can be obtained by rescoring the lattice. It can be considered as a rough measurement of the lattice quality.
- The standard WER: the WER of the recognition results on the evaluation and development set after MMIE training with the pruned lattices.

Fig. 3 shows the relationship between the lattice density and the lattice WER of different lattices generated from the beam pruning results and the posterior probability pruning results. We can find that in contrast to beam pruning and the arc pruning, the node pruning can only compress the lattices to a certain degree. This is because node pruning only removes the duplicate entry times of a word hypothesis, even if we only keep the best one, there is still at least one entry time for every different word hypothesis. It can also be seen as the

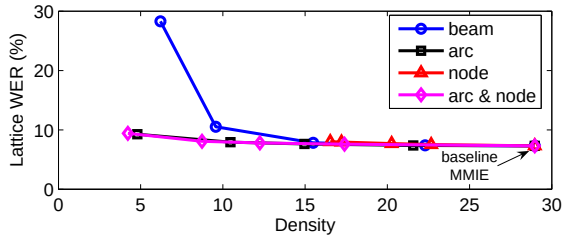

Fig. 3. The relationship between the lattice density and the lattice WER

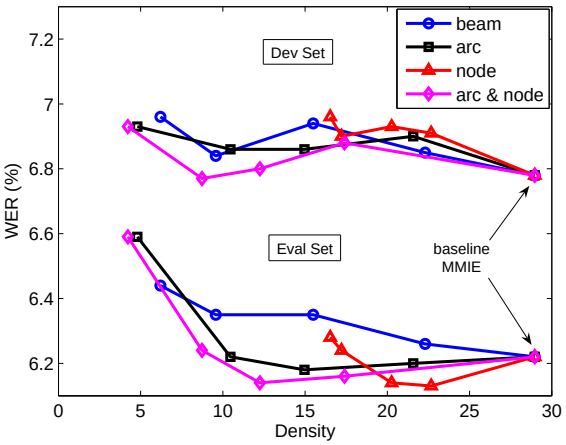

Fig. 4. The relationship between the lattice density and the MMIE performance on the Nov. '92 5k-word task development and evaluation set

lattices become smaller, there is a large gap between the beam pruning curve and the arc pruning curve. This indicates that arc pruning can produce smaller lattices with high quality, while beam pruning cannot. In addition, when combining the arc and node pruning, the lattices can be further pruned a little bit.

The relationship between the lattice density and the performance of MMIE training with different lattices is shown in Fig. 4. For those four lattice pruning methods, as the lattices become smaller, the WER increases. Furthermore, compared to the beam pruning results, using posterior probability lattice pruning, we can perform MMIE training with lattices only 1/2 or 1/3 of the size of the original lattices without hurting the MMIE performance. And when combining the posterior probability arc pruning and node pruning, we can achieve the best results. In Fig. 4, we can also find that sometimes the MMIE performance on the evaluation set could be even better than the MMIE baseline result when trained with smaller lattices. This might because in a word lattice, there are many word hypotheses with the same word, but different entry or exit times. When doing MMIE training with such lattices, most of time a model just competes with itself but not other confusable models. By applying posterior probability pruning, we actually remove those duplicate word hypotheses from the lattice. As a result, MMIE training may penalize the confusable models more and yield a better model. In addition, it can be seen that the development set curves are not as smooth as those evaluation set ones. Again, this may because the training did not generalize well to the development data.

7. CONCLUSION

Considering the use of the word lattices in MMIE training, different lattice pruning methods were investigated in this paper. Because of the special characteristics of the lattices used in MMIE training: 1) unigram language model probabilities are used in training, 2) the time alignment information needs to be kept for the EBW computation, 3) the Gaussian occupation count is weighted by the posterior probability of a word hypothesis, many lattice pruning techniques can't be applied. So we proposed a posterior probability lattice pruning method to directly remove unnecessary arcs and nodes with low posterior probabilities in the lattice. From the experiments, we found with the benefit of the posterior probability lattice pruning, MMIE training can save about 40% of the total running time and produce the same or even more improvement compared to the baseline MMIE results. And we believe that the techniques described in this paper will eventually be able to support discriminative training in low-resource environments and in live systems.

8. ACKNOWLEDGMENT

This work was supported by the National Science Foundation (Grant 0713441). The authors would also like to thank David Huggins-Daines and Matthew Marge for valuable suggestions and discussions.

9. REFERENCES

[1] L. R. Bahl and P. F. Brown and P. C. de Souza and R.L. Mercer, "Maximum mutual information estimation of hidden Markov model parameter for speech recognition," *Proc. ICASSP-1986*, pp. 49-52, 1986.

[2] V. Valtchev and J. J. Odell and P. C. Woodland and S. J. Young, "MMIE training of large vocabulary speech recognition system," *Speech Communication*, vol. 22, pp. 303-314, 1997.

[3] P. S. Gopalakrishan and D. Kanevsky and A. Nadas and D. Nahamoo, "An inequality for rational functions with applications to some statistical estimation problems," *IEEE Transactions on Information Theory*, vol. 37, pp. 107-113, 1991.

[4] S. Ortmanns and H. Ney, "A word graph algorithms for large vocabulary continuous speech recognition," *Computer Speech and Language*, vol. 11, pp. 43-72, 1997.

[5] Y. Liu and M. P. Harper and M. T. Johnson and and L. H. Jamieson, "The effect of pruning and compression on graphical representation of the output of a speech recognizer," *Computer Speech and Language*, vol. 17, pp. 329-356, 2003.

[6] M. Mohri and M. Riley, "Weighted determinization and minimization for large vocabulary speech recognition," *Proc. EUROSPEECH-1997*, pp. 131-134, 1997.

[7] L. Mangu and E. Brill and A. Stolcke, "Finding consensus in speech recognition: word error minimization and other applications of confusion networks," *Computer Speech and Language*, vol. 14(4), pp. 373-400, 2000.

[8] R. Schluter and R. Muller and F. Wessel and H. Ney, "Interdependence of language model and discriminative training," *Proc. IEEE ASRU Workshop*, pp. 119-122, 1999.

[9] R. Schluter and W. Macherey and B. Muller and H. Ney, "Comparison of discriminative training criteria and optimization methods for speech recognition," *Speech Communication*, vol. 34, pp. 287-310, 2001.

[10] L. Qin and A. Rudnicky, "Implementing and improving MMIE training in SphinxTrain," *CMU Sphinx Workshop for Users and Developers (CMU-SPUD 2010)*, 2010. (submitted)

[11] D. Povey, "Discriminative training for large vocabulary speech recognition," Ph.D. thesis, Cambridge University, 2004.

[12] "The CMU Sphinx group open source speech recognition Engines," http://cmusphinx.sourceforge.net/html/cmusphinx.php.