SMOOTHED LANGUAGE MODEL INCORPORATION FOR EFFICIENT TIME-SYNCHRONOUS BEAM SEARCH DECODING IN LVCSR

Daniel Willett, Erik McDermott, Shigeru Katagiri

Speech Open Lab, NTT Communication Science Laboratories, NTT Corporation
2-4, Hikaridai, Seika-cho, Soraku-gun, Kyoto, Japan
{willett,mcd,katagiri}@cslab.kecl.ntt.co.jp

ABSTRACT

For performing the decoding search in Large Vocabulary Continuous Speech Recognition (LVCSR) with Hidden Markov Models (HMMs) and statistical language models, the most straight forward and popular approach is the time-synchronous beam search procedure. A drawback of this approach is that the time-asynchrony of the language model weight application during search leads to performance degradations. This is particularly so, when performing the search with a tight pruning beam. This study presents a method for smoothing the language model within the recognition network. The optimization goal is the smearing of transition probabilities from HMM state to HMM state in favor of a more time-synchronous language model weight application. In addition, state-based language model look-ahead is proposed and evaluated. Both language model smoothing techniques lead to a remarkable improvement in accuracy to run-time ratio, while their combined application yields only limited improvements.

1. INTRODUCTION

Recently, with the introduction of the concept of representing the search space of LVCSR as a weighted finite state network [1], the time-synchronous Viterbi decoding procedure with a single pass that incorporates the full statistical models became feasible even for state-of-the-art systems and tasks that involve cross-word context HMMs and n-gram language models [2]. In the weighted finite state network, network arcs are labelled with input and output symbols. Input symbols are HMMs or HMM states that are consumed along the arc and output symbols are the dictionary words. Language model likelihoods and HMM transition probabilities are compiled into this network as weights that the arcs are labelled with. The major concept of the Viterbi beam search procedure in this type of network is a fully time-synchronous breadth-first search strategy that extends possible decoding paths one frame after the other, while it neglects (*prunes*) those transitions whose log-likelihood drops below a certain beam threshold. In this procedure, the observation probabilities derived from the acoustic models can be applied truly time-synchronously with a log-likelihood share per frame and HMM state. In contrast, the time-synchronous application of language model weights is prohibitive, as these are defined per word and not per frame and can only be applied once the current word of a decoding path becomes unique, which might in the extreme case be only at the very end of the word. The asynchrony of the language model weight application leads to severe performance degradation, especially when a rather tight pruning beam is applied, which is usually the case whenever the computational cost for performing the decoding search is a critical issue.

The problem we are trying to tackle is best explained using an example. Figure 1 shows an excerpt of a recognition network. From the leftmost node, which might itself represent the n-gram node of a specific context, the words "DA", "SAKANA" and "DAKARA" have the language model likelihoods 0.05, 0.002 and 0.0005 respectively. The weights that the arcs are labelled with are minus log-likelihoods and the language model scale is 10.0.

The off-line precompilation of this network allows arbitrary off-line operations on it. In the network of Figure 1, language model look-ahead [3, 4] has been performed. This procedure factorizes the language model weights so that the minimum minus log-likelihoods are incorporated into the search as early as possible. It has been observed that this kind of factorization leads to a dramatic speed up of the beam search procedure. This is due to two reasons. On the one hand, language model look-ahead enables an earlier application of language model information, which leads to earlier pruning of potentially wrong decoding paths. On the other hand, it makes the language model incorporation smoother to some extent, which reduces the danger of pruning a path only because other paths appear to be better that simply do not have the current word's lm-score added to their score yet.

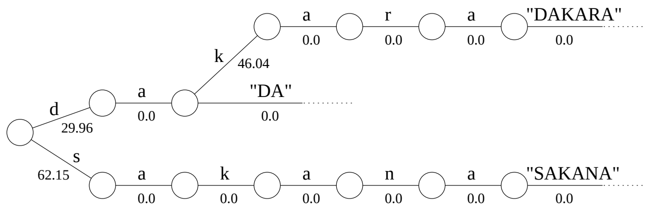

Fig. 1. Baseline recognition network with language model look-ahead performed

However, despite this language model factorization, the language model information is incorporated into the search not time-synchronously, but only when entering very particular arcs. Looking at Figure 1, it is obvious that although "SAKANA" might globally be the best choice for a specific part of the utterance, its large weight at the entering s-arc might cause the pruning of those hypothesis that are about to enter the s-HMM against those entering the d-HMM, simply because the d-arc does not have the full “DAKARA” weight. Thus, the beam search decoding might decode “DAKARA”, although “SAKANA” is the candidate that globally would offer the better score.

The weight smearing approach described in the following paragraph aims at reducing the number of this kind of pruning error by synchronizing the language model weight adaptation with time. Paragraph 3 then describes how to perform the language model look-ahead on the state level making use of state clustering information, which indirectly also leads to a slightly delayed, but smoother language model application.

2. LANGUAGE MODEL SMOOTHING

From the previous paragraph, it is obvious that for good beam search performance, two slightly contradictory goals have to be followed as far as the language model weight application is concerned. On the one hand, the (factorized) language model should be made use of at an early stage, and on the other hand, the application of the language model should be synchronized with time, in order to reduce the number of hypothesesthat get pruned just because their language model incorporation is far ahead of those of other paths.

2.1. Weight smoothing within the WFST

In order to account for both issues, our approach to language model smearing is to start from the network as displayed in Figure 1, in which the full look-ahead has been performed. In this network then, weights are pushed backwards in an iterative smearing procedure.

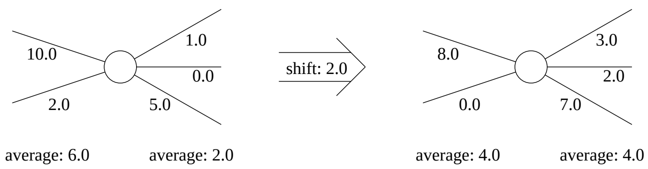

Fig. 2. Weight relaxation within the smoothing procedure

The iterative procedure consists of computing for every network node the average weight of incoming and the average weight of leaving arcs. In case the average incoming weight exceeds the average outgoing weight, weight adjustment is performed in a way that half the difference is shifted so that afterwards incoming and outgoing average weights of this particular node are equal, as displayed in Figure 2.

This local balancing procedure results in unbalanced weights at neighboring nodes. Therefore, multiple iterations of this procedure are performed. After some iterations, performed off-line on the precompiled network, this procedure finally ends up with the network as displayed in Figure 3 for the example introduced in the previous section. Nodes with leaving arcs that are pointing backwards, i.e. nodes with leaving arcs that are pointing to nodes closer to the network's entry node, are omitted from this smearing procedure.

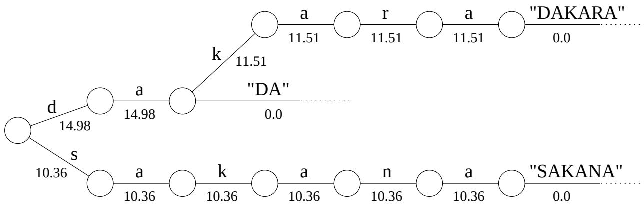

Fig. 3. Smoothed recognition network

2.2. Arc internal smoothing

In our beam search decoder, network arcs represent arbitrary linear sequences of HMM states. This might only be a single HMM state or a long linear sequence of concatenated HMM-states. Thus, a single arc itself represents a linear network, in which, additionally to the smearing described above, the arcs' weight can be applied smoothly.

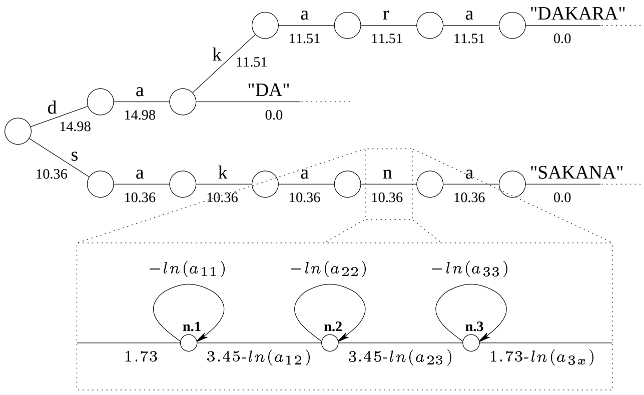

Fig. 4. Minus log-likelihood of 10.36 applied smoothly within the HMM

For the example network, in which the arcs represent three state HMMs, the arc internal smoothing is illustrated in Figure 4. Each state to state transition is given an equal share of the arc's overall weight. Entry and exit transition both only account for half of this equal share, as they do not connect HMM states directly, but build state to state transitions only in connection with other entry and exit transitions.

2.3. Remarks on network topology

The approach as outlined so far tried to smooth the language model application during beam search by shifting weights within the fixed given network structure without changing the topology itself. However, in favor of language model smoothness the network topology itself could be subject of optimization. So, as has already been stated in [5], sacrificing network determinization for earlier language model application can have a positive effect on recognition performance. In this respect, the state-based language model look-ahead as described in the next paragraph can be regarded as a network topology adjustment that leads to a delayed, but smoother language model weight application.

3. STATE-BASED LANGUAGE MODEL LOOK-AHEAD

In state-of-the-art recognition systems, state clustering is essential for obtaining robust models for unseen states and states with too little training data. The most popular approach is tree-based state clustering [6]. The recognition network as described so far used HMM names as network arc labels. When applying this kind of network in decoding, the expansion of the HMM names into sequences of (possibly clustered) states (see Fig. 4) is performed on-line by the decoder itself. However, performing the HMM to state expansion off-line as an operation on the recognition network, offers the opportunity to merge arcs that are labelled with clustered HMM states and to perform the language model look-ahead on this state level. Assuming the (admittedly rather unlikely) clustering of the first states of d and s and also assuming, they share self- and exit-transition probability, the expansion of these HMMs into a network in which arcs represent single states looks as follows:

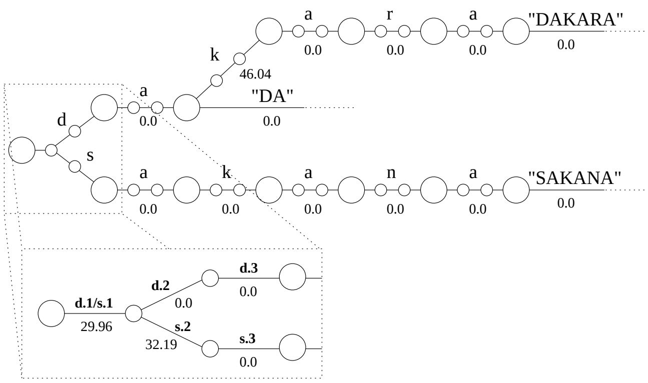

Fig. 5. State-based language model look-ahead

For readability, self-transition loops and HMM state transition probabilities have been omitted in the figure. It is obvious that the merging of clustered states leads to a stronger language model factorization. In the example, the full "SAKANA" weight can now only be applied once entering the second state of the s-HMM. This language model delay within the HMM leads to a smoother language model application and can result in an improved decoding performance.

4. EVALUATION

We evaluated the language model smoothing approaches using an HMM-based precompiled recognition network for the recognition of lecture speeches. Vocabulary size is 10k, full cross-word context dependent triphone names and the trigram language model as supplied within the Spontaneous Speech Priority Program [7] are compiled into the network. The performance is measured on four full lecture speech turns of approximately 100 minutes in total length using unsupervised adapted acoustic models, that origin from models set up in the IPA project. They consist of 2,000 tree-based clustered HMM states of 16 Gaussian mixture components each. With its peak performance of only 59.5% word accuracy (or 40.5% word error), it is a particularly difficult task in terms of efficient decoding with only little additional error. The time-synchronous search is organized following very much the principles of token passing as introduced in [8]. The beam pruning is applied in a way that the threshold is set relative to the current best score with a permanent update during frame propagation (see [2]). Run-times are measured on a 667 MHz Compaq Alpha machine.

	HMM-based lm look-ahead		+ smearing according to 2.1 and 2.2	
beam	search error	time	search error	time
120	0.5%	17.700s	0.1%	22.200s
110	1.4%	12.200s	0.2%	15.500s
100	3.4%	8.300s	0.7%	11.900s
90	7.9%	5.300s	1.9%	7.500s
80	15.0%	3.400s	4.5%	5.100s
70	24.7%	2.100s	18.5%	2.400s

Table 1. Measured run-time and recognition accuracy with different pruning beam widths

Table 1 shows the recognition performance in terms of search error and decoding speed observed for different beam widths. Search error refers to the additional word error endured because of pruning. Thus, it resembles the measured word error minus the peak performance of 40.5%. The accuracy achieved with any of the beam widths improves drastically with the introduced smearing approach. This vast improvement in accuracy comes along with only a moderate increase in recognition time, so that decoding in the weight smeared network is considerably faster than the decoding with a wider beam that achieves a similar recognition performance without the smearing.

	state-based lm look-ahead		+ smearing according to 2.1 and 2.2	
beam	search error	time	search error	time
120	0.2%	14.400s	0.2%	19.300s
110	0.6%	10.000s	0.3%	14.200s
100	1.4%	7.500s	0.9%	10.500s
90	4.0%	5.900s	1.9%	7.900s
80	8.1%	3.900s	4.0%	5.900s
70	16.4%	2.600s	9.3%	4.500s

Table 2. Evaluation of the state-based language model look-ahead with and without additional weight smoothing

The left hand side of Table 2 shows run-time and search error achieved with the state-based look-ahead of Section 3. Comparing it with the results in Table 1, it is obvious that the observed search error rate reduction is not as good as is was with the full smearing, but that decoding times are even lower now, which might make this the preferable choice.

The right hand side of Table 2 shows the recognition performance when applying the weight smearing as introduced in Section 2 on the state level network on which the state-based look-ahead has been performed beforehand. We observe another search error reduction relative to a fixed beam width. The run-time to search error ratio, however, only improves slightly for tight pruning beams. For wider beam widths, it rather has a slightly negative effect. These

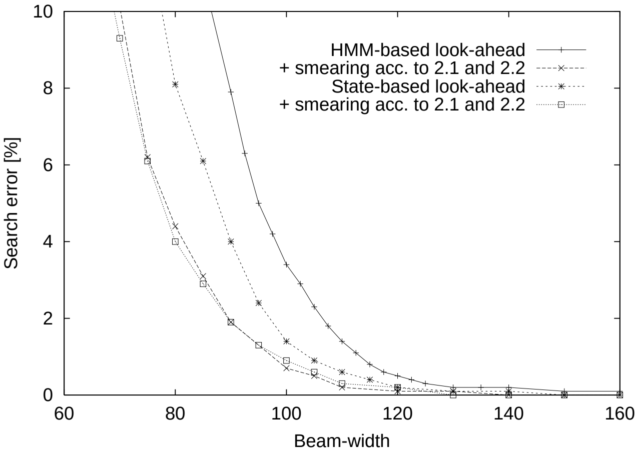

Fig. 6. Beam width to search error ratio

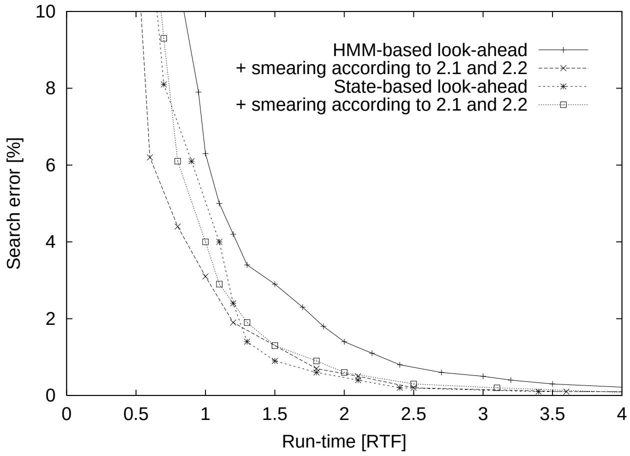

Fig. 7. Run-time [Real-Time Factor] to search error ratio

observations are more obvious looking at Figure 6 and 7.

Figure 6 summarizes the measured ratio between pruning beam width and endured search error. Particularly obvious is that the performance of standard beam pruning vastly improves with the introduced language model smearing approach, and that it is still of good effect when it is applied in addition to state-based language model look-ahead.

However, the performance improvements measured relative to the beam width as displayed in Figure 6 do not directly translate into an improved run-time to accuracy ratio. This is due to the fact that with any type of delay in the language model application bad paths also tend to get pruned later which slows down decoding. Figure 7 shows the run-time to recognition performance ratio. With any of the proposed weight smoothing approaches, the curve approaches the baseline of no search error much more directly than with only the HMM-based language model look-ahead performed. From a run-time of 1.5 RTF and above, the search error observed in the baseline network is at least two times higher than with the smoothing approaches applied. For very fast decoding (between 0.5 and 1.0 RTF), the proposed smearing approach of Section 2 seems to be the best choice.

5. CONCLUSION

This study presented two approaches of smoothing the language model within the precompiled recognition network of a standard speech recognition task. Optimization goal of the one was the relaxation of transition probabilities from HMM state to HMM state while preserving the network's overall topology. The other consisted of precompiling the recognition network down to the HMM state level while merging clustered HMM states and of performing the language model lookahead on the state level. Both approaches offer a considerably better tradeoff between decoding speed and accuracy over conventional beam search in an HMM-based language model look-ahead network. The experimental results strongly suggest to perform either the weight smoothing or the state-based language model lookahead for robost performance in recognition scenarios that are limited by some computational demands, like in decoding under real-time constraints.

Acknowledgement. We thank the Japanese Science and Technology Agency Priority Program "Spontaneous Speech: Corpus and Processing Technology" for providing the speech data, language model and dictionary, and also the IPA project for providing the acoustic models. Furthermore, we thank MIT's SLS group for providing the tools that network minimization was performed with. Thanks also to Lee Hetherington for a couple of fruitful discussions that inspired parts of the work.

6. REFERENCES

[1] F. Pereira, M. Riley, "Speech Recognition by Composition of Weighted Finite Automata", in Finite State Language Processing, MIT Press, 1997.

[2] D. Willett et al., "Time and Memory Efficient Viterbi Decoding for LVCSR using a Precompiled Search Network", Eurospeech'01.

[3] V. Steinbiss, B.-H. Tran, H. Ney, "Improvements in Beam Search", ICSLP'94, pp. 2143–2146.

[4] S. Ortmans et al., "Look-Ahead Techniques for Fast Beam Search", ICASSP'96, pp. 1783–1786.

[5] C. Neukirchen, D. Willett, G. Rigoll, "Reduced Lexicon Trees for Decoding in a MMI-Connecionist/HMM Speech Recognition System", Eurospeech'97, pp. 2639–2642.

[6] J. J. Odell, "The Use of Context in Large Vocabulary Speech Recognition", PhD thesis, University of Cambridge, 1996.

[7] S. Furui et al., "Toward the Realization of Spontaneous Speech Recognition - Introduction of a Japanese Priority Program and Preliminary Results", ICSLP'00, pp. 518–521.

[8] S. J. Young et al., "Token Passing: a Simple Conceptual Model for Connected Speech Recognition Systems", Technical Report TR38, Cambridge University, 1989.

[9] M. Mohri, M. Riley, "Network Optimizations for Large Vocabulary Speech Recognition", Speech Communication, 25:3, 1998.