LANGUAGE-MODEL LOOK-AHEAD FOR LARGE VOCABULARY SPEECH RECOGNITION

S. Ortmanns, H. Ney, A. Eiden

Lehrstuhl für Informatik VI, RWTH Aachen – University of Technology,
D-52056 Aachen, Germany

ABSTRACT

In this paper, we present an efficient look-ahead technique which incorporates the language model knowledge at the earliest possible stage during the search process. This so-called language model look-ahead is built into the time synchronous beam search algorithm using a tree-organized pronunciation lexicon for a bigram language model. The language model look-ahead technique exploits the full knowledge of the bigram language model by distributing the language model probabilities over the nodes of the lexical tree for each predecessor word. We present a method for handling the resulting memory requirements. The recognition experiments performed on the 20 000-word North American Business task (Nov.'96) demonstrate that in comparison with the unigram look-ahead a reduction by a factor of 5 in the acoustic search effort can be achieved without loss in recognition accuracy.

1. INTRODUCTION

This paper describes a language-model (LM) look-ahead technique for large vocabulary continuous speech recognition. The basic idea of this technique is to incorporate the language model probabilities as early as possible in the pruning process of the beam search strategy.

In the context of the word conditioned tree search algorithm, this method seems to require huge memory costs. Therefore, in [9], an approximation based on a unigram language model was used successfully. However, if the bigram language model is fully integrated into the beam search, the beam search method is expected to result in a higher efficiency. So, in [2], a large static state network including the bigram language model was proposed and successfully tested. However, this method exploits the special structure of the bigram language model and is less efficient if the bigram language model is fully trained, i.e. if most of the relevant word bigrams were seen in training.

Therefore, in this paper, we present a method for incorporating any type of bigram language model in the tree-based beam search strategy. To use the look-ahead for a bigram language model, we have to factor the bigram probabilities over the nodes of the (prefix) lexical tree for each copy of the lexical tree [1, 6, 8]. Therefore in principle, we have to keep a huge table in computer memory, containing the factored language model probabilities for each tree node and for each tree copy. In this paper, we present a couple of techniques in order to reduce these memory costs and to obtain an efficient implementation of the LM look-ahead:

- To reduce the memory requirements, we first generate a compressed LM look-ahead tree by eliminating those arcs of the lexical tree that only have one successor arc. The construction of this compressed tree can be performed in a pre-processing step.
- The LM look-ahead tree probabilities are computed only for those tree copies that are hypothesized during the search process. To compute these look-ahead probabilities efficiently on demand, we use a backward dynamic programming scheme.

The paper is organized as follows. In Section 2 we will give a brief review of the standard pruning methods which are integrated into the word conditioned tree search algorithm. In Section 3, the language model look-ahead technique will be described along with a method for handling the memory requirements. To measure the improvements by the bigram look-ahead, we will analyze the search space in Section 4. The recognition experiments have been carried out on the North American Business task (20 000-word vocabulary, NAB'94 H1 development corpus).

2. BEAM SEARCH AND PRUNING METHODS

In this section, we briefly describe the pruning methods that are used in the so-called word conditioned tree search algorithm. This algorithm is based on a strictly time-synchronous left–to–right search method combined with a tree-organized pronunciation lexicon [4].

To incorporate a bigram language model, we use the word conditioned tree search algorithm. For each predecessor word v, we introduce a separate copy of the lexical tree so that during the search process we know the predecessor word v when a leaf of the lexical tree, e.g. the final state of a word w, is reached. This allows us to apply the bigram probability $p(w|v)$. To formulate the dynamic programming approach, we introduce the following quantity [5, 7]:

$Q_v(t, s)$:= score of the best path up to time t that ends in state s of the lexical tree for predecessor v.

The dynamic programming recursion for $Q_v(t, s)$ in the word interior is:

$$Q_v(t,s) \quad = \quad \max_{\sigma} \ \{ \ q(x_t, s|\sigma) \cdot Q_v(t-1,\sigma) \ \} \ ,$$

where $q(x_t, s|\sigma)$ is the product of transition and emission probabilities of the underlying Hidden Markov Model. At the word level, we have to find the best predecessor word for each word hypothesis. For this purpose, we define:

$$H(w;t) \quad := \quad \max_{v} \ \{ \ p(w|v) \cdot Q_v(t, S_w) \ \} \ ,$$

where S_w denotes a terminal state of the lexical tree for word w. To start up new words, we have to initialize $Q_v(t, s)$ as:

$$Q_v(t-1,0) \quad = \quad H(v;t-1) \ ,$$

where the fictitious state $s = 0$ is used to initialize a tree.

Since full search is prohibitive, we use the time synchronous beam search strategy, where at each time frame only the most promising hypotheses are retained. The pruning approach consists of three steps that are performed every 10-ms time frame [9]:

- *Standard beam pruning* or so-called acoustic pruning is used to retain only hypotheses with a score close to the best state hypothesis for further consideration. Denoting the best scored state hypothesis by

  $$Q_{AC}(t) := \max_{(v,s)} \ \{ \ Q_v(t,s) \ \} \ ,$$

  we prune a state hypothesis $(s, t; v)$ if:

  $$Q_v(t,s) < f_{AC} \cdot Q_{AC}(t) \ .$$

  The so-called beam width, i.e. the number of surviving state hypotheses, is controlled by the so-called acoustic pruning threshold f_{AC}.

- *Language model pruning* is applied only to hypotheses of tree start-ups as follows. At word ends, the bigram probability is incorporated into the accumulated score, and the best score for each predecessor word is used to start up the corresponding tree. The scores of these tree start-up hypotheses are subjected to an additional pruning step:

  $$Q_{LM}(t) := \max_{v} \ \{ \ Q_v(t, s=0) \ \} \ .$$

  Thus a hypothesis is removed if:

  $$Q_v(t, s=0) < f_{LM} \cdot Q_{LM}(t) \ ,$$

  where f_{LM} is the so-called LM pruning threshold.

- *Histogram pruning* limits the number of surviving state hypotheses to a maximum number (*MaxHyp*). If the number of active states is larger than *MaxHyp*, only the best *MaxHyp* hypotheses are retained. This pruning method is called histogram pruning because we use a histogram of the scores of the active states [9].

3. LANGUAGE MODEL LOOK-AHEAD

3.1. Basic Concept

The basic idea of this pruning method is to incorporate the language model knowledge as early as possible into the search process. This is achieved by factoring the language model probabilities over the nodes of the lexical tree. For a bigram language model, the factored LM probability $\pi_v(s)$ for state s and predecessor word v is defined as:

$$\pi_v(s) \ := \ \max_{w \in \mathcal{W}(s)} p(w|v) \ ,$$

where $\mathcal{W}(s)$ is the set of words that can be reached from tree state s. The term $p(w|v)$ denotes the conditional bigram probabilities. Strictly speaking, we should use the tree nodes rather than the states of the Hidden Markov models that are associated with each node. However, each initial state of a phoneme arc can be identified with its associated tree node.

After the LM look-ahead tree factorization, i.e. computing $\pi_v(s)$, each node (or phoneme arc) of a lexical tree copy corresponds to the maximum bigram probability over all words that are reachable from this specific node with predecessor word v. An example is shown in Fig. 1.

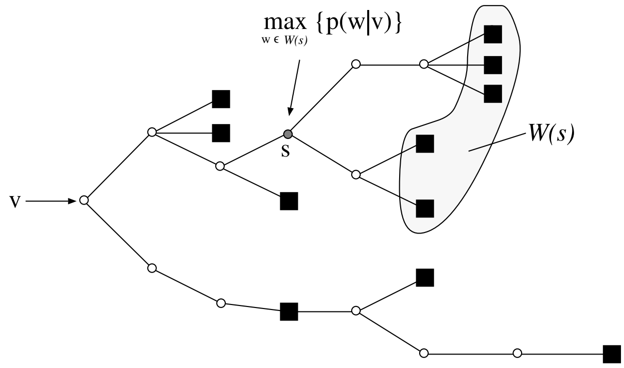

Figure 1: Illustration of LM tree factorization.

We incorporate the factored LM probabilities $\pi_v(s)$ into the dynamic programming recursion across phoneme boundaries:

$$Q_v(t,s) \quad = \quad \frac{\pi_v(s)}{\pi_v(\tilde{s})} \ \cdot \ \max_{\sigma} \ \{ \ q(x_t, s|\sigma) \cdot Q_v(t-1,\sigma) \ \} \ ,$$

where $\tilde{s}$ is the parent node of s. For state transitions not involving phoneme boundaries, we have to use the same equation as described in Section 2. To compute the start-up score $H(w, t)$, we have the dynamic programming equation:

$$H(w;t) \quad := \quad \max_{v} \ \{ Q_v(t, S_w) \ \} \ .$$

Strictly speaking, this equation has to be modified when the word end represented by the state S_w is associated with a *word interior* node of the tree. This happens if a word is at the same time a prefix of another word. As a result of this LM look-ahead, we can use a tighter acoustic pruning threshold f_{AC} in the acoustic pruning as the recognition experiments will show.

When computing all entries of the table $\pi_v(s)$ beforehand, we have to keep a huge table in main memory. In our recognition experiments, the lexical tree consists of 63 000 phoneme arcs which are made up from an inventory of 4688 context dependent phoneme models for the 20 000-word NAB task. Therefore, about $20\,000 \cdot 63\,000$ LM factored probabilities would have to be stored. Since the size of this table is prohibitive, we use a different approach. The main idea is to calculate the LM factored probabilities on demand, i.e. only for those tree copies for which active state hypotheses exist.

To reduce the memory and computational cost, this approach of on-demand calculation is further refined by additional steps which we describe in detail in the following.

3.2. Compression of the Lexical Tree

The memory cost for storing the LM look-ahead probabilities depends on the number of nodes of the original pronunciation tree. This tree can be compressed because there are many tree nodes that have only one successor node. In the NAB'94 task, the number of nodes is thus reduced from 63155 to 29270 nodes, i.e. more than halved. In general, to represent W words, a compressed tree never needs more than $2 \cdot W$ nodes. To provide a mapping from the original lexical tree on the compressed tree, we use an additional mapping array. An example of a compressed LM look-ahead tree is given in Fig. 2. In this example, the lexical tree copy depends on predecessor word v and includes 11 words. Furthermore in this example we have $p(w_{i+1}|v) > p(w_i|v)$ for $i = 1, ..., 10$. A further reduction of the memory cost can be achieved without significant loss in the recognition accuracy if we only consider the first 2-4 arc generations of the lexical tree.

3.3. Factorization of the LM Look-Ahead Tree

Instead of calculating the LM factored probabilities for all possible tree copies beforehand, we calculate the LM factored probabilities on demand for each new tree copy depending on predecessor word v. In a typical case, we have a maximum of, say, 300 active trees. So before computing the LM factored probabilities, we have to check whether these probabilities for the required tree copy exist in the lookup table or not.

A dynamic programming procedure allows us to compute the LM factored probabilities in an efficient way. We initialize the leaves of the LM look-ahead tree with the bigram language model probabilities $p(w|v)$. Then the LM factored probabilities are propagated backwards from the tree leaves to the tree root. For each node, the successor node with maximum look-ahead probability is selected.

4. RECOGNITION EXPERIMENTS

The experimental analysis of the language model look-ahead technique was performed on the North American Business (NAB, Nov.'94) H1 development corpus. The test set contains 310 sentences with 7 387 spoken words. In the experiments, we used a 20 000-word vocabulary and a bigram language model with a test set perplexity of $PP_{bi} = 205.4$. 199 of the spoken words were not part of the vocabulary. The corresponding lexical tree of the 20 000-word vocabulary consists of 63 155 phoneme arcs which are distributed over 17 arc generations. The training of the emission probability distributions was carried out on the WSJ 0 and WSJ 1 training data as described in [3].

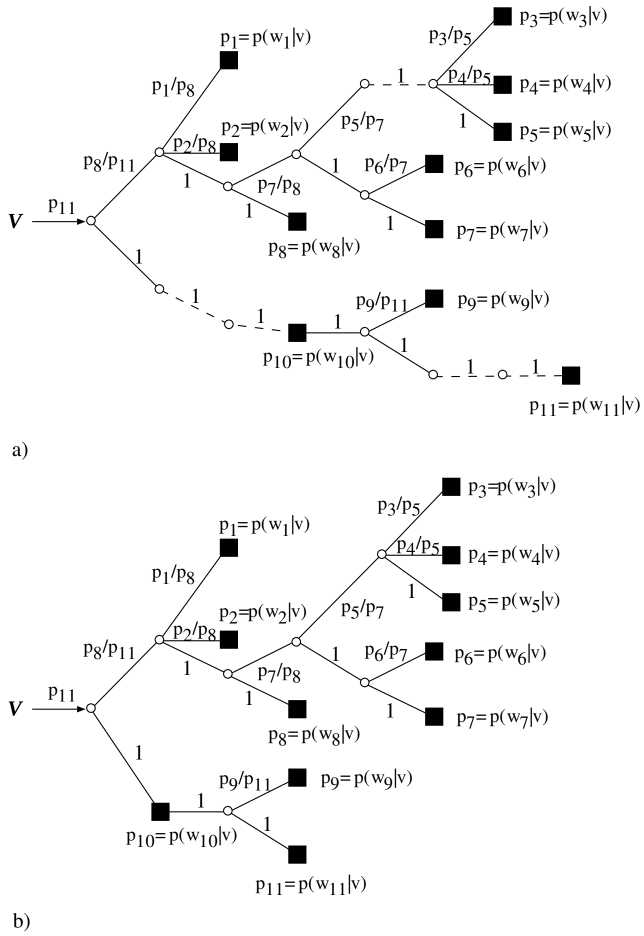

Figure 2: LM look-ahead tree: a) before compression, b) after compression.

Table 1 shows the results for various LM look-ahead types. For each LM look-ahead type the table shows the total number of arcs, arc generations and the maximum number of LM look-ahead trees. In addition, the search space and the recognition word error rate (WER[%]) are given. In an initial experiment, we performed tests without any language model look-ahead. To achieve a word error rate of 16.6 %, 65 907 states per time frame were needed. For comparison purposes, we tested two approximations for the LM look-ahead, namely the unigram [9] and the so-called maximum bigram approximation which is defined as $\pi(s) := \max_v\{\pi_v(s)\}$. For the unigram approximation, we used three different values for the acoustic pruning threshold f_{AC}. The unigram approximation seems to be slightly more efficient than the max-bigram approximation and

Table 1: Effect of the LM look-ahead on the search effort and recognition results for the NAB'94 H1 development set using a bigram language model ($PP_{bi} = 205.4$).

type of LM look-ahead	LM look-ahead tree generations	arcs	trees	search space states	arcs	trees	DEL–INS	WER[%]
no look-ahead	–	–	–	65907	17050	21	180 - 197	16.6
	–	–	–	50299	13129	20	182 - 193	16.8
unigram	17	63155	1	26493	7223	46	179 - 195	16.5
($PP_{uni} = 972.6$)	17	63155	1	16377	4509	37	180 - 196	16.6
	17	63155	1	9333	2593	28	181 - 197	16.8
max-bigram	17	63155	1	23183	6056	16	179 - 200	16.7
bigram	17	29270	300	3385	918	13	180 - 199	16.6
($PP_{bi} = 205.4$)	4	18625	300	3317	934	13	180 - 198	16.6
	3	12002	300	3329	936	13	181 - 197	16.7
	2	4097	300	3676	1029	13	182 - 203	17.0

Table 2: Computational effort using a unigram and a bigram (29270 arcs) LM look-ahead (subset of NAB'94 H1 development corpus: 10 female speakers, 10 sentences = 273 spoken words = 107 sec, $PP_{bi} = 241.6$). The experiments were run on a SGI workstation (Indy R4600, SpecInt'92: 61).

look-ahead	unigram		bigram	
search: states/arcs/trees	17643/4837/40		3582/997/14	
time	[sec]	[%]	[sec]	[%]
log-likelihood comput.	7210	77.3	4990	73.6
LM look-ahead	–	–	990	14.6
acoustic search	1830	19.6	570	8.4
LM recombination	70	0.7	50	0.7
other operations	220	2.4	180	2.7
overall recognition	9330	100.0	6780	100.0

reduces the search space by a factor of 3.5 without any loss in recognition accuracy.

Finally, we tested the bigram LM look-ahead as described in this paper. The acoustic pruning threshold f_{AC} was fixed. Instead, we tested different numbers of arc generations of the LM look-ahead trees, namely 2,3,4 and 17 (which is the full look-ahead tree). Comparing these results with the other results shown in Table 1, we see that the bigram LM look-ahead results in an additional reduction of the search space over the unigram LM look-ahead by a factor of about 5. In comparison with no LM look-ahead, we have a reduction by a factor of about 15 without loss in recognition accuracy.

Assuming a maximum number of 300 active trees, the bigram LM look-ahead needs $29\,225 \cdot 300$ table entries. Using only the first 3 arc generations, this memory cost is reduced down to 40%. Table 2 shows a breakdown of the computational cost over the various operations in search using a unigram and a bigram (29270 arcs) LM look-ahead. In both cases, the log-likelihood calculations for the $290\,000$ densities take about 75 % of the computation time. The bigram LM look-ahead requires 14.6% in comparison with 8.4% for the acoustic search. The cost of the LM look-ahead includes both accessing the bigram probabilities $p(w|v)$ and the LM factorization by dynamic programming.

Acknowledgment. This work was in part supported by the Siemens Aktiengesellschaft, Munich.

REFERENCES

1. F. Alleva, X. Huang, M.-Y. Hwang: "Improvements on the Pronunciation Prefix Tree Search Organization", Proc. Int. Conf. on Acoustics, Speech and Signal Processing, Atlanta, GA, pp. 133 - 136, May 1996.
2. G. Antoniol, F. Brugnara, M. Cettolo, M. Federico: "Language Model Representations for Beam-Search Decoding", Proc. Int. Conf. on Acoustics, Speech and Signal Processing, Detroit, MI, Vol. 1, pp. 588 - 591, May 1995.
3. C. Dugast, R. Kneser, X. Aubert, S. Ortmanns, K. Beulen, H. Ney: "Continuous Speech Recognition Tests and Results for the NAB'94 Corpus", Proc. ARPA Spoken Language Technology Workshop, Austin, TX, pp. 156-161, January 1995.
4. R. Haeb-Umbach, H. Ney: "Improvements in Time-Synchronous Beam Search for 10000-Word Continuous Speech Recognition", IEEE Trans. on Speech and Audio Processing, Vol. 2, pp. 353-356, April 1994.
5. H. Ney: Search Strategies for Large-Vocabulary Continuous-Speech Recognition. NATO Advanced Studies Institute, Bubion, Spain, June-July 1993, pp. 210–225, in A.J. Rubio Ayuso, J.M. Lopez Soler (eds.): "Speech Recognition and Coding – New Advances and Trends", Springer, Berlin, 1995.
6. J.J. Odell, V. Valtchev, P.C. Woodland, S.J. Young: "A One Pass Decoder Design for Large Vocabulary Recognition", Proc. ARPA Human Language Technology Workshop, Plainsboro, NJ, Morgan Kaufmann, pp. 405 - 410, March 1994.
7. S. Ortmanns, H. Ney, F. Seide, I. Lindam: "A Comparison of Time Conditioned and Word Conditioned Search Techniques for Large Vocabulary Speech Recognition", Proc. Int. Conf. on Spoken Language Processing, Philadelphia, PA, October 1996.
8. S. Renals, M. Hochberg: "Efficient Search Using Posterior Phone Probability Estimates", Proc. Int. Conf. on Acoustics, Speech and Signal Processing, Detroit, MI, Vol. 1, pp. 596 - 599, May 1995.
9. V. Steinbiss, B.-H. Tran, H. Ney: "Improvements in Beam Search", Proc. Int. Conf. on Spoken Language Processing, Yokohama, pp. 2143-2146, September 1994.