LOOK-AHEAD TECHNIQUES FOR IMPROVED BEAM SEARCH

S. Ortmanns, H. Ney, A. Eiden, N. Coenen

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

Abstract. This paper presents two look-ahead techniques for large vocabulary continuous speech recognition. These two techniques, which are referred to as language model look-ahead and phoneme look-ahead, are incorporated into the pruning process of the time-synchronous one-pass beam search algorithm. The search algorithm is based on a tree-organized pronunciation lexicon in connection with a bigram language model.

Both look-ahead techniques have been tested on the 20 000-word NAB'94 task (ARPA North American Business Corpus). The recognition experiments show that the combination of bigram language model look-ahead and phoneme look-ahead reduces the size of search space by a factor of about 27 without affecting the word recognition accuracy.

1 Introduction

In this paper, we describe two look-ahead techniques for improved beam search, namely language model look-ahead and phoneme look-ahead, for large vocabulary continuous speech recognition. The basic idea of the language model look-ahead is to fully incorporate the language model (LM), e.g. a bigram or trigram language model, as early as possible into the pruning process of the time-synchronous search algorithm using word dependent copies of the lexical prefix tree (or word models). To use the look-ahead for a bigram language model, we factor the bigram probabilities over the nodes of the (prefix) lexical tree for each copy of the lexical tree [Steinbiss et al. 94, Odell et al. 94, Steinbiss et al. 94, Renals & Hochberg 95, Alleva et al. 96, Ortmanns et al. 96]. In principle, we have to keep a huge table in computer memory, containing the factored language model probabilities for each tree node of each lexical tree copy. To reduce the memory and the computational cost, we present special implementation details which are based on a so-called compressed LM look-ahead tree [Ortmanns et al. 96]. We use a dynamic programming scheme to compute the LM look-ahead probabilities in an efficient way. In addition to the LM look-ahead, we present a phoneme look-ahead which is similar to the method described in [Haeb-Umbach & Ney 94, Ney et al. 92]. The idea of this look-ahead technique is to estimate the likelihood of each phoneme ahead of the current time frame. This probability estimate is then used in an additional pruning step. To reduce the computational cost of the phoneme look-ahead, we introduce suitable simplifications of the phoneme models.

The organization of this paper is as follows. In Section 2, we review the one-pass beam search using a tree-organized pronunciation lexicon in combination with a bigram language model. In Section 3, we present the language model look-ahead. In Section 4, we present the phoneme look-ahead. In Section 5, we give experimental results on the NAB'94 20000-word development data.

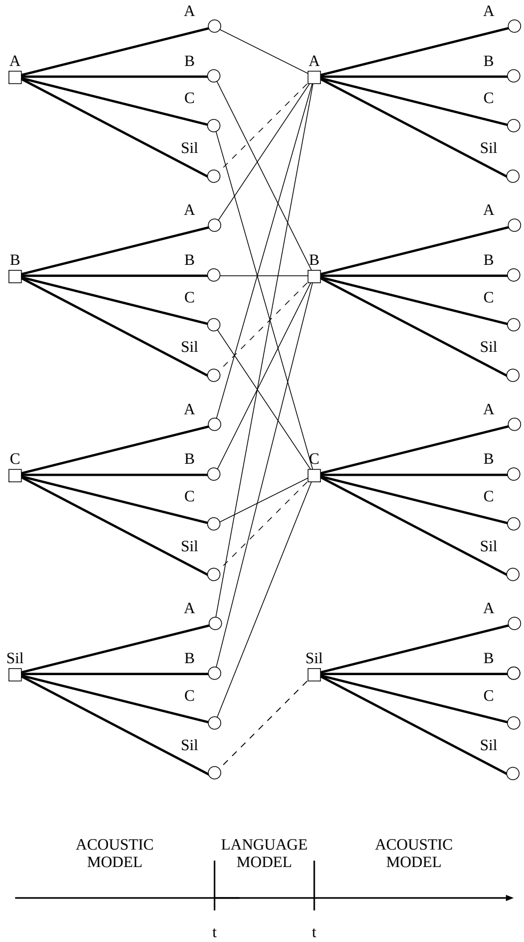

Figure 1: Bigram language model recombination and intraphrase silence handling for a tree lexicon (three-word vocabulary: A,B,C).

2 Review of the Lexical Tree Search Algorithm

2.1 Search Space Organization

In this section, we review the widely used time-synchronous one-pass dynamic programming search algorithm combined with a bigram language model [Ney 93]. For large vocabulary speech recognition, it is a very attractive idea to organize the pronunciation lexicon as a prefix tree. However, in the framework of dynamic programming search, the use of a lexical prefix tree in connection with a bigram language model requires a careful structuring of the search space. Typically, we face the problem that the identity of the hypothesized word w is known only when a leaf of the lexical prefix tree has been reached. Therefore the language model probabilities can only be fully incorporated after reaching the terminal state of the second word of the bigram. As a result, we can apply the language model probability only at the end of a tree. To make the application of the dynamic programming principles possible, we structure the search space as follows. For each predecessor word v, we introduce a separate copy of the lexical tree so that during the search process we always know the predecessor word v when a word end w is hypothesized.

Fig. 1 illustrates this concept for a three-word vocabulary A, B, C, where the lexical tree is depicted in a simplified schematic form. In the set-up of Fig. 1, we apply the bigram probability $p(w|v)$ when the final state of word w with predecessor v has been reached, and use the resulting overall score to start up the corresponding lexical tree, i.e. the tree that has word w as predecessor. To handle intraphrase silence models, we add a separate copy of the silence model (*Sil*) to each tree. In addition, we have a separate copy of the lexical tree for the first word in the sentence; this tree copy is assigned silence as predecessor word. As a result of this approach, the silence model copies do not require a special treatment, but can be processed like regular words of the vocabulary. However, there is one exception: at word boundaries, there is no language model probability for the silence models. As shown in Fig. 1, there are two types of path extensions and recombinations, namely in the interior of the words or lexical trees and at word boundaries. In the word interior, we have the bold lines representing the transitions in the Hidden Markov models (HMM). At word boundaries, we have the thin and the dashed lines, which represent the bigram language model recombinations. Like the acoustic recombinations, they, too, are performed each time frame (10 ms). The dashed lines are related to recombinations for interphrase silence copies. To start up a new word hypothesis, we have to incorporate the bigram probabilities into the word end scores and to determine the best predecessor word v. This best score is then propagated into the root of the associated lexical tree, which is represented by the symbol $\Box$. The symbol $\circ$ denotes a word end.

For a quantitative specification of the search procedure, we assume that each arc of the lexical tree is represented by a HMM. We will use the state index s directly and assume that the lexical structure is captured by the transition probabilities of the HMM. To formulate the dynamic programming approach, we introduce the following two quantities [Ney 93]:

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

$B_v(t,s)$:= starting time of the best partial path that ends at time t in state s of the lexical tree for predecessor v.

Both quantities are evaluated using the dynamic programming recursion for $Q_v(t,s)$:

$$\begin{aligned} Q_v(t,s) &= \max_{\sigma} \{ \, q(x_t, s|\sigma) \cdot Q_v(t-1,\sigma) \, \} \\ B_v(t,s) &= B_v(t-1, \sigma_v^{max}(t,s)) \ , \end{aligned}$$

where $\sigma_v^{max}(t,s)$ is the optimum predecessor state for the hypothesis (t,s) in the lexical tree of predecessor word v. $q(x_t, s|\sigma)$ is the product of transition and emission probabilities of the HMMs used for the context dependent or independent phoneme models. The back pointers $B_v(t,s)$ are propagated according to the dynamic programming decision. Unlike the predecessor word v, the index w for the word under consideration is only needed and known when a path hypothesis reaches an end node of the lexical tree. Each end node of the lexical tree is labeled with the corresponding word of the vocabulary.

Using a suitable initialization for $\sigma = 0$, this equation includes the optimization over the unknown word boundaries. At word boundaries, we have to find the best predecessor word v for each word w. To this purpose, we define:

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

where the state S_w denotes the terminal state of word w in the lexical tree. To propagate the path hypothesis into the lexical tree hypotheses or to start them up if they do not exist yet, we have to pass on the score and the time index *before* processing the hypotheses for time frame t:

$$\begin{aligned} Q_v(t-1, s=0) &= H(v; t-1) \\ B_v(t-1, s=0) &= t-1 \ . \end{aligned}$$

The details of the algorithm are summarized in Table 1. It should be mentioned that the one-pass dynamic search algorithm which determines the single most likely word sequence can be easily modified to produce a word graph [Aubert & Ney 95].

Table 1: One–pass algorithm using the lexical tree organization.

proceed over time t from left to right			
	ACOUSTIC LEVEL: process states of lexical trees		
		– initialization: $Q_v(t-1, s=0) = H(v; t-1)$ $B_v(t-1, s=0) = \ t-1$	
		– time alignment: $Q_v(t,s)$ using DP – propagate back pointers $B_v(t,s)$	
		– prune unlikely hypotheses – purge bookkeeping lists	
	WORD PAIR LEVEL: process word ends		
		for each pair $(w;t)$ do	
			$H(w;t) = \max_v \ \{ \ p(w\|v) \ Q_v(t, S_w) \ \}$ $v_0(w;t) = \arg\max_v \ \{ \ p(w\|v) \ Q_v(t, S_w) \ \}$
			– store best predecessor $v_0 := v_0(w;t)$ – store best boundary $\tau_0 := B_{v_0}(t, S_w)$

2.2 Standard Pruning Methods

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 [Steinbiss et al. 94]:

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

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

  we prune a state hypothesis $(t, s; v)$ 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 tree start-up hypotheses as follows. At word end hypotheses, 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 hypothesis or is propagated into this tree hypothesis if it already exists. The scores of these tree start-up hypotheses are subjected to an additional pruning step:

  $$Q_{LM}(t) := \max_v \ \{ \ Q_v(t, s=0) \ \} \ ,$$

  where $s = 0$ is a fictitious state used for initialization. Thus a tree start-up hypothesis $(t, s=0; v)$ is removed if:

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

where f_{LM} is the so-called language model 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 and the other hypotheses are removed. This pruning method is called histogram pruning because we use a histogram of the scores of the active states [Steinbiss et al. 94].

The efficiency of these pruning methods is improved by including into the pruning process the so-called look-ahead techniques like phoneme look-ahead [Haeb-Umbach & Ney 94, Ney et al. 92] and language model look-ahead [Alleva et al. 96, Antoniol et al. 95, Odell et al. 94, Ortmanns et al. 96, Renals & Hochberg 95, Steinbiss et al. 94]. In the following two sections, we describe these two look-ahead techniques in detail.

3 Language Model Look-Ahead

3.1 Basic Concept

The basic idea of the language model look-ahead is to incorporate the language model probabilities as early as possible into the search process and thus into the associated pruning 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 (or arcs) 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 via this specific node from predecessor word v. An example is shown in Fig. 2. We incorporate the factored LM probabilities $\pi_v(s)$ into the dynamic programming recursion across phoneme boundaries:

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

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 to take into account that, at the end nodes of the lexical trees, the language model probabilities have already been included. Hence we have simply:

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

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 [Dugast et al. 95, Ortmanns & Ney 95]. 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 more detail in the following.

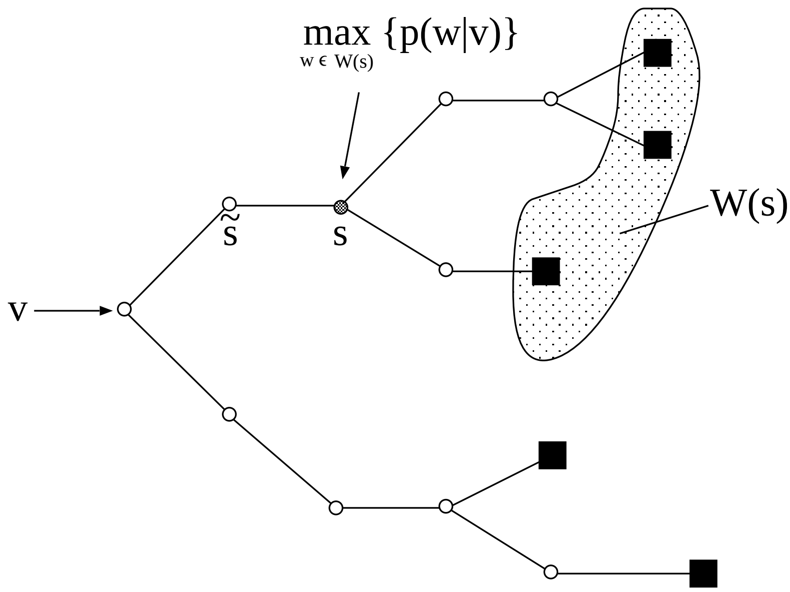

Figure 2: Concept of LM tree factorization.

3.2 Factorization and 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 20 000-word 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 arcs of the original lexical tree on the arcs of the compressed tree, we use an index array. An example of the LM tree factorization before path compression is shown in Fig. 3 (top). This example shows for predecessor word v the lexical tree which consists of 5 words and which has the following bigram LM probabilities: $p(w_1|v) = 0.3$, $p(w_2|v) = 0.2$, $p(w_3|v) = 0.1$, $p(w_4|v) = 0.15$ and $p(w_5|v) = 0.05$. In Fig. 3 (top), each arc is assigned a value between 0 and 1. These values are computed from the factored LM probabilities such that the following property holds. When considering the tree associated with a predecessor word v and following a path from the tree root to a tree end node representing a word w, the product of these values is exactly the LM probability $p(w|v)$. The compressed LM look-ahead tree is shown in Fig. 3 (bottom). A further reduction of the memory cost can be achieved with virtually no loss in the recognition accuracy if we consider only the first 2-4 arc generations of the lexical 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 and store these factored probabilities in a look-up table. In a typical case, this look-up table is able to store the factored probabilities for a maximum of, say, 300 lexical tree copies. So before computing the LM factored probabilities, it is first checked whether the probabilities of the required tree copy exist already 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 by using a dynamic programming recursion, which, for each tree node, determines the successor node with maximum look-ahead probability.

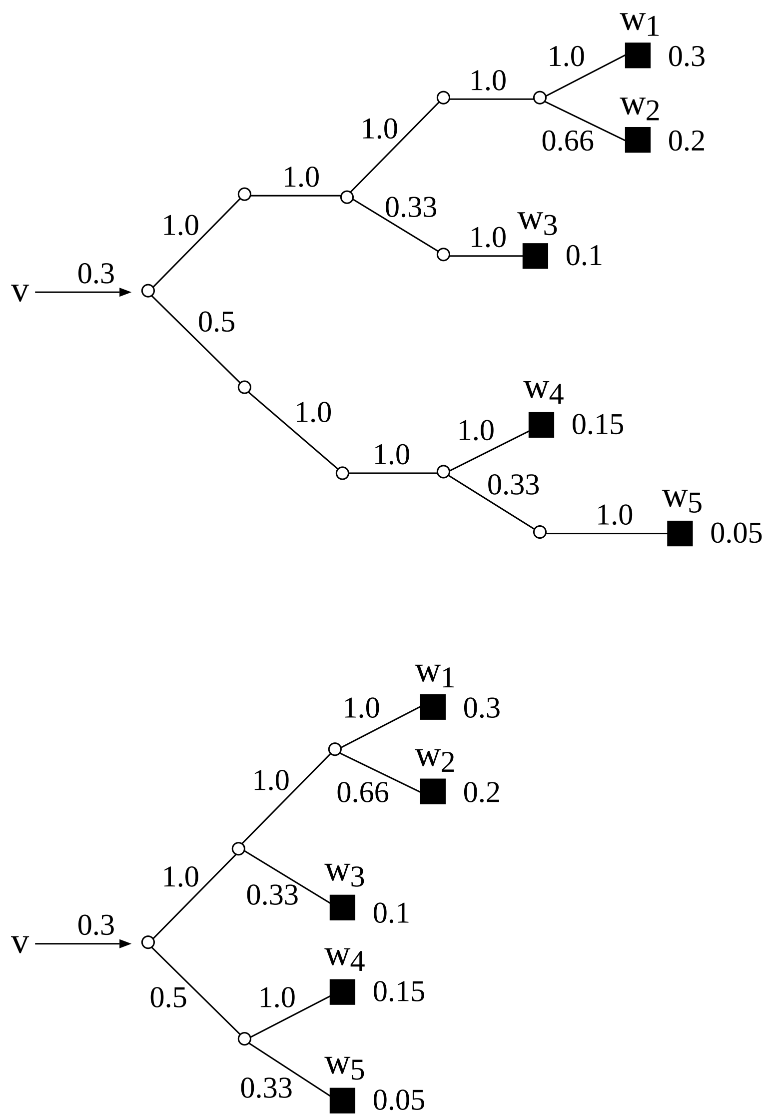

Figure 3: LM tree factorization before (top) and after (bottom) path compression.

4 Phoneme Look-Ahead

4.1 Basic Concept

The phoneme look-ahead is based on the following concept [Ney et al. 92]. Each time a hypothesis is formed about a new phoneme arc to be started in the search process, it is first checked whether this new phoneme arc hypothesis is likely to survive the pruning steps that will be performed for the next future time frames. To this purpose, we compute an approximate probability estimate for each possible phoneme arc that can be activated at a given time frame in the beam search. This approximate probability estimate, which is referred to as look-ahead score, is then combined with the detailed score of its predecessor phoneme and used in an additional pruning step, in which all hypotheses of phoneme arcs to be started up are considered time-synchronously in the usual spirit of beam search.

To formulate the phoneme look-ahead and the associated pruning operation in detail, we use the following notation:

α: one of the phoneme arcs to be started in the lexical prefix tree. Note that the same phoneme arc α may occur in different copies of the lexical tree.

$\tilde{\alpha}$: the unique parent arc of α in the lexical tree, for which one of the final states has been reached in the search process. Note that this mapping $\alpha \rightarrow \tilde{\alpha}$ captures the lexical constraints as given by the pronunciation lexicon.

$\hat{q}(\alpha; t, \Delta t)$: probability that the phoneme α produces the acoustic vectors $x_{t+1}, ..., x_{t+\Delta t}$. Δt is in the order of an average phoneme duration, i.e. 6 or 7 10-ms time frames.

For the phoneme look-ahead pruning, we combine this look-ahead score $\hat{q}(\alpha; t, \Delta t)$ with the detailed score $Q_v(t, s)$. Thus for a given time frame t, we compute the following score for each possible pair (α, v) of phoneme arc α and lexical tree for predecessor word v:

$$\hat{Q}_v(t, \alpha) \quad := \quad \hat{q}(\alpha; t, \Delta t) \cdot Q_v(t, S_{\tilde{\alpha}}) \quad ,$$

where $S_{\tilde{\alpha}}$ denotes the final state of phoneme arc $\tilde{\alpha}$. For notational simplicity, we have assumed that there is exactly one final state. If there are several final states, we select the best one. As in all time-synchronous pruning methods, the pruning is based on computing the best score $Q_{LA}(t)$ of all hypotheses under consideration for time t:

$$Q_{LA}(t) \quad := \quad \max_{(v,\alpha)} \left\{ \hat{Q}_v(t, \alpha) \right\} \quad .$$

A phoneme arc hypothesis (α, v) at time t is removed (or, depending on the viewpoint, not started at all in the detailed search) if

$$\hat{Q}_v(t, \alpha) < f_{LA} \cdot Q_{LA}(t) \quad ,$$

where f_{LA} denotes the phoneme look-ahead pruning threshold. In the experimental tests, we have found that there is no loss in performance when we use the following (or a similar) approximation for $Q_{LA}(t)$:

$$Q_{LA}(t) \quad \cong \quad \max_{\alpha} \{ \hat{q}(\alpha; t, \Delta t) \} \cdot \max_{(v,\beta)} \; \{ \; Q_v(t, S_\beta) \} \quad ,$$

where the symbol β stands for an arbitrary phoneme arc *independent* of phoneme arc α. This means that we do *not* use the lexical constraints when computing the reference score for the pruning step. However for each individual arc hypothesis, it is very important to take the exact lexical constraints into account.

4.2 Calculation of the Look-Ahead Score

In order to compute the phoneme look-ahead score $\hat{q}(\alpha; t, \Delta t)$, we perform a time alignment for each hypothesized phoneme α. To this purpose, we define:

$\phi_\alpha(\tau, s; t)$: score of time aligning the acoustic vectors $x_{t+1}, ..., x_{t+\tau}$ with the states $1, ..., s$ of phoneme arc α.

The time alignment scores $\phi_\alpha(\tau, s; t)$ are computed by dynamic programming. The details and the computational effort depend on the type of phoneme models and of the underlying HMM. In general, we use a 6-state HMM representing a phoneme model [Ney 90]. For such a 6-state HMM, the concept of the look-ahead time alignment is illustrated in Fig. 4. The shadowed area in Fig. 4 marks the potential states in which the look-ahead time alignment path may end.

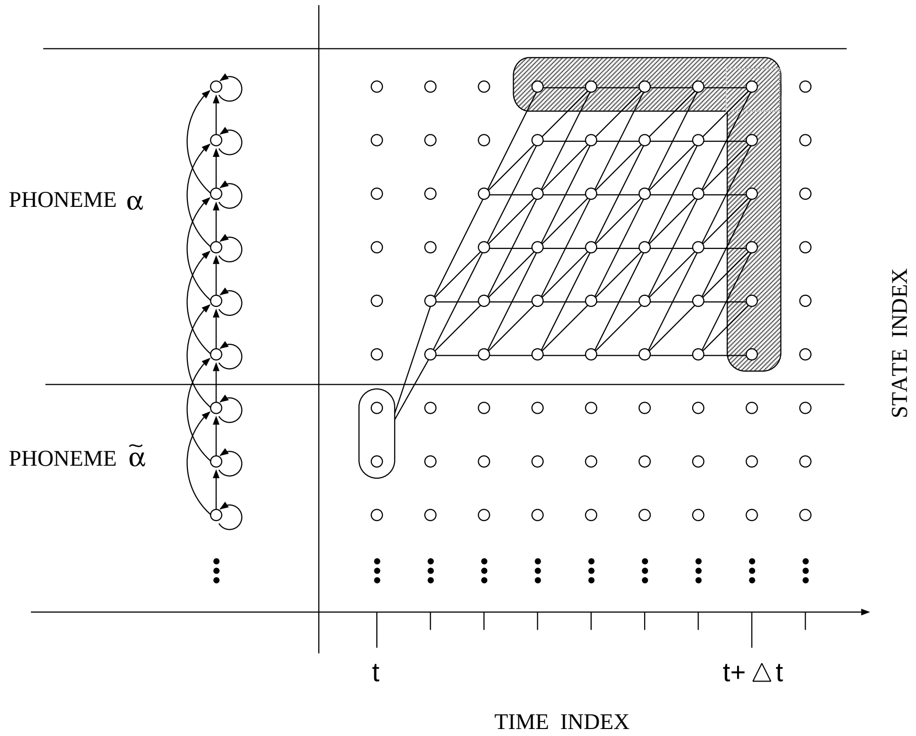

Figure 4: Phoneme look-ahead using a 6-state HMM.

To compute the phoneme look-ahead score $\hat{q}(\alpha; t, \Delta t)$, we have to consider the scores of the potential ending states of the time alignment path. By normalizing the scores with respect to different durations τ, we obtain the following equation for the phoneme look-ahead score $\hat{q}(\alpha; t, \Delta t)$:

$$\hat{q}(\alpha; t, \Delta t) \quad := \quad \max \left\{ \max_s \{\phi_\alpha(\Delta t, s; t)\} \; , \; \max_\tau \left\{ \phi_\alpha(\tau, S; t)^{\Delta t/\tau} \right\} \right\} \quad ,$$

where, as usual, the symbol S stands for the final state of the HMM.

So far, we have not considered the computational cost of computing the time alignment look-ahead scores. Evidently, the phoneme look-ahead can only result in a speed-up of the search process if this additional computational effort is sufficiently small. Using the same phoneme models in both the detailed search and the look-ahead time alignment is prohibitive for the following reason. Like most other speech recognition systems, we use context dependent (CD) phoneme models rather than context independent (CI) phoneme models in the detailed search process. The number of these CD models is typically in the range of several thousands. In addition, for the emission distributions of the HMMs, we use mixture distributions with a huge number of component densities.

Therefore to keep the effort for computing the phoneme look-ahead scores small, we consider the following methods:

- Instead of CD phoneme models, we use CI phoneme models, say $40 - 50$, for the phoneme look-ahead.

- We use only a small number of component densities, e.g a total of a few hundreds, to model the emission distributions.

- The calculation of the phoneme look-ahead can be performed every second time frame [Haeb-Umbach & Ney 94].

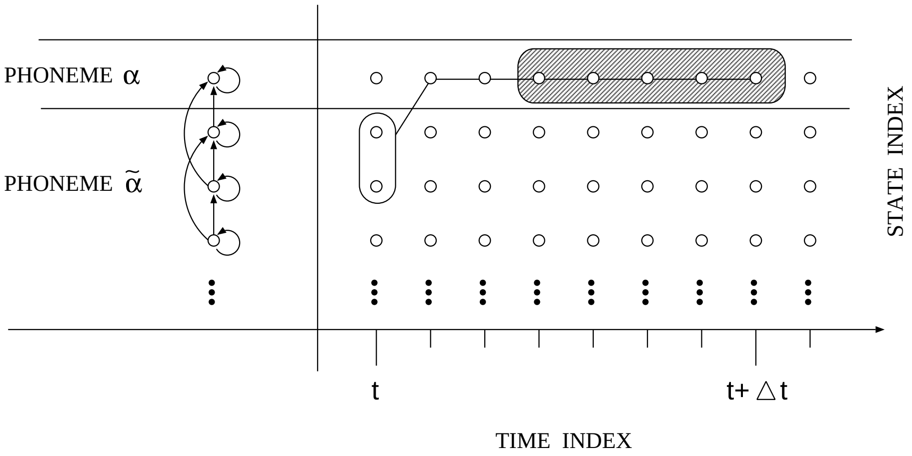

Figure 5: Phoneme look-ahead using a 1-state HMM.

- To further reduce the amount of computation, we simplify the structure of each phoneme by collapsing all states into only one state as shown in Fig. 5 [Bahl et al. 93]. As a result, each model has only one emission probability distribution.

It is possible to extend the phoneme look-ahead approach to a 2-phoneme look-ahead, which attempts to take into account the two successor phonemes of a given ancestor phoneme. However, in this paper, we consider only the 1-phoneme look-ahead.

5 Experimental Results

5.1 Recognition Task and Database

The experimental tests were carried out on the ARPA North American Business (NAB'94) H1 development corpus comprising 310 sentences with a total of 7387 words spoken by 10 male and 10 female speakers. 199 of the spoken words were out-of-vocabulary words. The training of the emission probability distributions of the underlying Hidden Markov models was performed on the so-called WSJ0 and WSJ1 training data as described in [Dugast et al. 95]. In all experiments, we used about 290 000 Laplacian mixture densities (with a single pooled vector of absolute deviations) for each gender and a bigram language model with a perplexity (PP) of 198.4.

5.2 Bigram LM Look-Ahead

First, we investigated the effect of the LM look-ahead on the size of search space and the word error rate. Table 2 shows the results of several recognition tests. For each test, the table shows the set-up of the LM look-ahead tree in terms of the number of arcs and arc generations and of the maximum number of LM look-ahead trees. In addition, the search space, the recognition word error rate (DEL–INS and WER[%]) and the real time factor (RT) are given. The experiments were performed on a SGI workstation with a R4400 processor (91.7 SpecInt92). In an initial experiment, we performed two tests without any language model look-ahead using two different values f_{AC} of the acoustic pruning threshold. To achieve a word error rate of 16.6%, 50020 state hypotheses per time frame are needed on the average. Then we tested the unigram LM look-ahead as described in [Steinbiss et al. 94]. For this unigram LM look-ahead, two recognition experiments were performed using again two values of f_{AC} as shown in Table 2. The unigram LM look-ahead reduces the search space by a factor of about 4 without loss in recognition accuracy. Finally, we tested the bigram LM look-ahead as described in

Table 2: Effect of the LM look-ahead on the search effort and recognition results (NAB'94 H1 development set; bigram LM with $PP = 198.4$).

	LM look-ahead tree			search space			recognition errors		
LM look-ahead	gen.	arcs	trees	states	arcs	trees	DEL–INS	WER[%]	RT
no	–	–	–	65568	16932	26	180 - 186	16.3	139.3
	–	–	–	50020	13034	20	182 - 187	16.6	115.7
unigram	17	63155	1	16960	4641	32	181 - 184	16.4	86.2
($PP = 972.6$)	17	63155	1	9443	2599	22	191 - 184	16.8	68.9
bigram	17	29270	300	3312	935	13	181 - 190	16.5	41.6
($PP = 198.4$)	4	18625	300	3263	922	13	191 - 198	16.5	39.9
	3	12002	300	3277	924	13	179 - 191	16.5	39.8
	2	4097	300	3611	1012	12	178 - 193	16.9	40.8
	1	544	300	5786	1643	11	191 - 207	17.0	45.8

this paper. While keeping the acoustic pruning threshold f_{AC} fixed, we tested various numbers of arc generations used for the LM look-ahead trees, namely 17, 4, 3, 2, 1 as shown in Table 2. The full tree consists of 17 arc generations. We see that the best results are obtained for 3 and more arc generations. The search space is reduced by a factor of about 5 over that of the unigram LM look-ahead [Steinbiss et al. 94]. In comparison with no LM look-ahead, we have a reduction by a factor of about 20 with only a negligible loss in the word recognition accuracy.

5.3 Phoneme Look-Ahead

In a second recognition experiment, we added the phoneme look-ahead to the bigram LM look-ahead and studied the effect on the search effort. In a first series of recognition tests, it was found that the best results were obtained by choosing $\Delta t = 7$ time frames as the anticipatory time of the phoneme look-ahead, which for the 10-ms frame period used is roughly equivalent to an average duration of a phoneme. In contrast to the detailed search, in which 4688 context dependent phoneme models are used, the inventory of the look-ahead phoneme models consists of only 43 context independent phoneme models. In addition, there is a silence model that always comprised a single state. We tested the following variants of the look-ahead models:

- 6-state models with a total of either 498 or 1226 densities. As shown in Fig. 4, each of these models results in a search area of 36 grid points.
- 1-state models with a total of 175 densities. As shown in Fig. 5, each of these models results in a search area of 7 grid points.

Table 3 summarizes the results for different types of HMMs and different numbers of mixture densities used in the phoneme look-ahead. For each condition, two recognition tests were carried out using different values of the pruning threshold f_{LA}. It can be seen that the 6-state look-ahead models produce better results than the 1-state models. Increasing the number of mixture densities in the phoneme look-ahead leads to a small reduction of the search effort. For the tests reported here, the computational cost of the phoneme look-ahead is negligible in comparison with the effort for the detailed search. For the best choice of conditions, the size of the search space *and* the total recognition time are halved while the word error rate goes up only from 16.5% to 16.7%.

Table 3: Effect of the phoneme look-ahead ($\Delta t = 7$) in combination with the bigram LM look-ahead (17 phoneme generations) on the search effort and recognition results (NAB'94 H1 development set; bigram LM with $PP = 198.4$).

phoneme look-ahead		search space			recognition errors		
model	densities	states	arcs	trees	DEL–INS	WER[%]	RT
no	-	3312	935	13	181 - 190	16.5	41.6
6-state HMM	498	2213	548	12	181 - 188	16.6	26.8
	498	1571	370	11	186 - 196	16.8	20.9
	1226	1862	455	12	181 - 182	16.6	23.3
	1226	1589	381	12	182 - 196	16.7	21.2
1-state HMM	175	2255	554	12	181 - 191	16.6	25.8
	175	1551	359	9	179 - 196	16.7	24.1

6 Summary

This paper has presented and studied two look-ahead techniques for large vocabulary continuous speech recognition, namely language model look-ahead and phoneme look-ahead. The experiments performed on the NAB'94 20 000-word task have shown that the combination of the two look-ahead methods leads to a reduction of the size of search space by a factor of about 27 with virtually no loss in the recognition accuracy. Due to the cost of the likelihood calculations for the detailed search, this results in an overall speed-up of the recognition process by a factor of about 5.

References

[Alleva et al. 96] F. Alleva, X. Huang, M.-Y Hwang: Improvements on the Pronunciation Prefix Tree Search Organization. Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing, Atlanta, GA, pp. 133-136, May 1996.

[Antoniol et al. 95] G. Antoniol, F. Brugnara, M. Cettolo, M. Federico: Language Model Representations for Beam-Search Decoding. Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing, Detroit, MI, Vol. 1, pp. 588-591, May 1995.

[Aubert & Ney 95] X. Aubert, H. Ney: Large Vocabulary Continuous Speech Recognition using Word Graphs. Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing, Detroit, MI, pp. 49-52, May 1995.

[Bahl et al. 93] L.R. Bahl, S.V. De Gennaro, P.S. Gopalakrishnan, R.L. Mercer: A Fast Approximate Acoustic Match for Large Vocabulary Speech Recognition. IEEE Trans. on Speech and Audio Processing, Vol. 1, pp. 59-67, January 1993.

[Dugast et al. 95] 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.

[Haeb-Umbach & Ney 94] 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.

[Ney 90] H. Ney: Acoustic Modelling of Phoneme Units for Continuous Speech Recognition. Proc. Fifth European Signal Processing Conference, Barcelona, pp. 65-72, September 1990.

[Ney et al. 92] H. Ney, R. Haeb-Umbach, B.-H. Tran, M. Oerder: Improvements in Beam Search for 10000-Word Continuous Speech Recognition. Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing, San Francisco, CA, pp. 13-16, March 1992.

[Ney 93] 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.

[Odell et al. 94] J. J. Odell, V. Valtchev, P. C. Woodland, S. J. Young: A One-Pass Decoder Design for Large Vocabulary Recognition. Proc. ARPA Spoken Language Technology Workshop, Plainsboro, NJ, pp. 405-410, March 1994.

[Ortmanns & Ney 95] S. Ortmanns, H. Ney: Experimental Analysis of the Search Space for 20000-Word Speech Recognition. Proc. Fourth European Conference on Speech Communication and Technology, Madrid, pp. 901-904, September 1995.

[Ortmanns et al. 96] S. Ortmanns, H. Ney, A. Eiden: Language-Model Look-Ahead for Large Vocabulary Speech Recognition. Proc. Int. Conf. on Spoken Language Processing, Philadelphia, PA, October 1996.

[Renals & Hochberg 95] S. Renals, M. Hochberg: Efficient Search Using Posterior Phone Probability Estimates, Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing, Detroit, MI, Vol. 1, pp. 596-599, May 1995.

[Steinbiss et al. 94] V. Steinbiss, B.-H. Tran, H. Ney: Improvements in Beam Search. Proc. Int. Conf. on Spoken Language Processing, Yokohama, Japan, pp. 2143-2146, September 1994.