Hierarchical Search For LVCSR   Search

Extending the Baseline
One-Pass Algorithm Toward
New Dimensions

# DYNAMIC PROGRAMMING SEARCH *for Continuous Speech Recognition*

earch strategies based on dynamic programming (DP) are currently being used successfully for a large number of speech recognition tasks, ranging from digit string recognition through medium-size vocabulary recognition using heavily constrained grammars to large-vocabulary continuous speech recognition (LVCSR) with virtually unconstrained speech input.

Several variants of DP search were already known in the early days of automatic speech recognition [24], [37], [65], [66], [75], [76], [77]. Over the past three decades, these and related DP strategies have turned out to be surprisingly successful in handling vocabularies of 20k or more words. Nevertheless, until recently, among the experts, it was a highly controversial issue whether high-perplexity LVCSR could be handled by DP.

The skepticism seems to have been concerned mainly with the following issues, which we will address especially in this article:

▲ The extension from a 10-digit vocabulary to a 20k-word vocabulary would blow up the search space dramatically. Could this huge search space be handled by DP in an efficient way?

▲ In particular, each variant of DP search in speech recognition is more or less "notorious" for its operations at the 10-ms frame level. How could this

© Lorena Rodgers
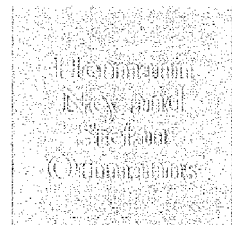
low-level acoustic search interact efficiently with the high-level knowledge sources in the recognition system such as the pronunciation lexicon and the language model (LM)? In addition, in order to narrow down the search, an early integration of these knowledge sources might be mandatory.

▲ DP typically computes only the single best sentence. But in many recognition systems, it is desirable for various reasons to produce alternative sentences or a word graph. Could the conventional DP strategy be extended to generate a word graph rather than only the single best sentence?

Where do we stand in speech recognition in comparison with 20 years ago? At that time, widely held opinions were quite different, with respect to both acoustic-phonetic modeling and to search. A number of experts predicted that considerable progress could be made by getting rid of "primitive techniques" like statistical pattern recognition and beam search. However, the experience gained over the last two decades has shown that the judgement passed by Klatt on the principles of the DRAGON and HARPY systems developed in 1976 is now more true than ever before [26, pp. 261]:

"...the application of simple structured models to speech recognition. It might seem to someone versed in the intricacies of phonology and the acoustic-phonetic characteristics of speech that a search of a graph of expected acoustic segments is a naïve and foolish technique to use to decode a sentence. In fact, such a graph and search strategy (and probably a number of other simple models) can be constructed and made to work very well indeed if the proper acoustic-phonetic details are embodied in the structure."

By extending Klatt's statement to include the language model, we obtain the topic of this article. Table 1 summarizes the definitions of some frequently used terms in the context of the search process in speech recognition. In this article, we will attempt to give a unifying view of the dynamic programming approach to the search problem. For a discussion of other search strategies and related topics, see a companion article appearing in this issue [16].

The organization of the article is as follows. In "System Architecture," we will review the search problem from the statistical point-of-view and show how the search space results from the acoustic and language models required by the statistical approach. "One-Pass DP Search Using a Linear Lexicon" presents a baseline algorithm that will then be extended to handle a prefix tree organization of the pronunciation lexicon in "One-Pass DP Search Using a Tree Lexicon." In "Refinements and Implementation Issues," we will discuss the practical implementation of the search strategy and related issues such as the details of the pruning operations and the language model look-ahead. In "One-Pass DP Search for Word Graph Construction," we will extend the one-pass strategy from the single best sentence to a word graph in order to gener-

Table 1. Definitions and Explanations of Frequently Used Terms.

Decoder: In an analogy with the terminology of finite-state methods for decoding [20] in information theory, the search algorithm in speech recognition is often referred to as a decoding algorithm.

Integrated search: We call a search strategy integrated if *all* available knowledge sources, e.g., acoustic-phonetic models, the constraints of the pronunciation lexicon, and the language model, are exploited in the search process at the same time; typically this concept is implemented in a one-pass strategy.

Time-synchronous: A search strategy is called time-synchronous if the search hypotheses are formed in a time-synchronous fashion over the sequence of acoustic vectors. Typically, the time-synchronous concept goes hand-in-hand with the one-pass search strategy. A^* search or stack decoding is an example of a search strategy that is *not* necessarily time-synchronous.

One-pass vs. multi-pass: We call a search a one-pass strategy if there is one single pass over the input sentence, as opposed to a multi-pass or multi-level concept. The one-pass search strategy is virtually always based on dynamic programming.

Word-conditioned vs. time-conditioned: These terms refer to the way in which the search space, especially in the context of dynamic programming, is structured. In a word-conditioned search, each search hypothesis is conditioned on the predecessor word. This implies that the optimization over the unknown end time of the predecessor word, i.e., the word boundary between the predecessor word and the word under consideration, is already carried out in an early phase of the search. Therefore, this method is different from a time-conditioned search, where, for each search hypothesis, the dependence on the end time of the predecessor word is explicitly retained and the optimization over the unknown word boundaries is performed as a final step of the search.

Single best vs. word graph: The attribute "single best" is used to denote a search concept that determines the single most likely word sequence. The alternatives are, among others, *n*-best concepts and word graph methods. The idea of a word graph here is to organize the high-ranking sentence hypotheses in the form of a graph whose edges represent the hypothesized single words. Sometimes, the term "word lattice" is used synonymously. However, in this article, by the term "word graph," we imply that gaps or overlaps between word hypotheses are not allowed.

Linear vs. tree lexicon: For a small-vocabulary task, it is sufficient to have a separate representation of each vocabulary word in terms of phonemes or HMM states (HMM = hidden Markov model). In nearly all cases, this is just a linear sequence of phonemes or HMM states. Therefore, this approach is referred to as linear lexicon. For a large vocabulary, however, it is typically very useful to organize the pronunciation lexicon as a tree, whose arcs are the phonemes.

ate sentence alternatives. Finally, we will present experimental results for the search algorithms on a 64k-word speech recognition task.
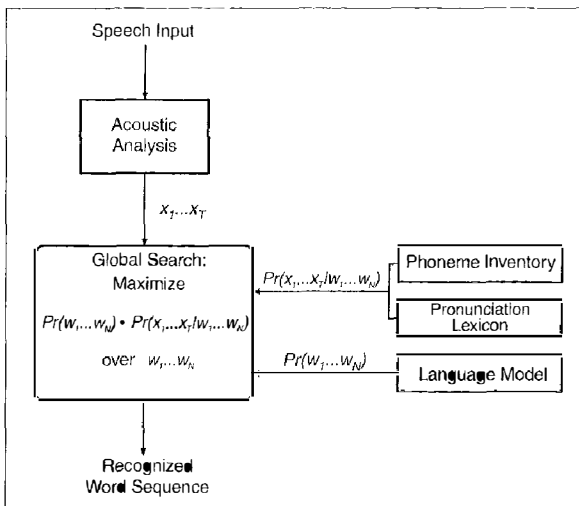
System Architecture

Why Is CSR Hard?

The ultimate long-term goal of automatic speech recognition is to build a system or machine that can "hear" in the sense that, for a spoken utterance, it converts the acoustic signal into the sequence of written words. The major problems for unrestricted, continuous speech recognition can be summarized as follows:

▲ In the acoustic signal, there is no clear indication or no indication at all of the boundaries between words or phonemes. Thus, not only the spoken words, but also the phoneme boundaries and the word boundaries are unknown.

▲ There is a large variation in the speaking rates in continuous speech.

▲ The words and especially the word endings are pronounced less carefully in fluent speech than in an isolated speaking mode.

▲ There is a great deal of inter- as well as intra-speaker variability, caused by a number of factors such as sex, physiological, and psychological conditions.

▲ The quality of the speech signal may be affected by environmental noise or the transfer function of the transmission system, e.g., microphone and telephone.

▲ For unrestricted natural-language speech input, the task-inherent syntactic-semantic constraints of the language should be exploited by the recognition system, in a way similar to human-to-human communication.

Bayes Decision Rule

Every approach to automatic speech recognition is faced with the problem of making decisions in the presence of ambiguity and context and of modeling the interdependence of these decisions at various levels. If it were possible to recognize phonemes or words with a very high reliability, it would not be necessary to rely heavily on delayed decision techniques, error correcting techniques, and statistical methods. Considering the experience gained over the last 30 years, we do not expect that this problem of reliable and virtually error-free phoneme or word recognition without using high-level knowledge will be possible, especially for LVCSR. As a consequence, the recognition system has to deal with a large number of hypotheses about phonemes, words and sentences, and ideally has to take into account the "high-level constraints" as given by syntax, semantics and pragmatics. Given this state of affairs, statistical decision theory tells us how to minimize the probability of recognition errors [7]:

Maximize the posterior probability $Pr(w_1...w_N|x_1...x_T)$, i.e. determine the sequence of words $w_1...w_n...w_N$ of unknown length N, which has most probably caused the observed sequence of acoustic vectors $x_1...x_t...x_T$ over time $t = 1...T$, which are derived from the speech signal in the preprocessing step of acoustic analysis.

▲ By applying Bayes theorem on conditional probabilities, the problem can be written in the following form: Determine the sequence of words $w_1...w_n...w_N$ which maximizes

$$Pr(w_1...w_n...w_N) \cdot Pr(x_1...x_t...x_T|w_1...w_n...w_N).$$

This so-called Bayes decision rule is illustrated in Fig. 1. It requires two types of probability distributions, which we refer to as stochastic knowledge sources, along with a search strategy:

▲ The language model, i.e., $Pr(w_1...w_N)$, is independent of the acoustic observations and is meant to incorporate (probabilistic) restrictions on how to concatenate words of the vocabulary to form whole sentences. These restrictions result from the syntactic, semantic, and pragmatic constraints of the recognition task, and may be modeled by probalistic or nonprobalistic (yes/no) methods. In large-vocabulary recognition tasks, the language model probabilities are typically approximated by bigram or trigram models:

$$Pr(w_n|w_1...w_{n-1}) = p(w_n|w_{n-1})$$
$$Pr(w_n|w_1...w_{n-1}) = p(w_n|w_{n-2}, w_{n-1}).$$

▲ The acoustic-phonetic model, i.e., $Pr(x_1...x_T|w_1...w_N)$, is the conditional probability of observing the acoustic vectors $x_1...x_T$ when the speaker utters the words $w_1...w_N$. Like the language model probabilities, these probabilities are estimated during the training phase of the recognition system. For a large-vocabulary system, typically, there is a set of basic recognition units that are smaller than whole words. Examples of these so-called subword units are phonemes, demisyllables, or syllables. The word models are then obtained by concatenating the subword models according to the phonetic transcription of the words in a pronunciation lexicon or dictionary. In most systems, these subword units are modeled by hid-

▲ 1. Bayes decision rule for speech recognition.

den Markov models (HMMs). HMMs are stochastic finite-state automata (or stochastic regular grammars) that consist of a Markov chain of acoustic states, modeling the temporal structure of speech, and a probabilistic function for each of the states, modeling the emission and observation of acoustic vectors [7], [9], [25], [35], [62]. In the experiments reported in this article, the phoneme models have a structure that is depicted in Fig. 2 along with the resulting search space. The phoneme X consists of three parts (X_1, X_2, X_3), resulting in a linear arrangement of six states. Words are obtained by concatenating the HMM phoneme units according to the baseline phonetic transcription as it can be found in a pronunciation dictionary. Usually, for a given state s' in a word model w, we have a transition probability $p(s|s', w)$ for going to state s, and an emission probability (density) $p(x_t|s', w)$ for observing vector x_t. For the following, it is sufficient to consider only the product of the emission and transition probabilities:

$$p(x_t, s|s'; w) = p(s|s'; w) \cdot p(x_t|s; w),$$

which is the conditional probability that, given the state s' in word w, the acoustic vector x_t is observed and the state s is reached.

Specification of the Search Problem

The decision on the spoken words must be made by an optimization procedure that combines information from several knowledge sources: the language model, the acoustic-phonetic models of single phonemes, and the pronunciation lexicon. The optimization procedure is usually referred to as search in a state space defined by the knowledge sources.

For a hypothesized word sequence $w_1^N = w_1 \dots w_N$, we imagine a super HMM that is obtained by concatenating the corresponding phoneme HMMs using a pronunciation lexicon (see Fig. 2). Note that by this process, we end up with a large number of copies for each phoneme and that these copies must be kept separate during the search process to satisfy the constraints of the pronunciation lexicon. At phoneme and word boundaries, we must allow for transitions that link the terminal states of any predecessor HMM to the initial states of any successor HMM. In such a way, we can compute the joint probability of observing the sequence $x_1^T = x_1 \dots x_T$ of acoustic input vectors and the state sequence $s_1^T = s_1 \dots s_T$ through this super HMM:

$$Pr(x_1^T, s_1^T|w_1^N) = \prod_{t=1}^{T} p(x_t, s_t|s_{t-1}, w_1^N),$$
$$= \prod_{t=1}^{T} [p(s_t|s_{t-1}, w_1^N) \cdot p(x_t|s_t)],$$

where $p(x_t, s_t|s_{t-1}, w_1^N)$ denotes the product of the transition and emission probabilities for the super HMM w_1^N. The decomposition has been formulated in such a way

Instead of summing over all paths, we consider only the most probable path.

that we can distinguish between two components of the approach:

▲ The reference models with the emission probability distributions $p(x_t|s)$ for the acoustic state s, e.g., after tying the emission distributions using decision trees [80] or some other method. Note that, for the emission probability $p(x_t|s)$, we stretch the notation a little bit and do not necessarily distinguish between the state s in a phoneme or word model and its associated generic emission probability distribution.

▲ The transition probabilities $p(s_t|s_{t-1}, w_1^N)$ depending on a word sequence hypothesis w_1^N; this implies a huge finite network of states (super HMM) that must be considered for each word sequence hypothesis w_1^N.

Denoting the language model (LM) probability by $Pr(w_1^N)$, the Bayes decision rule results in the following optimization problem:

$$\hat{w}_1^N = \underset{w_1^N}{\arg\max} \left\{ Pr(w_1^N) \cdot \sum_{s_1^T} Pr(x_1^T, s_1^T|w_1^N) \right\}$$
$$= \underset{w_1^N}{\arg\max} \left\{ Pr(w_1^N) \cdot \max_{s_1^T} Pr(x_1^T, s_1^T|w_1^N) \right\}.$$

Here, we have made use of the so-called maximum approximation, which is also referred to as Viterbi approximation [25]. Instead of summing over all paths, we consider only the most probable path. Note that for the maximum approximation to work, we need only the assumption that the resulting optimal word sequences are the same, not necessarily that the maximum provides a good approximation to the sum.

In this maximum approximation, the search space can be described as a huge network through which the best
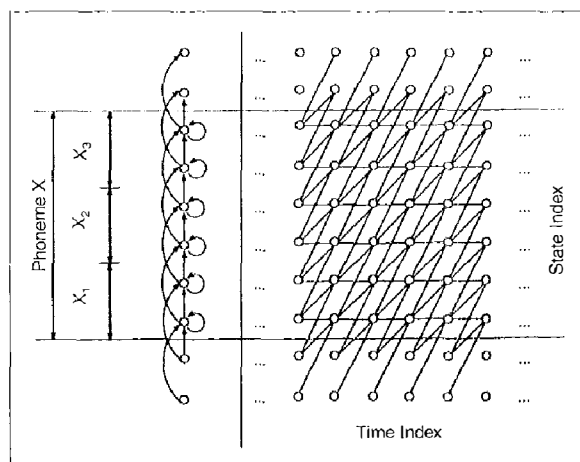
▲ 2. Structure of a phoneme model and search space.
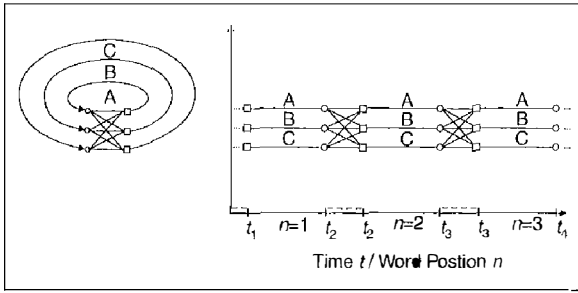
▲ 3. *Illustration of the search problem for a three-word vocabulary (A,B,C).*

time alignment path has to be found. The search has to be performed at two levels: at the state level (s_1^T) and at the word level (w_1^N). As we will see, as a result of the maximum approximation, it will be possible to *recombine* hypotheses efficiently at both levels by DP. Thus, the combinatorial explosion of the number of search hypotheses can be limited, which is one of the most important characteristics of DP. At the same time, the search hypotheses are constructed and evaluated in a strictly left-to-right time-synchronous fashion. This characteristic property allows an efficient pruning strategy to eliminate unlikely search hypotheses, which is usually referred to as beam search.

One-Pass DP Search Using a Linear Lexicon

Definition of the Search Space

In this section, for a linear lexicon, we describe the one-pass algorithm that forms the baseline for all search strategies described in this article. Originally the one-pass algorithm had been designed for small-vocabulary recognition tasks like digit string recognition [13], [41], [42], [77]. Over the last 30 years, however, these algorithms and their extensions have turned out to be surprisingly successful in handling vocabularies of 20,000 or more words.
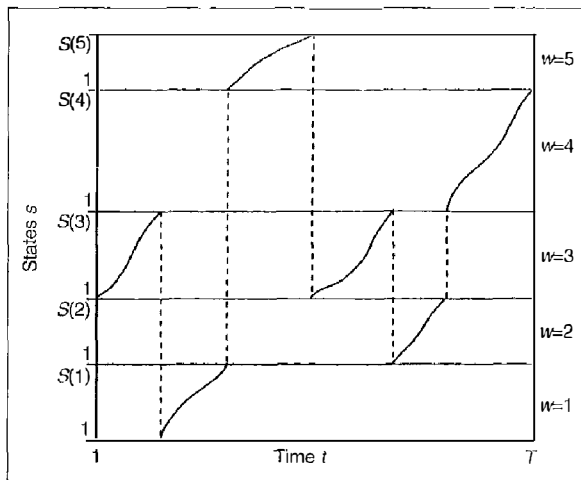
▲ 4. *Example of a time alignment path.*

The term "linear lexicon" denotes the fact that the words are kept strictly separate in the search process. Unlike a tree lexicon, there is no sharing between the words as far as the search hypotheses are concerned. For a three-word vocabulary, the search space is illustrated in Fig. 3. There are two types of transitions, namely the acoustic transitions representing the probabilities of the acoustic word models (A,B,C in Fig. 3), and the language transitions representing the language model probabilities. In Fig. 3, a bigram language model is assumed. For each possible word bigram (v,w), there is a LM transition that is assigned the conditional bigram LM probability $p(w|v)$, and that links the end of predecessor v to the beginning of word w. For recognition, as shown in Fig. 3, we unfold the finite-state machine along the time axis of the spoken utterance. For the sake of simplicity, Fig. 3 does not cover the details of the acoustic models and shows the language model transitions at times t_2 and t_3, only. Both the acoustic transitions (as shown in Fig. 2) and the language transitions must be considered every 10-ms time frame. As a result, there is a huge number of possible state sequences, and all combinations of state and time must be considered systematically for recognition.

In the maximum approximation, the search problem can be specified as follows. We wish to assign each acoustic vector observed at time t to a (state,word) index pair. This mapping can be viewed as a time alignment path, which is a sequence of (state,word) index pairs (stretching notation):

$$(s_1, w_1), \dots, (s_t, w_t), \dots, (s_T, w_T).$$

An example of such a time alignment path in connected word recognition is depicted in Fig. 4. For such paths, there are obvious continuity constraints or transition rules as shown in Fig. 5. Since the word models are obtained by concatenating phoneme models, the transition rules in the word interior (Fig. 5, top) are those of the used HMMs as shown in Fig. 2. At word boundaries (Fig. 5, bottom), we have to allow for transitions that link the terminal state S_v of any predecessor word v to the beginning states $s = 1$ and $s = 2$ of any word w. The dynamic programming search to be presented will allow us to compute the probabilities (stretching notation)

$$Pr(w_1 \dots w_t) \cdot Pr(x_1 \dots x_t ; s_1 \dots s_t | w_1 \dots w_t)$$

in a left-to-right fashion over time t and to carry out the optimization over the unknown word sequence at the same time. Note that the unknown word sequence and the unknown state sequence are determined simultaneously. Within the framework of the maximum approximation or Viterbi criterion, the dynamic programming algorithm presents a closed-form solution for handling the interdependence of nonlinear time alignment, word boundary detection, and word identification in continuous speech recognition [13], [34], [37], [40], [42], [65], [77].
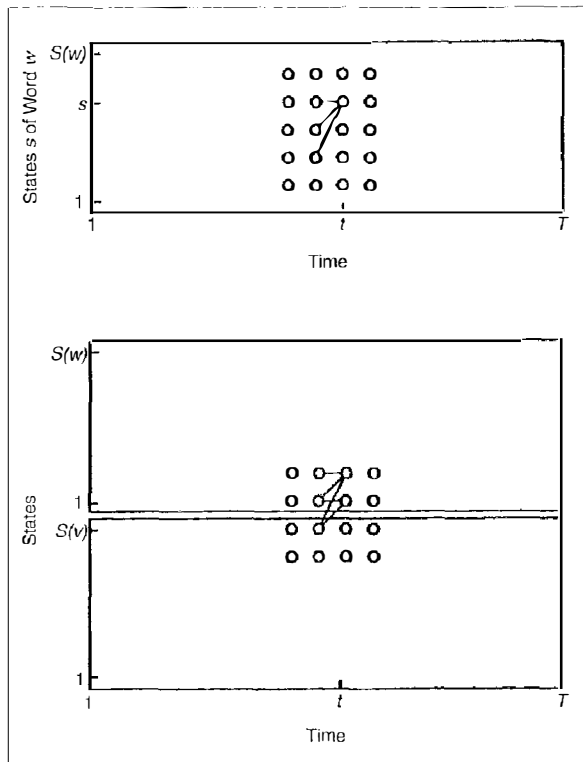
▲ 5. Path recombinations.

DP Recursion

The key concept of the dynamic programming strategy is based on the following two quantities:

$Q(t,s;w) :=$ score of the best path up to time t that ends

in state s of word w.

and

$B(t,s;w) :=$ start time of the best path up to time t

that ends in state s of word w.

Looking at the main memory sizes available today, we should add that the back pointer $B(t,s;w)$ is not absolutely needed for small-vocabulary tasks like digit string recognition. For vocabularies of 20,000 or more words, however, it is essential to reduce the storage requirements as much as possible.

As shown in Fig. 4, there are two types of transition rules for the path, namely rules in the word interior and at word boundaries. The concept of dynamic programming is to use these rules to decompose the path into two parts and formulate recurrence relations that can be solved by filling in tables, which, in this case, is the table $Q(t,s;w)$. In a more general setting of optimization problems, this concept is often referred to as *Bellman's principle of optimality* [10]. In the word interior, we have the recurrence equation:

$$Q(t,s;w) = \max_{s'}\{ p(x_t,s|s';w) \cdot Q(t-1,s';w) \}$$

$$B(t,s;w) = B(t-1,s_{\max}(t,s;w);w),$$

where $s_{\max}(t,s;w)$ is the optimum predecessor state for the hypothesis $(t,s;w)$. The back pointers $B(t,s;w)$ are propagated simply according to the DP decision as shown in Fig. 6 and report the start time for each word end hypothesis. When encountering a potential word boundary, we must perform the recombination over the predecessor words and, therefore define:

$$H(w;t) := \max_v \{ p(w|v) \cdot Q(t,S_v;v) \},$$

where $p(w|v)$ is the conditional LM probability of word bigram (v,w). The symbol S_v denotes the terminal state of word v. To allow for successor words to be started, we introduce a special state $s = 0$ and pass on both the score and the time index:

$$Q(t-1,s=0;w) = H(w;t-1)$$

$$B(t-1,s=0;w) = t-1.$$

This equation assumes that first the normal states $s = 1,...,S_w$ are evaluated for each word w before the start-up states $s = 0$ are evaluated. The same time index t is used intentionally, because the language model does not "absorb" an acoustic vector. Note that the scores $Q(t,s;w)$ capture both the acoustic observation dependent probabilities resulting from the HMM and the language model probabilities.

The operations to be performed are summarized in Table 2. The sequence of acoustic vectors extracted from the input speech signal is processed strictly from left to right. According to the DP equations, two levels are distinguished in Table 2: the acoustic level at which the word internal recombinations are performed and the word pair level at which the bigram LM recombinations are performed. The search procedure works with a
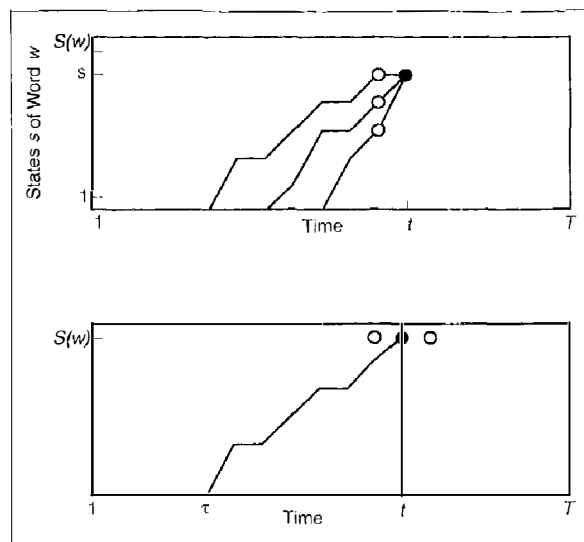
▲ 6. Illustration of back pointers.
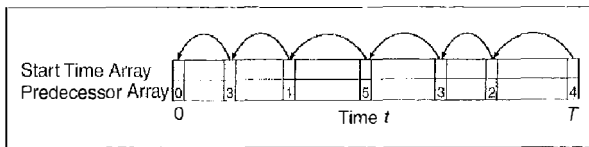
▲ 7. Illustration of traceback arrays.

time-synchronous breadth-first strategy, i.e., all hypotheses for word sequences are extended in parallel for each incoming acoustic vector. To reduce the storage requirements, it is suitable to introduce a traceback array in addition to the back pointers. For each time frame, the traceback array is used to record the decision about the best word end hypothesis and its start time. Using the traceback array, the recognized word sequence can be recovered efficiently by a few table look-ups into the traceback arrays at the end of the utterance as shown in Fig. 7.

Beam Search

Since, for a fixed time frame, all (word,state)-hypotheses cover the same portion of the input, their scores can be directly compared. This enables the system to avoid an exhaustive search, and to perform a data-driven search instead, i.e., to focus the search on those hypotheses that are most likely to result in the best state sequence [37]. Every 10-ms frame, the score of the best hypothesis is determined, then all hypotheses whose scores fall short of this optimal score by more than a fixed factor are pruned, i.e., are removed from further consideration. The experimental tests indicate that for this type of beam search, depending on the acoustic input and the language model constraints, only a small fraction of the overall number of possible (word,state)-hypotheses have to be processed for every 10 ms of the input speech, while at the same

time, the number of recognition errors is virtually not increased. This beam search strategy will be considered in full detail later in the context of a tree organization of the pronunciation lexicon. In addition, to fully exploit the computational advantages of this beam search strategy, a dynamic construction of the active search space is suitable as we will also discuss later. This one-pass dynamic programming algorithm in combination with beam search forms the foundation of the search component in many successful systems for both small-vocabulary and large-vocabulary speech recognition [1],[5], [14], [17], [21], [23], [30], [33], [34], [39], [46], [55], [72], [79].

One-Pass DP Search Using a Tree Lexicon

Definition of the Search Space

When applying the algorithm presented to large-vocabulary recognition, say a 20,000-word task, it seems natural and very desirable, for efficiency reasons, to organize the pronunciation lexicon in the form of a prefix tree, in which each arc represents a phoneme model, be it context dependent or independent [22], [46], [55]. A part of such a lexical pronunciation tree is shown in Fig. 8. This idea of using a tree representation was already suggested in the '70s in the CASPERS system [28] and in the lexical access from spectra (LAFS) system [27]. However, when using such a lexical tree in the framework of a language model, e.g., a bigram model, and dynamic programming, there are DP-specific technical details that must be taken into account and require a suitable structuring of the search space [22], [46]. Next we will present the search algorithm for such a context in full detail.

When using a bigram LM in connection with such a tree representation of the pronunciation lexicon, we face the problem that the identity of the hypothesized word w is known only when a leaf of the 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 will always know the predecessor word v when a word end hypothesis w is hypothesized. Fig. 9 illustrates this concept for a three-word vocabulary (A, B, C), where the lexical tree is depicted in a simplified schematic form. To avoid any potential misconceptions, we would like to stress that Fig. 9 shows the conceptual search space, which is too big to be constructed as a whole. Instead, as we will show later, we will construct the active portions of this search space dynamically in combination with beam search. In the set-up of Fig. 9, we apply the bigram LM 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.

Table 2. One-Pass DP Algorithm Using a Linear Lexicon.
Proceed over time t from left to right
Acoustic Level: Process (word,state)-hypotheses
- Initialization: $Q(t-1, s = \bullet; w) = H(w; t-1)$ $B(t-1, s = 0; w) = t-1$
- Time alignment: $Q(t, s; w)$ using DP   - Propagate back pointers $B(t, s; w)$
- Prune unlikely hypotheses   - Purge bookkeeping lists
Word Pair Level: Process word end hypotheses
For each word w do
$H(w; t) = \arg\max_v \{p(w|v)\, Q(t, S_w; w)\}$ $v_0(w; t) = \arg\max_v \{p(w|v)\, Q(t, S_w; w)\}$
- Store best predecessor $v_0 := v_0(w; t)$ - Store best boundary $\tau_0 := B(t, S_{v_0}; v_0)$

In the recognition process, in addition to the spoken words, we must account for possible pauses between the spoken words. To handle these so-called intraphrase pauses, we have a special HMM silence model and add a separate copy of this model (Sil) to each tree. Furthermore, for the possible pause at the sentence beginning, we have a separate copy of the lexical tree for the first word in the sentence; this tree copy is given silence as its 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. 9, 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 HMMs. At word boundaries, we have the thin and the dashed lines, which represent the bigram LM recombinations. Like the acoustic recombinations, they, too, are performed every 10-ms time frame. The dashed lines are related to recombinations for intraphrase silence copies. To start up a new word hypothesis, we must incorporate the bigram probability into the score and determine the best predecessor word. This best score is then propagated into the root of the associated lexical tree, which is represented by the symbol ⊔. The symbol ○ denotes a word end.

DP Recursion

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 [45]:
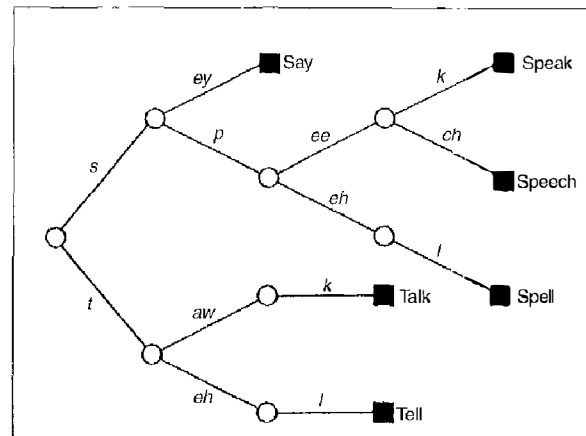
$Q_v(t,s) :=$ score of the best partial path that ends at time t in state s of the lexical tree for predecessor v.

$B_v(t,s) :=$ start time of the best partial path that ends at time t in state s of the lexical tree for predecessor v.
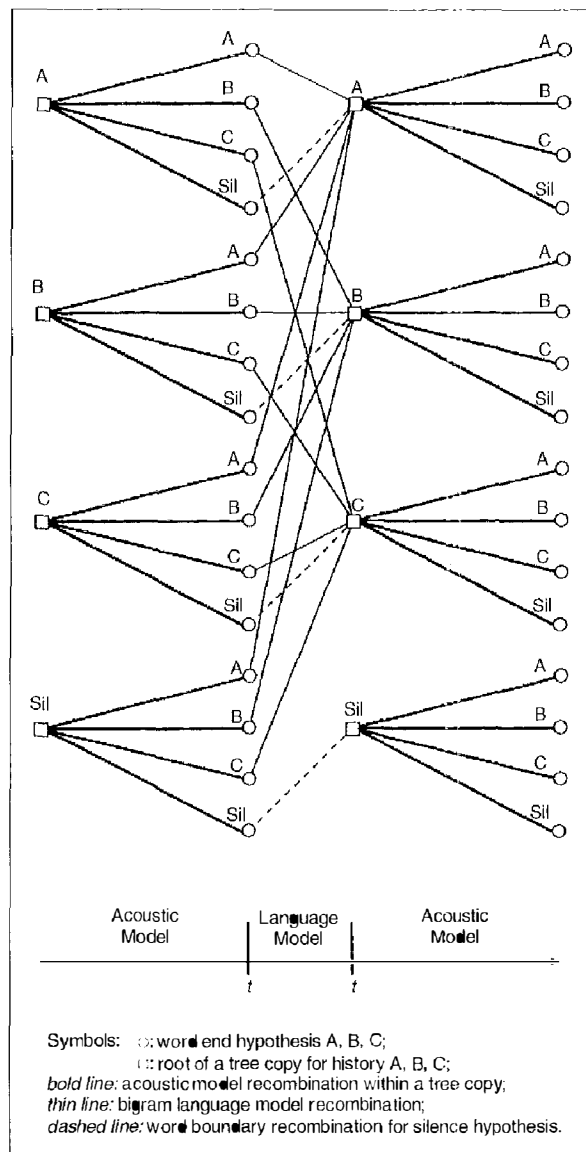
In other words, $B_v(t,s)$ is the back pointer that points back to the start time of the lexical tree copy for predecessor word v. This back pointer is needed because the definition of the score $Q_v(t,s)$ implies that the optimization over the unknown start time of the lexical tree copy for predecessor word v has been carried out. Both quantities are evaluated using the dynamic programming recursion for $Q_v(t,s)$:

$$Q_v(t,s) = \max_{s'}\{p(x_t,s|s') \cdot Q_v(t-1,s')\}$$

$$B_v(t,s) = B_v(t-1,s_r^{\max}(t,s))$$

▲ 8. Tree-organized pronunciation lexicon.

Symbols: ○: word end hypothesis A, B, C;
           ⊔: root of a tree copy for history A, B, C;
bold line: acoustic model recombination within a tree copy;
thin line: bigram language model recombination;
dashed line: word boundary recombination for silence hypothesis.

▲ 9. Bigram LM recombination and intraphrase silence (Sil) handling for a tree lexicon (three-word vocabulary: A,B,C).

where $s_p^{max}(t,s)$ is the optimum predecessor state for the hypothesis (t,s) and predecessor word v. As before, 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 $s = 0$, 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. As in the case of a linear lexicon, we define:

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

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 in case they do not exist yet, we pass on the score and the time index *before* processing the hypotheses for time frame t:

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

$$B_v(t-1,s=0) = t-1.$$

The details of the algorithm are summarized in Table 3.

Extension to Trigram Language Models

So far, we have considered the one-pass search approach only in the context of a bigram language model. To extend the tree search method from a bigram to a trigram LM, we take into account that for a trigram, the language model probabilities are conditioned on the previous two words rather than on one predecessor word in the case of

Table 3. One-Pass DP Algorithm Using a Tree Lexicon.
Proceed over time t from left to right
Acoustic Level: Process (tree,state)-hypotheses
- 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 end hypotheses
For each word do
$H(w;t) = \arg\max_v \{ p(w|v) Q_v(t,S_w) \}$ $v_0(w;t) = \arg\max_v \{ p(w|v) Q_v(t,S_w) \}$
- Store best predecessor $v_0 := v_0(w;t)$ - Store best boundary $\tau_0 := B_{v_0}(t,S_w)$

a bigram LM [45], [50], [57]. Therefore, the incorporation of a trigram LM into the tree search method requires a restructuring of the search space organization. Fig. 10 illustrates the search space using a trigram model. For each two-word history (u,v), we introduce a separate copy of the lexical tree; in Fig. 10, the root of each tree copy is labeled with its two-word history. As in the case of a bigram LM, the structure of the search space is defined in such a way that, in the search network, the probabilities or costs of each edge depend *only* on the edge itself (along with its start and end vertex) and nothing else. This prop-
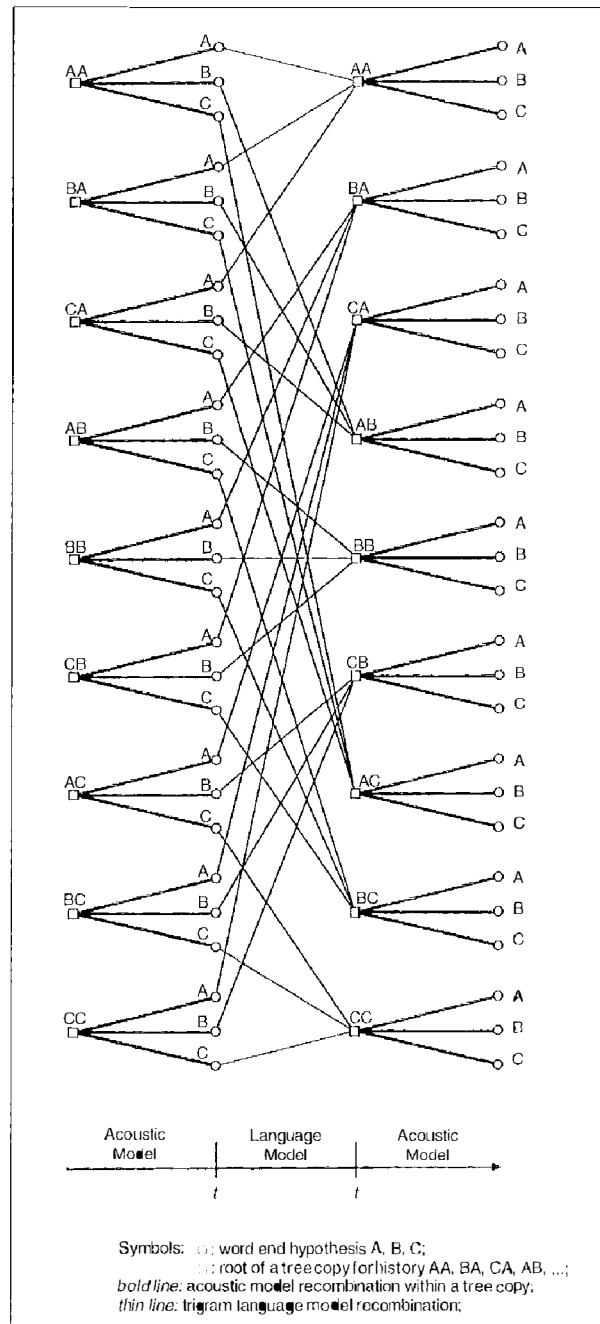

10. Trigram LM recombination for a tree lexicon (three-word vocabulary: A,B,C).

erty of the search network allows us to directly apply the principle of dynamic programming. Note that in comparison with a bigram LM organized search space, the size of the *potential* search space is increased drastically by an additional factor, which is the vocabulary size. Hence, in order to keep the search effort manageable, an efficient pruning strategy as described before is even more crucial for the case of a trigram language model.

For simplicity, we have omitted the silence copies in Fig. 10. To allow for intraphrase silence, we use the same concept as for the bigram language model [22], [46]. For the trigram language model recombination, we need the identity of the two non-silence predecessor words, and therefore, a separate copy of the silence model is required for each pair of non-silence predecessor words.

Refinements and Implementation Issues

To obtain an estimate of the overall number of state hypotheses, we consider a typical task such as the 20k-word NAB task [29]:
▲ 20,000-word vocabulary with a 65,000-arc tree for the pronunciation lexicon
▲ Bigram language model
▲ Six states per HMM arc
For this task, we obtain the following size of the potential search space:

20k trees · 65k arcs/tree · 6 HMM states/arc = $7.8 \cdot 10^9$ HMM states.

Therefore, in full DP search, there are this many HMM states for which the DP recursions have to be evaluated every 10-ms time frame of the input signal. In contrast with this astronomic number, the experiments will show that, without loss in recognition accuracy, it is sufficient to evaluate only 10,000 and fewer state hypotheses on average per 10-ms time frame.

Pruning Refinements
Evidently, full search is prohibitive. As a result, we use the time synchronous beam search strategy, where, for every time frame, only the most promising hypotheses are retained. The pruning approach consists of three steps that are performed every 10-ms time frame [73]:
▲ *Acoustic pruning* is used to retain only hypotheses with a score close to the best state hypothesis for further consideration. 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}.

The basic idea of the language model look-ahead is to incorporate the language model probabilities as early as possible into the search process.

▲ *Language model pruning (or word end pruning)* is only applied to tree start-up hypotheses as follows. For word end hypotheses, the bigram LM 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 the fictitious state of the tree root 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 (M_{Sta}). If the number of active states is larger than M_{Sta}, only the best M_{Sta} hypotheses are retained while the other hypotheses are removed. This pruning method is called histogram pruning because we use a histogram of the scores of the active states [73].

Language Model Look-Ahead
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 anticipating the LM probabilities as a function of the nodes of the lexical tree so that each node corresponds to the maximum LM probability over all words that can be reached via this specific node. Using the bigram LM conditional probability $p(w|v)$, the anticipated LM probability $\pi_v(s)$ for state s and predecessor word v is defined as:

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

where $W(s)$ is the set of words that can be reached from tree state s. Strictly speaking, we should use the tree nodes (or arcs) rather than the states of the HMMs that are associated with each arc. However, each initial state of a phoneme arc can be identified with its associated tree node. The concept of anticipating the LM probabilities for each node of the lexical tree is illustrated in Fig. 11.

To keep the computational cost down in this forward-recombination scheme, it is important to have *direct access* to each new successor hypothesis.

To incorporate the anticipated LM probabilities into the three pruning operations of the search, we combine the anticipated LM probabilities with the score of the hypothesis $(t,s;v)$ and define a modified score $\tilde{Q}_v(t,s)$:

$$\tilde{Q}_v(t,s):=\pi_v(s)\cdot Q_v(t,s).$$

For the acoustic pruning, we compute the modified score of the best state hypothesis:

$$\tilde{Q}_{AC}(t):=\max_{(v,s)}\{\tilde{Q}_v(t,s)\}.$$

and prune a hypothesis $(t,s;v)$ if

$$\tilde{Q}_v(t,s)<f_{AC}\cdot\tilde{Q}_{AC}(t).$$

The same modified score is used for the histogram pruning. As the recognition experiments will show, the number of state hypotheses can be significantly reduced by this look-ahead.

When computing all entries of the table $\pi_v(s)$ beforehand, we have to keep a huge table in main memory. In the task under consideration, about $20,000 \cdot 65,000$ anticipated LM probabilities would have to be stored. Since the size of this table is prohibitive, we compute the anticipated LM probabilities for the tree copies on demand and cache these anticipated probabilities in a look-up table for a maximum of, say, 300 LM look-ahead trees. So, before computing the anticipated LM probabilities, we first check whether or not the probabilities of the required tree copy already exist in the look-up table. In addition, it is often sufficient to compute the anticipated LM probabilities only for the first, say four, arc generations of the lexical tree [57]. This LM look-ahead or similar variants are now used in many systems for large-vocabulary speech recognition [2], [3], [5], [6], [49], [50], [54], [63], [73].

Implementation
Although a full description of the implementation is out of the scope of this article, we will present some concepts and details.

Dynamic Search Space Construction
In this paragraph, a dynamic construction of the search is derived from the time-synchronous DP beam search by introducing a set representation of the active hypotheses. The basic difficulty with the search implementation is to perform the recombination of path hypotheses efficiently without explicitly constructing the overall search space. The naive implementation of the DP equations would require that each state hypothesis be processed or at least checked whether it is still active. Thus, there would be a computational overhead that is proportional to the number of state hypotheses in the overall search space.

For search using a linear lexicon, the dynamic construction of the search space was described in detail in [47]. To arrive at an efficient implementation for tree search, we use the following concepts:

▲ *Set representation of active hypotheses*: For each time frame t, we maintain sets of active hypotheses. For efficiency reasons, these active hypotheses are stored in static arrays whose maximum sizes are specified beforehand. These hypotheses are organized in a three-level hierarchy:

  ▲ At the highest level, we have the set of active trees or predecessor words v.
  ▲ For each tree, there is the set of active phoneme arcs. Due to our notational scheme, the arc dependence is not explicitly captured in the DP equations.
  ▲ For each tree v and for each arc, we have the set of active HMM states. Each state hypothesis consists of three parts, namely state index s, score $Q_v(t,s)$, and back pointer $B_v(t,s)$.

▲ *Forward DP recombination*: DP recombination occurs at three levels: word boundaries, phoneme boundaries and HMM states. To confine the computational effort to the active search space, we convert the DP recursions from the usual backward direction (as expressed by the equations) into a *forward* direction; using the active hypotheses, we construct dynamically the set of *successor* hypotheses. To keep the computational cost down in this forward-recombination scheme, it is important to have *direct access* to each new successor hypotheses.

This is achieved by an array-based representation of sets in combination with a stack of indirected pointers [38, pp. 289-290x], [47], [78, pp. 121-123]. In this way, there is no need to search through lists to find a hypothesis. This forward recombination is of varying importance at the
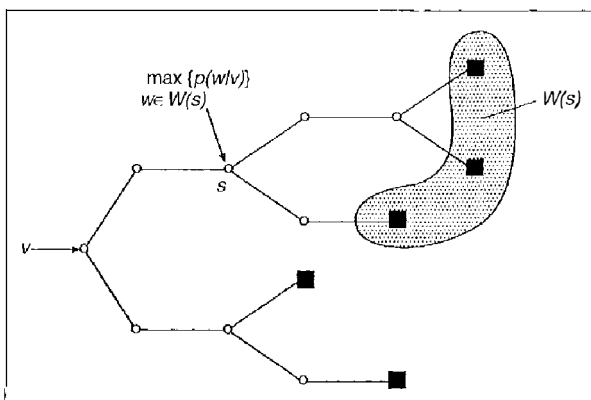
▲ *11. Anticipated LM probabilities for LM look-ahead.*

three levels. For the HMM state level, there is no real problem because *within* a phoneme are of the tree, there is a maximum number of six states. At the arc level, i.e., for recombination across phoneme boundaries, this is much more important because, for a fixed tree, there might be up to 65,000 active arc hypotheses.

There is one condition for which we cannot use the direct access approach, namely the LM recombination for a trigram LM. The problem is that there might be up to W^2 predecessor histories for a vocabulary of W words. Therefore, we replace the direct access method by a hashing approach [59]. The index for the hash table is computed by hashing a bijective function of the word pair index (u,v), e.g., $f(u,v) = W \cdot u + v$.

Traceback and Garbage Collection

For large-vocabulary recognition, it is essential to keep the storage costs as low as possible. To this purpose, in addition to back pointers, we use a special traceback array whose entries keep track of word end hypotheses. The concept of this traceback array is based on an extension of the traceback method presented in Fig. 7. For each word end hypothesis, we store the following pieces of information: word index, end time of the predecessor word, score, and back pointer, i.e., a pointer into the array itself for finding the predecessor word end hypothesis. The end time of the predecessor word is not really needed, but useful for diagnostic purposes. During the recognition process, many of the hypothesis entries in the traceback arrays will become obsolete because their path extensions die out over time due to both the recombination and the pruning of hypotheses. In order to remove these obsolete hypothesis entries from the traceback arrays, we apply a garbage collection or purging method as follows. Each entry of the traceback array is extended by an additional component called the "time stamp," as suggested by Steinbiss [71]. Using the back pointers $B_v(t,s)$ of the state hypotheses, we perform a traceback for each hypothesis and mark the traceback entries reached with the current time frame as time stamps. Hence, all traceback entries that have a time stamp differ-

The basic idea is to represent all these word sequences by a word graph in which each edge represents a word hypothesis.

ent from the current time frame can be re-used to store new hypotheses. Note that this garbage collection process is controlled using only the *state hypotheses and reachable traceback entries* so that the number of *dead* traceback entries does not matter. In principle, this garbage collection process can be performed every time frame, but to reduce the overhead, it is sufficient to perform it in regular time intervals, say every 50-th time frame.

Memory Requirements
By using the above methods, we obtain typical memory costs as shown in Table 4. For each of the various arrays used, we simply report the number of bytes required without going into all technical details. The three arrays of tree, arc, and state hypotheses store the corresponding hypotheses per time frame, and thus, we have a maximum of 20,000 tree hypotheses, 200,000 arc hypotheses (over all tree hypotheses), and a total of 600,000 state hypotheses (over all tree and arc hypotheses). For the recombination at the arc level, an auxiliary array is included in Table 4. As a result, the total storage cost for the 20,000-word task is about 14 Mbytes for the search procedure.

One-Pass DP Search for Word Graph Construction

The main idea of a word graph is to come up with word alternatives in regions of the speech signal, where the ambiguity about the actually spoken words is high. The expected advantage is that the acoustic recognition process is decoupled from the application of a complex language model and that this language model can be applied in a subsequent postprocessing step. Examples of long-span language models are cache-based language models [31], trigger-based language models [74] and long-range trigram language models [15] that can be viewed as stochastic lexicalized context-free grammars. The number of word alternatives should be adapted to the level of ambiguity in the acoustic recognition.

Word Graph Specification
In this section, we will formally specify the word graph generation problem and pave the way for the word graph algorithm. We start with the fundamental problem of word graph generation:

Hypothesizing a word w and its end time t, how can we find a limited number of "most

likely" predecessor words? This task is difficult since the start time of word w may very well depend on the predecessor word under consideration, which results in an interdependence of start times and predecessor words.

In view of the most successful one-pass beam search strategy, what we want to achieve conceptually is to keep track of word sequence hypotheses whose scores are very close to the locally optimal hypothesis, but that do not survive due to the recombination process.

The basic idea is to represent all these word sequences by a word graph in which each edge represents a word hypothesis. Each word sequence contained in the word graph should be close (in terms of scoring) to the single best sentence produced by the one-pass algorithm. In the one-pass algorithm for computing the single-best sentence, we have computed the hypotheses in a time-synchronous fashion and propagated the hypotheses from left to right over the time axis. We will use the same principle of time synchrony for the word graph generation. To this purpose, we introduce the following definitions:

$$h(w;\tau,t) := \max_{s_{\tau+1}^{t}} Pr(x_{\tau+1}^{t}, s_{\tau+1}^{t} \mid w)$$

= Conditional probability that word w produces the acoustic vectors $x_{\tau+1}^{t}$

$$G(w_1^n;t) := Pr(w_1^n) \cdot \max_{s_1^t} Pr(x_1^t, s_1^t \mid w_1^n)$$

= Joint probability of observing the acoustic vectors x_1^t and a word sequence w_1^n with end time t.

Using these definitions, we can isolate the probability contributions of a particular word hypothesis with respect to both the language model and the acoustic model (see Fig. 12). This decomposition can be visualized as follows:

$$\underbrace{x_1\cdots,\cdots,x_\tau}_{G(w_1^{n-1};\tau)} \underbrace{x_{\tau+1}\cdots,x_t}_{h(w_n;\tau,t)} \underbrace{x_{t+1},\cdots,x_T}_{\cdots}$$

From this decomposition, it is clear that the score $G(w_1^n;t)$ can be computed from the scores $G(w_1^{n-1};\tau)$ and $h(w;\tau,t)$ by optimizing over the unknown word boundary τ:

$$G(w_1^n;t) = \max_\tau \{Pr(w_n \mid w_1^{n-1}) \cdot G(w_1^{n-1};\tau) \cdot h(w_n;\tau,t)\}$$
$$= Pr(w_n \mid w_1^{n-1}) \cdot \max_\tau \{G(w_1^{n-1};\tau) \cdot h(w_n;\tau,t)\},$$

where we have used the *conditional* probability $Pr(w_n \mid w_1^{n-1})$ of the language model. To construct a word graph, we introduce a formal definition of the word boundary $\tau(w_1^n;t)$ between the word hypothesis w_1^n ending at time t and the predecessor sequence hypothesis w_1^{n-1}:

$$\tau(t;w_1^n) := \arg\max_\tau \{G(w_1^{n-1};\tau) \cdot h(w_n;\tau,t)\},$$

It should be emphasized that the language model probability does *not* affect the optimal word boundary
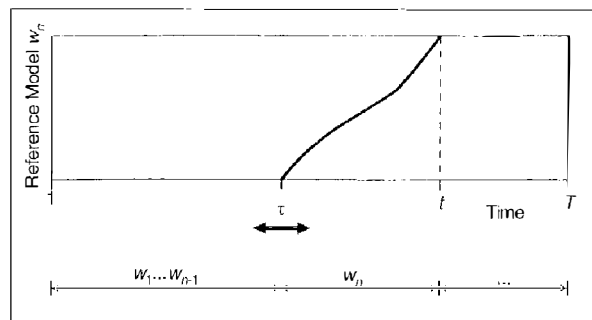
A *12. Illustration of path decomposition.*
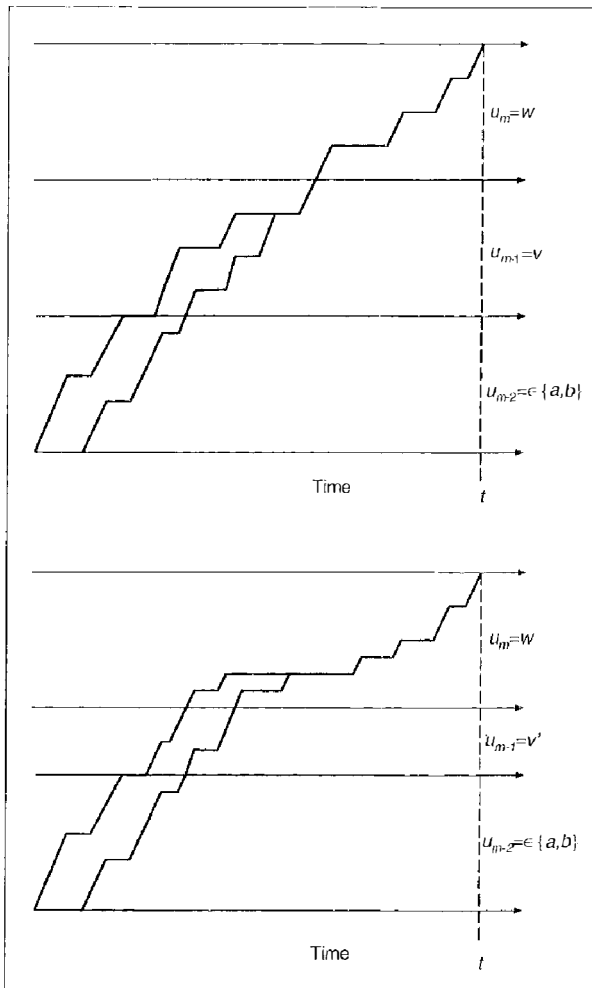
A *13. Illustration of the word pair approximation for two cases. Top, good example: the predecessor word $u_{m-1} := v$ of word $u_m := w$ is sufficiently long. Bottom, bad example: the predecessor word $u_{m-1} := v$ of word $u_m := w$ is too short.*

and is, therefore, omitted in the definition of the word boundary function $\tau(w_1^n;t)$. Thus far we have considered the most general case in two aspects: First, the word boundary function has not been constrained in any way. Second, the language model has not been constrained in any way. We will first narrow down the language model

to the widely used m-gram language models and come back to the word boundary function later.

Exploiting an m-gram language model $p(u_m | u_1^{m-1})$, we can recombine word sequence hypotheses at the phrase level if they do not differ in their final $(m-1)$ words. Therefore, it is sufficient to distinguish partial word sequence hypotheses w_1^n only by their final words $u_2^m := w_{n-m+2}^n$. The corresponding score is denoted by $H(u_2^m ; t)$, and is defined as the joint probability of generating the acoustic vectors $x_1 \ldots x_t$ and a word sequence with *ending* sequence u_2^m and *ending* time t:

$$H(u_2^m ; t) := \max_{w_1^n} \{ Pr(w_1^n) \cdot Pr(x_1^t | w_1^n) : w_{n-m+2}^n = u_2^m \},$$

where, as expressed by the notation, the final portion u_2^m of the word sequence w_1^n is not subjected to the maximization operation. Using the above definition, we can write the dynamic programming equation at the word level:

$$H(u_2^m ; t) = \max_{u_1} \hat{H}(u_1^m ; t)$$

$$\text{with} \quad \hat{H}(u_1^m ; t) := p(u_m | u_1^{m-1}) H(u_1^{m-1} ; \tau(t ; u_1^m))$$

$$h(u_m ; \tau(t ; u_1^m), t)$$

Here, we have used the function $\tau(t ; u_1^m)$ to denote the word boundary between u_{m-1} and u_m for the word sequence with final portion u_1^m and end time t. Note that we have included the language model to achieve a better pruning strategy. For the word boundary itself, we have to use the quantity $H(u_2^m ; t)$ rather than $G(w_1^N ; t)$:

$$\tau(t ; u_1^m) := \arg \max_\tau \{ H(u_1^{m-1} ; \tau) h(u_m ; \tau, t) \}.$$

Word Pair Approximation

So far this has been just a notational scheme for the word boundary function $\tau(t ; u_1^m)$. The crucial assumption now is that the dependence of the word boundary $\tau(t ; u_1^m)$ can be confined to the final word pair u_{m-1}^m. The justification is that the other words have virtually no effect on the position of the word boundary between words u_{m-1} and u_m [67]. This so-called word pair approximation is illustrated in Fig. 13. For a word hypothesis w and an end time hypothesis t, this figure shows the time alignment path for the word $w = u_m$ itself and its predecessor words u_{m-2}^{m-1} to illustrate the definition of the word boundary $\tau(t ; u_1^m)$. In general, this boundary, i.e., the start time of word w as given by time alignment, will depend on the immediate predecessor word u_{m-1}. The question of whether this dependence reaches beyond the immediate predecessor word is illustrated by showing a good (Fig. 13, top) and a bad (Fig. 13, bottom) example. For simplification, we have assumed that the reference models of the predecessor words $u_{m-2} = a$ and $u_{m-2} = b$ have the same length. From this figure, it is obvious that the assumption of the word pair approximation is satisfied if the predecessor word u_{m-1} is sufficiently long. All time alignment paths then are recombined before they reach the final state of the predecessor

word. In formulae, we express the word pair approximation by the equation:

$$\tau(t ; u_1^m) = const(u_1^{m-2}) \quad \text{or} \quad \tau(t ; u_1^m) = \tau(t ; u_{m-1}^m),$$

i.e., the word boundary function does *not* depend on u_1^{m-2}. Assuming the word pair approximation, we have the following algorithm for word graph generation:

▲ At every time frame t, we consider all word pairs $u_{m-1}^m = (v, w)$. Using a beam search strategy, we will limit ourselves to the most probable word pairs.

▲ For each triple $(t ; v, w)$, we have to keep track of:
  ▲ the word boundary $\tau(t ; v, w)$
  ▲ the word score $h(w ; \tau(t ; v, w), t)$

▲ At the end of the speech signal, the word graph is constructed by tracing back through the bookkeeping lists.

As long as only a bigram language model is used, the word pair approximation is still exact (assuming a conservatively large pruning threshold). An even further simplification is the single word approximation used in [70] to produce a list of n-best sentences.

Word Graph Generation Algorithm

The computation of the word boundary function $\tau(t ; v, w)$ has not been specified yet. In principle, it can be computed by using either the so-called two-level algorithm [65] or the one-pass algorithm described before, both of which compute only the best single word sequence. However, to apply beam search, it is more convenient to use the one-pass algorithm presented in the preceding section. Using the tree organization of the pronunciation lexicon, the hypotheses have been distinguished by the predecessor word anyway.

To extend the one-pass word algorithm from single-best sentence computation to word graph generation, we only have to combine the two equations for calculating the word boundary function $\tau(t ; v, w)$ and the word score $h(w ; \tau, t)$. The word boundaries are obtained using the back pointers at the word ends:

$$\tau(t ; v, w) = B_v(t, S_w).$$

For each predecessor word v along with word boundary $\tau = \tau(t ; v, w)$, the word scores are recovered using the equation:

$$h(w ; \tau, t) := \frac{Q_v(t, S_w)}{H(v ; \tau)},$$

where we obtain $H(w ; t)$ as usual:

$$H(w ; t) = \max_v \{ p(w | v) \cdot Q_v(t, S_w) \}.$$

The details of the algorithm are summarized in Table 5. The operations are organized in two levels: the acoustic level and the word pair level. At the end of the utterance, the word graph is constructed by tracing back through the bookkeeping lists. A third level, the phrase level, has

been included for the final recognition. Depending on whether the phrase-level recognition is carried out in a time-synchronous fashion or not, we can distinguish the following two strategies by using a trigram or higher m-gram language model:

▲ *Extended one-pass approach*: The word pair approximation serves only as a simplification in the one-pass strategy in order to avoid the large number of copies of the lexical tree as required by the language model.

▲ *Two-pass approach*: First, a word graph is constructed. Then, at the phrase level, the best sentence is computed using a more complex language model. An example of a language model that is difficult to handle in integrated search is a cache-based language model [31].

From the concepts developed so far, it should be obvious that there is only a gradual difference between these two strategies.

What has to be added to the single-best one-pass strategy, is the bookkeeping at the word level: rather than just the best surviving hypothesis, the algorithm must memorize all the word sequence hypotheses that are recombined into just one hypothesis to start up the next lexical tree (or word models). In the single-best method, only the surviving hypothesis (ν_0, τ_0) has to be kept track of.

Table 5. Extension of the One-Pass DP Algorithm from Single Best Sentence to Word Graph Generation.

Proceed over time from left to right

  Acoustic Level: Process (tree,state)-hypotheses

    - Initialization: $Q_\nu(t-1, s=0) = H(\nu; t-1)$
                  $B_\nu(t-1, s=0) = t-1$

    - Time alignment: $Q_\nu(t, s)$ using DP
    - Propagate back pointers $B_\nu(t, s)$

    - Prune unlikely hypotheses
    - Purge bookkeeping lists

  Word Pair Level: Process word end hypotheses

    "single best": for each word w do

      $H(w; t) = \arg\max_\nu \{ p(w|\nu) \, Q_\nu(t, S_w) \}$
      $\nu_0(w; t) = \arg\max_\nu \{ p(w|\nu) \, Q_\nu(t, S_w) \}$

      - Store best predecessor $\nu_0 := \nu_0(w; t)$
      - Store best boundary $\tau_0 := B_{\nu_0}(t, S_w)$

    "Word graph": for each word pair (ν, w) store

      - Word boundary $\tau(t; \nu, w) := B_\nu(t, S_w)$
      - Word score $\quad h(w; \tau, t) := Q_\nu(t, S_w) / H(\nu; \tau)$

  -Phrase Level Search (optional)

An example of a word graph for a three-word vocabulary A, B, C (including silence at the sentence beginning and end) is shown in Fig. 14. The edges stand for word hypotheses, where the circles along with the word name denote the word end. Note the following principal properties, which are a result of the word graph algorithm:

▲ There is a maximum for the number of incoming word edges in any node, namely the vocabulary size, which is the maximum number of possible predecessor words.

▲ There is no maximum for the number of outgoing word edges; this effect is due to the fact that, even for the same predecessor word, a word can have different end-time hypotheses.

There are two refinements of the word graph method which suggest themselves:

▲ For short words like articles and prepositions, the quality of the word pair approximation might be questionable, and word triples or higher word m-tuples might be used instead in these cases.

▲ Long words with identical ending portions may waste search effort and could be merged when forming word pairs in the word graph algorithm.

In both cases, the obvious remedy is to make the word copies dependent on a suitably defined history using the phonetic script of the predecessor words.

Another refinement is concerned with the way in which the word graph is pruned. What we have used so far could be called *forward only* pruning as opposed to *forward-backward* pruning, which is a little bit better, but does not allow strict online operation [32], [51, pp. 811], [68].

For the sake of clarity, we have not included the case of intraphrase silence in the presentation of the algorithm. The algorithm can be extended for this case.

Experimental Results

The search concepts presented in this article are used in a large number of systems. Of course, the technical details of the implementation may vary from case to case [1], [2], [3], [4], [6], [16], [18], [19], [21], [36], [39], [49], [50], [52], [69], [81]. The exact implementation of this article was used in a number of experiments [5], [48], [54], [55], [56], [57], [58], [59].

Here, we will review only some of these experiments. All recognition experiments were carried out on the ARPA North American Business (NAB'94) H1 development corpus [29]. The test set comprised 10 female and 10 male speakers resulting in 310 sentences with 7,387 spoken words. The recognition vocabulary used in the experiments comprised 64,000 words with an out-of-vocabulary rate of 0.5% on the test data. The training of the emission probability distributions was performed on WSJ 0 and WSJ 1 training corpora. In this task, 4,058 context dependent phoneme models were used, sharing 4,699 emission probability distributions [18]. For these experiments, we used a total of 270,000

Laplacian mixture densities with a single pooled vector of absolute deviations for each gender [43].

Search Space

In the first series of experiments, we studied the size of the search space in the beam search strategy. The language model was either a bigram or a trigram LM. The search effort was measured in terms of the average number of tree and state hypotheses *after* recombination and pruning. The acoustic pruning threshold was varied, whereas both the LM pruning threshold and the maximum number of state hypotheses were kept fixed.

The results are shown in Table 6. These tests were carried out using a unigram LM look-ahead. Looking at the search effort, we can see that the results are comparable for the bigram LM and the trigram LM. For both types of LM, there is a saturation effect for the word error rate; beyond 30,000 state hypotheses, the word error rate remains virtually constant. When replacing a trigram LM by a bigram LM, the average numbers of tree and state hypotheses are affected in different ways. Whereas the number of state hypotheses remains more or less unchanged, the number of tree hypotheses goes up, but only by a factor of two or less. Despite the potential maximum of $W^2 = 64,000^2 = 4.1 \cdot 10^9$ possible tree hypotheses for the trigram LM, the average number is only 200 or less.

LM Look-Ahead

The second series of recognition experiments is concerned with the effect of the LM look-ahead on the size of the search space and the word error rate. Table 7 shows the results of several recognition tests. As before, the table shows the size of the search space in terms of the average number of state and tree hypotheses and the word error rate. In an initial experiment, we performed three tests without any LM look-ahead, using three different values of the acoustic pruning threshold. For the recognition scores, as opposed to the LM look-ahead scores, we used a bigram LM in these tests. To achieve a word error rate of 13.9%, an average of 168,000 state hypotheses per time frame are needed. By using the unigram LM
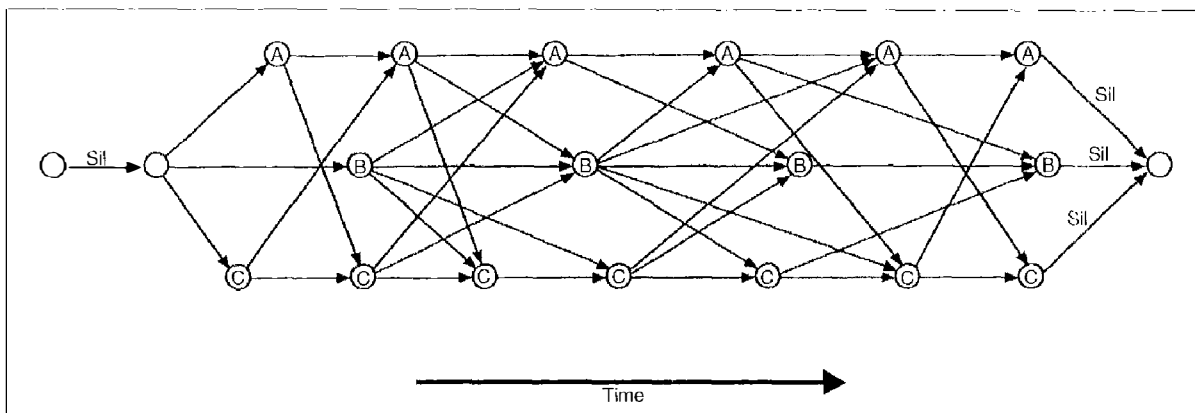
Table 6. Search Space and Word Error Rate (WER) for 64k-word NAB Task (Unigram LM Look-Ahead).

LM Type	Search Effort		WER
	Trees	States	%
Bigram	15	5,600	22.1
(PP=237)	24	10,800	16.0
	37	20,200	14.5
	51	33,700	13.9
	65	50,100	13.9
	99	116,500	13.8
Trigram	17	1,800	17.1
(PP=172)	29	3,900	14.0
	48	8,200	12.8
	73	15,800	12.1
	100	27,600	11.9
	125	42,800	11.9
	145	59,600	11.9
	208	133,600	11.9

look-ahead, we reduce the search space by a factor of three without any loss in recognition accuracy. Finally, by using the bigram LM look-ahead, the search effort is further reduced by a factor of six without a loss in recognition accuracy. Although the overhead caused by the bigram LM look-ahead is 20% and thus, not negligible, it pays off in terms of overall speed-up of the search process. As a result, we obtain for the final size of the active search space 7,900 state hypotheses on average, which should be compared with the total search-space size (see "Refinements and Implementation Issues"):

64k trees·300k arcs/tree·6 HMM states/arc =

$$1.2 \cdot 10^{11} \text{ HMM states.}$$

For a 20k-word vocabulary, the LM look-ahead overhead is much smaller, namely about 3% rather than 20%.

▲ 14. Example of a word graph (three-word vocabulary: A,B,C).

Table 7. Effect of LM Look-Ahead on Search Effort and Word Error Rate (WER) for 64k-word NAB Task (Bigram LM).				
LM Look-Ahead		**Search Effort**		**WER**
Type	*Overhead*	*Trees*	*States*	*%*
no	-	42	167,800	13.9
	-	33	138,200	14.0
	-	25	105,300	14.4
Unigram (PP=1257)	-	65	50,100	13.9
Bigram (PP=237)	20%	28	7,900	13.9

Table 8. Comparison: Word Graph vs. Integrated Search for 64k-word NAB Task (Unigram LM Look-Ahead).				
Method	**LM type**	**Search Effort**		**WER**
		Trees	*States*	*%*
Word graph generation	Bigram (PP=237)	113	39,100	13.9
+ LM rescoring	Trigram (PP=172)	-	-	12.1
Integrated search	Trigram (PP=172)	48	8,200	12.8
		73	15,800	12.1
		100	27,000	11.9

Having optimized the search strategy in such a way, we typically find that 70% or more of the total recognition effort is now spent on computing the log-likelihoods of the emission probability distributions in the HMMs [58]. To speed up these computations, several methods have been proposed [11], [12], [58].

Word Graph Method

In a third series of experiments, we compared the integrated search with the word graph method in conjunction with LM rescoring. The goal here was to experimentally check the validity of the word pair approximation and to show that there is virtually no loss in performance by using a word graph search rather than an integrated search. The results are shown in Table 8. For the word graph method, a word graph was generated using a bigram LM for each test sentence. Using the bigram LM, the single-best sentence word error rate was 13.9%. To be on the safe side, for each sentence, the word graph was generated using a conservatively large beam, namely 113 tree hypotheses and 39,100 state hypotheses on average. By rescoring each word graph with a trigram LM, the word error rate went down to 12.1%.

For the integrated search strategy using the trigram LM, Table 8 shows three recognition experiments that were selected from Table 6. These experiments result in search efforts of 8,200, 15,800, and 27,000 state hypotheses, and word error rates of 12.8%, 12.1%, and 11.9%, respectively. Also from Fig. 6, we know that even by increasing the beam size to 133,600 state hypotheses, there is no improvement in word error rate over 11.9%. Comparing this best word error rate with the word error rate of 12.1% for the word graph method, we can draw the important conclusion that the word pair approximation used for the word graph generation does not virtually deteriorate recognition accuracy. Again, we would like to emphasize that the experiments reported in Table 8 do not allow a comparison in terms of search effort since the word graphs generated were conservatively large.

In summary, we can say that these and more systematic experiments have shown [56] that the word pair approximation generates high-quality word graphs. In conjunction with LM rescoring, it is competitive with integrated search.

Extensions and Modifications

There are a number of issues that have not been addressed in this article:

▲ The look-ahead strategy can be extended to the acoustic vectors and is then referred to as phoneme look-ahead [22], [54].

▲ There is a type of recombination that has not been considered so far, namely the so-called subtree dominance [1], [2], [53]. This concept results in a sort of minimax criterion and allows whole subtrees of hypotheses to be pruned during search under certain conditions.

▲ The search method can be extended to handle across-word phoneme models [50]. This modification affects the LM and acoustic recombinations in the first are generation of the lexical tree.

▲ The search concept presented is based on what is called word-conditioned structure of the search space. An alternative is to consider a time-conditioned structure, for which the experiments have shown only slightly inferior results [59]. Such an approach has a larger resemblance to stack decoding [8], [60], [61], [63].

▲ The tree-based search can be used in a forward-backward concept, where a simplified lexicon tree produces forward scores at a small computational effort. A second pass, the backward pass, then produces the detailed scores and the final word sequence or word lattice [49]. By adding additional passes, we obtain the multi-pass approach [16].

▲ The search strategy presented here has been designed for bigram and trigram language models, for which, all m-gram language models are of the finite-state type. For other types of language models such as context-free grammars, the search strategy must be suitably modified [44].

Summary

In this article, we have attempted to present a unifying view of the dynamic programming approach to the search problem in continuous-speech recognition. Starting from the baseline one-pass algorithm using a linear organization of the pronunciation lexicon, we have extended the baseline algorithm toward various dimensions. To handle a large vocabulary, we have shown how the search space can be structured in combination with a lexical prefix tree organization of the pronunciation lexicon. In addition, we have shown how this structure of the search space can be combined with a time-synchronous beam search concept and how the search space can be constructed dynamically during the recognition process. In particular, to increase the efficiency of the beam search concept, we have integrated the language model look-ahead into the pruning operation. To produce sentence alternatives rather than only the single best sentence, we have extended the search strategy to generate a word graph. Finally, we have reported experimental results on a 64k-word task that demonstrate the efficiency of the various search concepts presented.

Hermann J. Ney is a Professor in the Computer Science Department of Aachen University of Technology, Aachen, Germany. His research interests cover all aspects of stochastic modeling for pattern and speech recognition, such as signal processing, statistical inference, language modelling, language translation and search strategies for decision making in context.

Stefan Ortmanns received his Dipl.-Ing. degree in mechanical engineering from Aachen University of Applied Sciences, Aachen, Germany, in 1987. He received his Dipl.-Inform. and Ph.D. in computer science from Aachen University of Technology in 1993 and 1998, respectively. From 1993 to 1998, he was a research assistant in the Computer Science Department of Aachen University of Technology. In July 1998, he joined the Dialogue Research Department of Bell Labs-Lucent Technologies, Murray Hill, NJ.

References

[1] F. Alleva, X. Huang, M.Y. Hwang, "An improved search algorithm using incremental knowledge for continuous speech recognition," *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Minneapolis, MN, vol. II, pp. 307-310, April 1993.

[2] 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.

[3] 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.

[4] 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.

[5] X. Aubert, C. Dugast, H. Ney, V. Steinbiss, "Large vocabulary, continuous speech recognition of *Wall Street Journal* corpus," *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Adelaide, Australia, vol. II, pp. 129-132, April 1994.

[6] S. Austin, R. Schwartz, P. Placeway, "The forward-backward search algorithm," *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Toronto, Canada, pp. 697-700, May 1991.

[7] L.R. Bahl, F. Jelinek, R.L. Mercer, "A maximum likelihood approach to continuous speech recognition," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 5, pp. 179-190, March 1983.

[8] L.R. Bahl, F. Jelinek, "Decoding for channels with insertions, deletions and substitutions with applications to speech recognition," *IEEE Trans. on Information Theory*, vol. 21, pp. 404-411, July 1975.

[9] J.K. Baker, "Stochastic modeling for automatic speech understanding," *Speech Recognition*, D.R. Reddy, Ed., New York: Academic Press, pp. 512-542, 1975.

[10] R.E. Bellman, *Dynamic Programming*, Princeton, NJ: Princeton University Press, 1957.

[11] P. Beyerlein, M. Ullrich, "Hamming distance approximation for a fast log-likelihood computation for mixture densities," *Proc. Europ. Conf. on Speech Communication and Technology*, Madrid, Spain, pp. 1083-1086, Sep. 1995.

[12] E. Bocchieri, "Vector quantization for the efficient computation of continuous density likelihoods," *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Minneapolis, MN, pp. 692-695, April 1993.

[13] J.S. Bridle, M.D. Brown, R.M. Chamberlain, "An algorithm for connected word recognition," *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Paris, pp. 899-902, May 1982.

[14] R. Cardin, Y. Normandin, R. DeMori, "High-performance connected digit recognition using codebook exponents," *Proc. IEEE Int. Conf. on Acoustics, Speech ans Signal Processing*, San Francisco, CA, vol. 1, pp. 505-508, March 1992.

[15] S. Della Pietra, V. Della Pietra, J. Gillett, J. Lafferty, H. Printz, L. Ures, "Inference and estimation of a long-range trigram model," *2nd Int. Coll. Grammatical Inference and Applications*, Alicante, Spain, Sep. 1994, pp. 78-92, Springer, Berlin, 1994.

[16] N. Desmukh, A. Ganapathiraju, J. Picone, "Hierarchical search for large vocabulary conversational speech recognition," *IEEE Signal Processing Magazine*, to appear, Sep. 1999.

[17] V. Digalakis, H. Murveit, "Genones: optimizing the degree of mixture tying in a large vocabulary hidden Markov model-based speech recognizer," *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Adelaide, Australia, vol. I, pp. 537-540, April 1994.

[18] 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, Jan. 1995.

[19] L. Fissore, E. Giachin, P. Laface, P. Massafra, "Using grammars in forward and backward search," *Proc. Europ. Conf. on Speech Communication and Technology*, Berlin, pp. 1525-1528, Sep. 1993.

[20] G.D. Forney, "The Viterbi algorithm," *Proc. IEEE*, vol. 61, pp. 268-278, March 1973.

[21] J.L. Gauvain, L.F. Lamel, G. Adda, M. Adda-Decker, "The LIMSI speech dictation system: evaluation on the ARPA *Wall Street Journal* task," *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Adelaide, Australia, vol. I, pp. 557-560, April 1994.

[22] R. Haeb-Umbach, H. Ney, "Improvements in time-synchronous beam search for 10,000-word continuous speech recognition," *IEEE Trans. on Speech and Audio Processing*, vol. 2, pp. 353-356, April 1994.

[23] M.-Y. Hwang, R. Rosenfeld, E. Thayer, R. Mosur, L. Chase, R. Weide, X. Huang, F. Alleva, "Improving speech recognition performance via phone-dependent VQ codebooks and adaptive language models in SPHINX II," *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Adelaide, Australia, vol. I, pp. 549-552, April 1994.

[24] F. Itakura, "Minimum prediction residual principle applied to speech recognition," *IEEE Trans. on Acoustics, Speech and Signal Processing*, vol. ASSP-23, pp. 67-72, Feb. 1975.

[25] F. Jelinek, "Continuous speech recognition by statistical methods," *Proc. IEEE*, vol. 64, no. 10, pp. 532-556, April 1976.

[26] D. H. Klatt, "Overview of the ARPA speech understanding project," pp. 249-271, in *Trends in Speech Recognition*, W.A. Lea, Ed., Englewood Cliffs, NJ: Prentice-Hall, 1980.

[27] D.H. Klatt, "SCRIBER and LAFS: two new approaches to speech analysis," pp. 529-555, in *Trends in Speech Recognition*, W.A. Lea, Ed., Englewood Cliffs, NJ: Prentice-Hall, 1980.

[28] J.M .Klovstad, L.F. Mondshein, "The CASPERS linguistic analysis system," *IEEE Trans. Acoustics, Speech and Signal Processing*, vol. 23, pp. 118-123, Feb. 1975.

[29] F. Kubala, "Design of the 1994 CSR benchmark tests," *Proc. ARPA Spoken Language Technology Workshop*, Austin, TX, pp. 41-46, Jan. 1995.

[30] F. Kubala, A. Anastasakos, J. Makhoul, L. Nguyen, R. Schwartz, "Comparative experiments on large vocabulary speech recognition," *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Adelaide, Australia, vol. 1, pp. 561-564, April 1994.

[31] R. Kuhn, R. De Mori, "A cache-based natural language model for speech recognition," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 12, pp. 570-583, June 1990.

[32] T. Kuhn, P. Fetter, A. Kaltenmeier, P. Regel-Brietzmann, "DP-based word graph pruning," *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Atlanta, GA, pp. 461-464, May 1996.

[33] R. Lacouture, Y. Normandin, "Efficient lexical access strategies," *Proc. Europ. Conf. on Speech Communication and Technology*, Berlin, Germany, pp. 1537-1540, Sep. 1993.

[34] C.H. Lee, L.R. Rabiner, "A frame-synchronous network search algorithm for continuous word recognition," *IEEE Trans. Acoustics, Speech and Signal Processing*, vol. ASSP-37, no. 11, pp. 1649-1658, Nov. 1989.

[35] S.E. Levinson, L.R. Rabiner, M.M. Sondhi, "An introduction to the application of the theory of probabilistic functions of a Markov process to automatic speech recognition," *The Bell System Technical Journal*, vol. 62, no. 4, pp. 1035-1074, April 1983.

[36] A. Ljolje, M. Riley, D. Hindle, F. Pereira, "The AT&T 60,000 word speech-to-text system," *Proc. ARPA Spoken Language Systems Technology Workshop*, Austin, TX, pp. 162-165, Jan. 1995.

[37] B.T. Lowerre, R. Reddy, "The HARPY speech understanding system," pp. 340-360, in *Trends in Speech Recognition*, W.A. Lea, Ed., Englewood Cliffs, NJ: Prentice-Hall, 1980.

[38] K. Mehlhorn, *Data Structures and Algorithms 1: Sorting and Searching*, Springer, Berlin, Germany, 1984.

[39] H. Murveit, J. Butzberger, V. Digalakis, M. Weintraub, "Large-vocabulary dictation using SRI's decipher speech recognition system: progressive-search techniques," *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Minneapolis, MN, vol. II, pp. 319-322, April 1993.

[40] C.S. Myers, L.R. Rabiner, "Connected digit recognition using a level-building DTW algorithm," *IEEE Trans. Acoustics, Speech and Signal Processing*, vol. ASSP-29, no. 3, pp. 351-363, June 1981.

[41] H. Ney, "Connected utterance recognition using dynamic programming," *Fortschritte der Akustik - FASE/DAGA'82*, Federation of Acoustic Societies of Europe/Deutsche Arbeitsgemeinschaft für Akustik, Göttingen, Germany, pp. 915-918, Sept. 1982.

[42] H. Ney, "The use of a one-stage dynamic programming algorithm for connected word recognition," *IEEE Trans. Acoustics, Speech and Signal Processing*, vol. ASSP-32, no. 2, pp. 263-271, April 1984.

[43] H. Ney, "Acoustic modelling of phoneme units for continuous speech recognition," *Proc. Fifth European Signal Processing Conference*, Barcelona, Spain, pp. 65-72, Sep. 1990.

[44] H. Ney, "Dynamic programming parsing for context-free grammars in continuous speech recognition," *IEEE Trans. Signal Processing*, vol. SP-39, no. 2, pp. 336-341, Feb. 1991.

[45] H. Ney, "Search strategies for large-vocabulary continuous-speech recognition," in *Speech Recognition and Coding—New Advances and Trends*, A.J. Rubio Ayuso, J.M. Lopez Soler, Eds., NATO Advanced Studies Institute, Bubion, Spain, June-July 1993, pp. 210-225, Berlin: Springer, 1995.

[46] H. Ney, R. Haeb-Umbach, B.-H. Tran, M. Oerder, "Improvements in beam search for 10,000-word continuous speech recognition," *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, San Francisco, CA, pp. 13-16, March 1992.

[47] H. Ney, D. Mergel, A. Noll, A. Paeseler, "Data driven organization of the dynamic programming beam search for continuous speech recognition," *IEEE Trans. Signal Processing*, vol. SP-40, no. 2, pp. 272-281, Feb. 1992.

[48] H. Ney, S. Ortmanns, "Extensions to the word graph method for large vocabulary continuous speech recognition," *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Munich, Germany, vol. 3, pp. 1787-1790, April 1997.

[49] L. Nguyen, R. Schwartz, "Single-tree method for grammar-directed search," *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Phoenix, AZ, vol. II, pp. 613-616, March 1999.

[50] J.J. Odell, V. Valtchev, P.C. Woodland, S.J. Young, "A one-pass decoder design for large vocabulary recognition," *ARPA Spoken Language Technology Workshop*, Plainsboro, NJ, pp. 405-410, March 1994.

[51] J.J. Odell, "The use of context in large vocabulary speech recognition," Ph.D. Thesis, Electrical Engineering Department, Cambridge University, UK, March 1995.

[52] M. Oerder, H. Ney, "Word graphs: an efficient interface between continuous speech recognition and language understanding," *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Minneapolis, MN, vol. 2, pp. 119-122, April 1993.

[53] S. Ortmanns, A. Eiden, H. Ney, "Improved lexical tree search for large vocabulary speech recognition," *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Seattle, WA, pp. 817-820, May 1998.

[54] S. Ortmanns, A. Eiden, H. Ney, N. Coenen, "Look-ahead techniques for fast beam search," *Int. Conf. on Acoustics, Speech and Signal Processing*, Munich, Germany, vol. 3, pp. 1783-1786, April 1997.

[55] S. Ortmanns, H. Ney, "Experimental analysis of the search space for 20,000-word speech recognition," *Proc. Fourth European Conference on Speech Communication and Technology*, Madrid, Spain, pp. 901-904, Sep. 1995.

[56] S. Ortmanns, H. Ney, X. Aubert, "A word graph algorithm for large vocabulary continuous speech recognition," *Computer, Speech and Language*, vol. 11, no. 1, pp. 43-72, Jan. 1997.

[57] S. Ortmanns, H. Ney, A. Eiden, "Language-model look-ahead for large vocabulary speech recognition," *Proc. Int. Conf. on Spoken Language Processing*, Philadelphia, PA, pp. 2095-2098, Oct. 1996.

[58] S. Ortmanns, H. Ney, T. Firzlaff, "Fast-likelihood computation methods for continuous mixture densities in large vocabulary speech recognition," *Fifth European Conf. on Speech Communication and Technology*, Rhodes, Greece, pp. 143-146, Sep. 1997.

[59] S. Ortmanns, H. Ney, F. Seide, I. Lindam, "A comparison of time conditioned and word conditioned search techniques for large vocabulary speech recognition," *Int. Conf. on Spoken Language Processing*, Philadelphia, PA, pp. 2091-2094, Oct. 1996.

[60] D.B. Paul, "Algorithms for an optimal search and linearizing the search in the stack decoder," *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Toronto, Canada, pp. 693-996, May 1991.

[61] D.B. Paul, "An efficient A* stack decoder algorithm for continuous speech recognition with a stochastic language model," *Proc. IEEE Int. Conf. on Acoustics, Speech ans Signal Processing*, San Francisco, CA, pp. 25-28, March 1992.

[62] L.R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proc. IEEE*, vol. 77, no. 2, pp. 257-286, Feb. 1989.

[63] S. Renals, M. Hochberg, "Efficient search using posterior phone probability estimates," *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Detroit, MI, pp. 596-599, May 1995.

[64] S. Renals, M. Hochberg, "Efficient evaluation of the LVCSR search space using the NOWAY decoder," *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Atlanta, GA, pp. 149-152, May 1996.

[65] H. Sakoe, "Two-level DP matching—a dynamic programming-based pattern matching algorithm for connected word recognition," *IEEE Trans. Acoustics, Speech and Signal Processing*, vol. ASSP-27, pp. 588-595, Dec. 1979.

[66] H. Sakoe, S. Chiba, "A dynamic programming approach to continuous speech recognition," *Proc. 7th Int. Congr. on Acoustics*, Budapest, Hungary, Paper 20 C 13, pp. 65-68, August 1971.

[67] R. Schwartz, S. Austin, "A comparison of several approximate algorithms for finding multiple (N-best) sentence hypotheses," *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Toronto, pp. 701-704, May 1991.

[68] A. Sixtus, S. Ortmanns, "High-quality word graphs using forward-backward pruning," *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Phoenix, AZ, vol. II, pp. 593-596, March 1999.

[69] F.K. Soong, E.-F. Huang, "A tree-trellis fast search for finding the N-best sentence hypotheses," *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Toronto, pp. 705-708, May 1991.

[70] V. Steinbiss, "A search organization for large vocabulary recognition based upon N-best decoding," *Proc. 2nd European Conf. on Speech Communication and Technology*, Genova, vol. 3, pp. 1217-1220, Sep. 1991.

[71] V. Steinbiss, *Personal Communication*, Philips Research Laboratories, Aachen, Germany, June 1992.

[72] V. Steinbiss, H. Ney, R. Haeb-Umbach, B.-H. Tran, U. Essen, R. Kneser et al., "The Philips research system for large-vocabulary continuous-speech recognition," *Third European Conference on Speech Communication and Technology*, Berlin, pp. 2125-2128, Sep. 1993.

[73] V. Steinbiss, B.-H. Tran, H. Ney, "Improvements in beam search," *Int. Conf. on Spoken Language Processing*, Yokohama, Japan, pp. 1355-1358, Sep. 1994.

[74] C. Tillmann, H. Ney, "Word triggers and the EM algorithm," *ACL Special Interest Group Workshop on Computational Natural Language Learning* (Assoc. for Comput. Linguistics), Madrid, pp. 117-124, July 1997.

[75] V.M. Velichko, N.G. Zagoruyko, "Automatic recognition of 200 words," *Int. Journal Man-Machine Studies*, vol. 2, pp. 223-234, June 1970.

[76] T.K. Vintsyuk, "Speech discrimination by dynamic programming," *Kibernetika (Cybernetics)*, vol. 4, No. 1, pp. 81-88, Jan.-Feb. 1968.

[77] T.K. Vintsyuk, "Elementwise recognition of continuous speech composed of words from a specified dictionary," *Kibernetika (Cybernetics)*, vol. 7, pp. 133-143, March-April 1971.

[78] D. Wood, *Data structures, Algorithms, and Performance*, Addison-Wesley, Reading, MA, 1993.

[79] P.C. Woodland, J.J. Odell, V. Valtech, S.J. Young, "Large vocabulary continuous speech recognition using HTK," *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Adelaide, Australia, vol. II, pp. 125-128, April 1994.

[80] S.J. Young, J.J. Odell, P.C. Woodland, "Tree-based state tying for high accuracy acoustic modelling," *ARPA Human Language Technology Workshop*, Plainsboro, NJ, San Mateo, CA: Morgan Kaufmann Publishers, pp. 286-291, March 1994.

[81] Q. Zhou, W. Chou, "An approach to continuous speech recognition based on a layered self-adjusting decoding graph," *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Munich, Germany, pp. 1779-1782, April 1997.