Progress in Dynamic Programming Search for LVCSR

Hermann Ney, Stefan Ortmanns
Lehrstuhl für Informatik VI, Computer Science Department
RWTH Aachen, University of Technology
D-52056 Aachen, Germany

Abstract - This paper gives an overview of the recent improvements in dynamic programming search for large vocabulary continuous speech recognition: search using lexical trees, time-conditioned search and word graph construction.

1 Introduction

Search 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 had already been known in the early days of automatic speech recognition [5, 14, 17]. Over the past two decades, these and related DP strategies have turned out to be surprisingly successful in handling vocabularies of 20k and more words. Nevertheless, until recently, it was among the experts a highly controversial issue whether high-perplexity LVCSR could be handled efficiently by DP. The scepticism seems to have been concerned mainly with the following issues:

- 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 low-level acoustic search interact efficiently with the higher knowledge sources in the recognition system such as the pronunciation lexicon and language model (LM) ?
- DP typically computes only the best single 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 ?

In the following, we try to present an overview of the progress in the applications of DP strategies to LVCSR. Due to space limitations, we present only the basic principles and omit most details.

2 The specification of the search problem

The acoustic models are given in terms of Hidden Markov Models (HMMs) representing either context dependent or independent phoneme units. For a hypothesized word sequence $w_1^N = w_1 ... w_N$, we imagine a super HMM that is obtained by concatenating the corresponding phoneme HMMs using a pronunciation lexicon. At phoneme and word boundaries, we have to allow for transitions that link the terminal states of any predecessor HMM to the initial states of any successor HMM. Thus, we can compute the joint probability of observing the sequence $x_1^T = x_1 ... x_T$ of acoustic input vectors and the state sequence $s_1^T = s_1 ... s_T$ through this super HMM:

$$Pr\left(x_1^T, s_1^T \middle| w_1^N\right) = \prod_{t=1}^{T} p(x_t, s_t | s_{t-1}, w_1^N) \ , \qquad (1)$$

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.

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

$$\max_{w_1^N} \left\{ Pr(w_1^N) \cdot \max_{s_1^T} \ Pr\left(x_1^T, s_1^T \middle| w_1^N\right) \right\} \ . \qquad (2)$$

Here, we have made use of the so-called maximum approximation. 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 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 at both levels efficiently 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 property allows an efficient pruning strategy to eliminate unlikely search hypotheses, which is referred to as beam search.

3 DP one-pass search using a lexical prefix tree

Originally, the one-pass DP algorithm [2, 6, 17] had been designed for small-vocabulary continuous speech recognition. Within the maximum approximation, the one-pass DP algorithm presents a closed-form solution for handling the interdependence of the various operations in continuous speech recognition: nonlinear time alignment, word boundary detection, taking into account the LM, and word identification.

When applying the one-pass DP algorithm to large-vocabulary recognition, say a 20k-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 unit [7]. This idea of using a tree representation was already suggested in the seventies in the CASPERS system [4] and in the LAFS system (LAFS = lexical access from spectra) [3]. However, when using such a lexical tree in connection with a typical LM, e. g. a bigram model, it is by no means clear how to apply the DP strategy to this task. The difficulty is that the DP strategy can be applied in a straightforward way *only* if there are no dependencies on previous decisions. In other words, the network for the path-finding task must have such a structure that the cost of each edge depends only on the edge and nothing else.

When using a bigram language model in connection with such a prefix tree representation of the pronunciation lexicon (for short: lexical tree), we face the problem that a complete word hypothesis is generated only when an end node of the lexical tree has been reached. Therefore the LM bigram probability $p(w|v)$ can only be fully incorporated after reaching the terminal state of the second word w of the bigram. For this operation, however, we have to make sure that the hypotheses for each possible predecessor word v are still available. Therefore, 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.

Throughout the paper for notational convenience, we will always consider the state index s of an arc rather than the arc itself and assume that the lexical structure is fully captured by the transition probabilities of the super HMM representing the lexical tree. To formulate the DP approach, we introduce the following quantity [12]:

$$Q_v(t,s) \quad := \quad \text{overall score of the best partial path that at time } t \text{ ends in state } s \text{ of the lexical tree for predecessor word } v.$$

Within words, i. e. for internal states of the lexical trees, we have then the well-known DP recursion for time alignment:

$$Q_v(t,s) \quad = \quad \max_{\sigma} \; \{ \; p(x_t, s|\sigma) \cdot Q_v(t-1,\sigma) \; \} \; , \tag{3}$$

where by construction, $p(x_t, s|\sigma)$, the product of HMM transition and emission probabilities does *not* depend on any word index. To allow for word boundaries, we have to find the best predecessor word u for each word v when a path hypothesis reaches the associated terminal state S_v. Thus, at the word level, we have the DP recursion:

$$Q_v(t, s = 0) \quad = \quad \max_u \ \{ \ p(v|u) \cdot Q_u(t, S_v) \ \} \ , \tag{4}$$

where we have assumed a fictitious state $s = 0$ for the root of the lexical tree. As a result, the optimization over the unknown word boundary is included in the above two DP recursions. The above DP recursions are evaluated in a strictly left-to-right time-synchronous fashion [7]. To avoid storing the full table $Q_v(t, s)$ and to arrive at a memory-efficient implementation, so-called back pointers were found to be extremely useful [2, 6, 8].

4 Beam search and LM look-ahead

For large vocabulary speech recognition, the search method described cannot be used without somehow limiting the possible search space. As an example, we estimate the size of the search space for the 20k-word WSJ system [12]:

20k trees * 60k arcs/tree * 6 HMM states/arc = 7.2 * 10^9 HMM states .

Therefore, in full search, there are this number of 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, experimental tests show that by using the beam search method (in its most refined form) this number can be reduced to less than an average of 4000 HMM states per time frame (without loss in performance). In beam search, at each time frame, only the most promising hypotheses $Q_v(t, s)$ are retained for further evaluation; the other hypotheses are removed [5].

Due to the delayed application of the LM probabilities in tree search, the efficiency of this pruning method is drastically improved by including the LM probabilities as early as possible into the pruning process. This operation, referred to as LM look-ahead, works as follows [9, 11, 16]. Considering a state s in the lexical tree for predecessor word v, we know that the extensions of this state hypothesis cannot produce all words w of the vocabulary, but only a certain subset, which we denote by $\mathcal{W}(s)$. Thus to anticipate the effect of the LM probabilities, we compute the probability of the most likely word that can be generated for each state s and predecessor word v:

$$\pi_v(s) \ := \ \max_{w \in \mathcal{W}(s)} p(w|v) \ . \tag{5}$$

To incorporate the anticipated LM probabilities $\pi_v(s)$ into the beam search strategy, we consider the modified scores:

$$\tilde{Q}_v(t, s) \quad = \quad \pi_v(s) \cdot Q_v(t, s) \ . \tag{6}$$

At time t, we compare each hypothesis $\tilde{Q}_v(t, s)$ with the *best* hypothesis at that time. We prune a hypothesis (v, s, t) with score $Q_v(t, s)$ if:

$$\tilde{Q}_v(t, s) < f_0 \cdot \max_{v' s'} \left\{ \tilde{Q}_{v'}(t, s') \right\} \ , \tag{7}$$

where the threshold $f_0 < 1$ is used to control the number of surviving state hypotheses. In addition to this type of pruning, there are other (however less important) types of pruning such as language model pruning and subtree dominance pruning [1, 12, 16].

To fully exploit the advantages of beam search, a dynamic construction of the search space is necessary [8]. In addition, when computing all entries of the table $\pi_v(s)$ beforehand, we have to keep a huge table in main memory which is prohibitive. Therefore we calculate the entries of the table $\pi_v(s)$ on demand and cache them [9, 11].

5 From word-conditioned to time-conditioned search

The search strategy described so far can be called a word-conditioned strategy because the hypotheses $Q_v(s,t)$ are conditioned on the predecessor words v. Another approach is to structure the search space by using the end time of the predecessor word or equivalently the start time of the successor word. This concept has already been used in other contexts, namely connected digit recognition [14] and word lattice generation [10]. Without a time-synchronous evaluation, this concept has certain similarities to stack decoding or A^* search. To describe the time-conditioned search strategy and to arrive at the DP formulation, we define the following quantities as introduced in [13]:

$$\begin{aligned} h(w;\tau,t) &:= \max_{s_{\tau+1}^t} Pr(x_{\tau+1}^t, s_{\tau+1}^t | w) \\ &= \text{probability that word } w \text{ produces the acoustic vectors } x_{\tau+1}^t. \\ G(w_1^n;t) &:= Pr(w_1^n) \cdot \max_{s_1^t} Pr(x_1^t, s_1^t | w_1^n) \\ &= \text{probability of observing the acoustic vectors } x_1^t \\ & \text{ and a word sequence } w_1^n \text{ with end time } t. \end{aligned}$$

By using these definitions, we can isolate the probability contributions of a particular word hypothesis with respect to both the LM and the acoustic model:

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

To extend the word sequence hypothesis w_1^{n-1} by one more word w_n, we have to optimize over the unknown word boundary τ:

$$\begin{aligned} G(w_1^n;t) &= \max_\tau \{Pr(w_n|w_1^{n-1}) \cdot G(w_1^{n-1};\tau) \cdot h(w_n;\tau,t)\} && (8) \\ &= Pr(w_n|w_1^{n-1}) \cdot \max_\tau \{G(w_1^{n-1};\tau) \cdot h(w_n;\tau,t)\}\ , && (9) \end{aligned}$$

where we have incorporated the conditional LM probability $Pr(w_n|w_1^{n-1})$. So far, we have not constrained the LM in any way, and no recombination of hypotheses is possible. This, however, is different for an m-gram LM $p(v_m|v_1^{m-1})$. We define:

$$\begin{aligned} H(v_2^m;t) \quad &:= \quad \max_{w_1^n} \left\{ Pr(w_1^n) \cdot \max_{s_1^t} Pr(x_1^t, s_1^t|w_1^n) : w_{n-m+2}^n = v_2^m \right\} \\ &= \quad \text{probability of observing the acoustic vectors } x_1^t \text{ and a} \\ & \qquad \text{word sequence with \textit{end} sequence } v_2^m \text{ and \textit{end} time } t. \end{aligned}$$

Thus, we have the DP recursion:

$$H(v_2^m;t) = \max_{v_1} \left\{ p(v_m|v_1^{m-1}) \cdot \max_{\tau} \{ H(v_1^{m-1};\tau) \cdot h(v_m;\tau,t) \} \right\} \ . \qquad (10)$$

To compute the scores $h(v_m;\tau,t)$ efficiently, we again use a prefix tree for the pronunciation lexicon and organize all evaluation steps in a left-to-right time-synchronous strategy. Stretching notation, we use the symbol $Q_\tau(t,s)$ to denote tree internal search hypotheses with respect to start time τ:

$$\begin{aligned} Q_\tau(t,s) \quad &:= \quad \text{overall score of the best partial path that at time } t \text{ ends} \\ & \qquad \text{in state } s \text{ of the lexical tree with start time } \tau. \end{aligned}$$

Before moving from time frame $(t-1)$ to t, we have to allow for a word boundary hypothesis. To this purpose, we start up a new copy of the lexical tree at time $\tau = t-1$:

$$Q_{t-1}(t-1,s) = \begin{cases} H_{max}(t-1) & \text{if } s = 0 \\ 0 & \text{if } s > 0 \ , \end{cases} \qquad (11)$$

where, again, the fictitious state $s = 0$ is used for initialization and $H_{max}(t-1)$ is defined to be the best existing hypothesis:

$$H_{max}(t-1) \quad := \quad \max_{v_1^{m-1}} H(v_1^{m-1};t-1) \ . \qquad (12)$$

Note that there are only hypotheses $Q_\tau(t,s)$ for $\tau < t$. In the tree interior, we have the usual DP recursion:

$$Q_\tau(t,s) = \max_{\sigma} \ \{ p(x_t,s|\sigma) \cdot Q_\tau(t-1,\sigma) \} \ . \qquad (13)$$

Of course, in beam search for a time frame t, typically only a few different start times τ, say 50, are expected to survive. When performing the word boundary optimization, we have to 'correct' each word end score $Q_\tau(t,S_{v_m})$ for the 'wrong' start score $H_{max}(\tau)$ and thus obtain the DP recursion:

$$H(v_2^m;t) = \max_{v_1} \left\{ p(v_m|v_1^{m-1}) \cdot \max_{\tau} \left\{ H(v_1^{m-1};\tau) \cdot \frac{Q_\tau(t,S_{v_m})}{H_{max}(\tau)} \right\} \right\} \ . \qquad (14)$$

Back pointers as used in the word-conditioned search method are not needed since each tree copy is directly conditioned on its start time τ.

6 From the single best sentence to a word graph

We consider the problem of word graph construction:

> Hypothesizing a word and its end time, how can we find a limited number of 'most likely' predecessor words? This task is difficult since the start time of each word may very well depend on the predecessor word under consideration, which results in an interdependence of start times and predecessor words.

Considering the success of the 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.

To construct a word graph, we introduce a formal definition of the word boundary $\tau(w_1^n;t)$ between the word hypothesis w_n ending at time t and the predecessor sequence hypothesis w_1^{n-1}:

$$\tau(t;w_1^n) \quad := \quad \arg\max_\tau \left\{G(w_1^{n-1};\tau)\cdot h(w_n;\tau,t)\right\} \ . \tag{15}$$

It should be emphasized that the LM probability does *not* affect the optimal word boundary according to Eq.(9) and is therefore omitted in the definition of the word boundary function $\tau(w_1^n;t)$. The crucial assumption now is that the dependence of the word boundary $\tau(t;w_1^n)$ can be confined to the final word pair w_{n-1}^n. The justification is that the other words have virtually no effect on the position of the word boundary between words w_{n-1} and w_n [15]. In general, this boundary, i. e. the start time of word w_n as given by time alignment, will depend on the immediate predecessor word w_{n-1}: if the predecessor word w_{n-1} is sufficiently long, all time alignment paths are recombined before they reach the final state of the predecessor word. In formulae, we express this word pair approximation by the equation:

$$\tau(t;w_1^n) = \text{const}(w_1^{n-2}) \quad \text{or} \quad \tau(t;w_1^n) = \tau(t;w_{n-1}^n) \ , \tag{16}$$

i. e. the word boundary between words w_{n-1} and w_n does *not* depend on the predecessor words w_1^{n-2}. Under the assumption of this word pair approximation, it turns out that only a small modification in the bookkeeping of the word-conditioned tree search is needed [12]: when hypothesizing a word boundary as expressed by Eq.(4), the algorithm must keep track of the score $Q_v(t,S_w)$ of *every* word pair (v,w) to generate a word graph rather than only the *best* predecessor word v for each successor word w.

7 Experimental results

The methods described in this paper were tested in many systems. We only summarize the results:

- In all experiments reported, e. g. see [1, 9, 13, 16], the word-conditioned DP search in connection with beam search and LM look-ahead was found to be surprisingly efficient. When using a trigram LM in lieu of a bigram LM, the average number of hypotheses is only slightly increased. Of course, in all these cases, it is important to make use of a clever memory organization.

- Systematic experiments [12] showed that the DP word graph generation based on the word pair approximation does not deteriorate performance, even for very short words. Thus there is virtually no degradation in recognition performance due to using DP word graphs.

- The time-conditioned DP search was tested successfully in [13]. In terms of search efficiency, it seems to be slightly inferior to word-conditioned DP search. However for larger vocabularies and more complex language models, the situation might be different.

References

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

[2] J. S. Bridle, M. D. Brown, R. M. Chamberlain: An Algorithm for Connected Word Recognition. IEEE Int. Conf. on Acoustics, Speech and Signal Processing, Paris, pp. 899-902, May 1982.

[3] D. H. Klatt: SCRIBER and LAFS: Two New Approaches to Speech Analysis. pp. 529-555, in W. A. Lea (ed.): ’Trends in Speech Recognition’, Prentice-Hall, Englewood Cliffs, NJ, 1980.

[4] J. W. Klovstad, L. F. Mondshein: The CASPERS Linguistic Analysis System. IEEE Trans. on Acoustics, Speech and Signal Processing, Vol. 23, pp. 118-123, Feb. 1975.

[5] B. T. Lowerre, R. Reddy: The HARPY Speech Understanding System. pp. 340-360, in W. A. Lea (ed.): ’Trends in Speech Recognition’, Prentice-Hall, Englewood Cliffs, NJ, 1980.

[6] H. Ney: The Use of a One-Stage Dynamic Programming Algorithm for Connected Word Recognition. IEEE Trans. on Acoustics, Speech and Signal Processing, Vol. ASSP-32, No. 2, pp. 263-271, April 1984.

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

[8] H. Ney, D. Mergel, A. Noll, A. Paeseler: Data Driven Organization of the Dynamic Programming Beam Search for Continuous Speech Recognition. IEEE Trans. on Signal Processing, Vol. SP-40, No. 2, pp. 272-281, Feb. 1992.

[9] 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.

[10] M. Oerder, H. Ney: Word Graphs: An Efficient Interface Between Continuous Speech Recognition and Language Understanding. IEEE Int. Conf. on Acoustics, Speech and Signal Processing, Minneapolis, MN, Vol.II, pp. 119-122, April 1993.

[11] S. Ortmanns, A. Eiden, H. Ney, N. Coenen: Look-Ahead Techniques for Fast Beam Search. Int. Conf. on Acoustics, Speech and Signal Processing, Munich, Vol. 3, pp. 1783-1786, April 1997.

[12] 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.

[13] 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.

[14] H. Sakoe: Two-Level DP Matching - A Dynamic Programming-Based Pattern Matching Algorithm for Connected Word Recognition. IEEE Trans. on Acoustics, Speech and Signal Processing, Vol. ASSP-27, pp. 588-595, December 1979.

[15] R. Schwartz, S. Austin: A Comparison of Several Approximate Algorithms for Finding Multiple (N-Best) Sentence Hypotheses. IEEE Int. Conf. on Acoustics, Speech and Signal Processing, Toronto, pp. 701-704, May 1991.

[16] V. Steinbiss, B.-H. Tran, H. Ney: Improvements in Beam Search. Int. Conf. on Spoken Language Processing, Yokohama, Japan, pp. 1355-1358, Sep. 1994.

[17] T. K. Vintsyuk: Elementwise Recognition of Continuous Speech Composed of Words from a Specified Dictionary. Cybernetics, Vol. 7, pp. 133-143, March-April 1971.