A DATA-DRIVEN ORGANIZATION OF THE DYNAMIC PROGRAMMING BEAM SEARCH FOR CONTINUOUS SPEECH RECOGNITION

H. Ney, D. Mergel, A. Noll, A. Paeseler

Philips GmbH Forschungslaboratorium Hamburg,
P.O. Box 54 08 40, D-2000 Hamburg 54, FRG

ABSTRACT- This paper describes a data-driven organization of the dynamic programming beam search for large vocabulary, continuous speech recognition. This organization can be viewed as an extension of the one-pass dynamic programming algorithm for connected word recognition. In continuous speech recognition we are faced with a huge search space, and search hypotheses have to be formed at the 10-ms level. The organization of the search presented has the following characteristics. Its computational cost is proportional only to the number of hypotheses actually generated and is independent of the overall size of the potential search space. There is no limit on the number of word hypotheses, there is only a limit to the overall number of hypotheses due to memory constraints. The implementation of the search has been studied and tested on a continuous speech data base comprising 20672 words.

1. INTRODUCTION

One of the most fundamental and successful concepts in connected word and continuous speech recognition is that of nonlinearly time-aligning an unknown input utterance with reference patterns, which results in a path finding problem [1]. However, in the case of continuous speech recognition, there is typically a huge number of possible paths, and therefore such a concept requires a careful organization. For a small search space as in connected word recognition, the one-pass dynamic programming (DP) algorithm [2,3] provides an efficient technique for performing the search, in particular in combination with a pruning strategy. However, its direct application to a large search space in connection with language constraints leads to a computationally expensive implementation, because the complete search space is explicitly constructed as a whole and each state must be checked as to whether it is still active.

Therefore, the purpose of this paper is to present a data-driven dynamic construction of the state space for the one-pass algorithm so that only the actually active hypotheses are explicitly generated during the process of recognition. Like the conventional one-pass algorithm, the new algorithm proceeds from left to right along the time axis of the input speech signal. The construction of the hypotheses is guided by the language model, the pronunciation lexicon and the inventory of subword units. The algorithm builds up a data structure that contains the relevant information to carry out the recursion for both nonlinear time alignment and the recombination of hypotheses at the nodes of the language model. There is no constraint imposed on the maximum number of active words, the only limitation (due to computer memory) is the absolute number of state hypotheses at the 10-ms level. The computational cost is linearly proportional to the number of calculated hypotheses and independent of the overall search space.

The organization of this paper is as follows. In Section 2, we review the search problem in continuous speech recognition and the DP approach. In Section 3, we develop the data-driven organization of the DP beam search. In Section 4, we report on recognition experiments.

2. DP BEAM SEARCH FOR CONTINUOUS SPEECH RECOGNITION

The system for continuous speech recognition we are considering is based on three knowledge sources which are given as a three-level hierarchy consisting of the language model, the pronunciation lexicon and the inventory of subword units. In principle,these three levels are kept separately and interact only via the search procedure. Since all knowledge sources are made use of by the search procedure, we call this approach an integrated one. For practical reasons, it is often convenient to combine the pronunciation lexicon and the subword units to what is called acoustic word models.

For such a system, the one-pass DP algorithm as described in [2,3] provides a sort of closed-form solution for finding the best explanation of the input data and taking account of the interdependence of nonlinear time alignment, word boundary detection, word identification and language constraints. The algorithm processes all reference patterns in parallel by moving along time axis i of the input utterance and thus performs a strictly left-to-right search. The implementation requires three loops, one over the input frames i, one over the reference words k and one over the states $j = 1,\ldots,J(k)$ of each reference word k. There are two quantities that have to be calculated for all argument triples or grid points (i,j,k): the acoustic score S(i,j,k) and the backpointer B(i,j,k). After the final input frame has been processed, the word sequence giving the best explanation of the input data is recovered by chaining down the traceback arrays [2,3].

The basic problem with this DP approach is that a full search of the state space as defined by the knowledge sources is prohibitive. There are two conflicting effects of the language model. On one hand, since there may be several copies of each acoustic word, the number of word transitions and thus the overall search space is much higher than without language constraints. The size of the overall search space is given by the number of grid points (i,j,k) that have to be processed for every input frame i, i.e. the product of the average number of states in an acoustic word model and the number of word transitions in the language model. As we will see later, there are typically 200 000 grid points to be processed every 10-ms. On the other hand, the language model imposes heavy constraints on the 'legal' word sequences, which results in a large amount of redundancy in the spoken utterance. In particular, from decoding theory it is known that it is sufficient

CH2396-0/87/0000-0833 $1.00 © 1987 IEEE

to consider only a small number of hypotheses without affecting the overall performance too much [5].

A pruning strategy that fits into this framework is usually referred to as beam search, DP beam search or pruned DP search [2,4,6]. It attempts to efficiently exploit the language constraints and to evaluate the DP equations only in the 'relevant' regions of the search space. The pruning strategy is based on the consideration that path hypotheses with matching scores significantly larger than the minimum matching score of the input frame under consideration are very unlikely to result in the optimal path. Thus for each input frame i, all matching scores exceeding a threshold set relative to the best matching score for input frame i can be removed from further consideration. There are two very attractive features about this pruning strategy. First, there is no need to normalize the matching scores under consideration because they are all time synchronous and explain the same beginning portion of the input utterance. Second, there is a self-focussing effect of the search in that the number of hypotheses automatically adjusts to the ambiguity in the speech signal.

3. THE DATA-DRIVEN ORGANIZATION OF THE DP BEAM SEARCH

Before describing our data-driven organization of the DP beam search, it is instructive to have a look at other search strategies that attempt to avoid a full search. In the area of artificial intelligence, a number of search strategies have been developed for machine simulation of cognitive processes, theorem proving and problem solving [7]. These techniques are usually referred to as best-first, ordered or heuristic search. The basic idea they have in common is to maintain an ordered list of path hypotheses and to expand the 'most promising' path hypotheses at each step. Recognition algorithms based on such a principle have been used in some systems [5,8]. In the IBM system [5], the ordered search strategy is referred to as 'stack decoding' and is part of a maximum likelihood decoding technique applied to continuous speech input in the framework of probabilistic finite state machines. In [8], the ordered search strategy is systematically applied to isolated word recognition.

From our point of view, there are a number of drawbacks that reduce the attractivity of these approaches. First, the principle is to minimize the number of visited grid points or nodes and not the cost of the complete search effort. Then there is a lot of overhead due to the ordering and searching of lists, which is usually neglected. To order the path candidates which can be of different lengths, a (heuristic) function that underestimates the cost of the remaining portion of the path is required, which is difficult to achieve because the scores are always highly data dependent.

Therefore, we adhere to the time synchronous DP beam search and combine it with an efficient organization by using lists. The basic problem is how to perform the recombination of paths or in other words the DP recursion efficiently without an explicit construction of the overall search space. Even in the case of a pruned search, the direct application of the DP equations would require that each grid point be processed or at least checked as to whether it is still active. Thus there would be a computational overhead that is proportional to the number of grid points in the overall search space and independent of the number of active path hypotheses. Therefore, our goal is to arrive at an algorithm the time complexity of which depends only on the number of active word hypotheses.

We could try replacing the array of (j,k)-grid points (for a given frame i) by a list. But then the list would have to be searched for each new grid point to check for recombination. This is prohibitive since the number of path hypotheses or active grid points may be in the range of 10000. Instead we use arrays that can be explicitly addressed to perform the recombination of path hypotheses. There are two types of recombinations which take place either in the word-interior time alignment or at syntax nodes according to the language constraints. Therefore we introduce two arrays. The first array is an auxiliary array that is used to perform the word-interior time alignment and is indexed by state index j for a given word. The second array is addressed by the node index and holds the acoustic score along with the backpointer of each syntax node. The entries of these arrays are initialized with infinitely large values and are changed if there is an active hypothesis.

To limit the search to 'active' grid points, we introduce lists at several levels. The lists have two functions: they are used to store the information about the path hypotheses and to guide the search at the same time. There are three lists: a list of syntax nodes, a list of words and a list of grid points or path hypotheses. Fig. 1 shows the list of words and the list of path hypotheses. An entry l in the list of words consists of the word index k(l) and two pointers to the list of path hypotheses. These pointers mark the range within which the path hypotheses of the word under consideration are stored. An entry in the list of grid points consist of state index j, score S(i,j,k(l)) and backpointer B(i,j,k(l)).

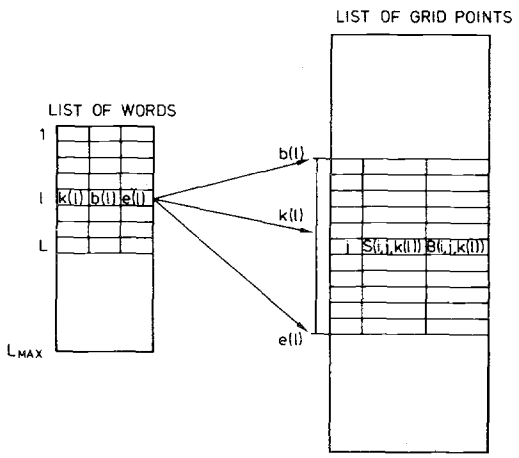

Fig. 1: Illustration of the list of words and the list of grid points.

A flow diagram of the data-driven organization of the search is given in Fig. 2. Fig. 2 shows the knowledge sources on the left-hand side, the processes in the middle and built-up data structures on the right-hand side. These data structures serve both as storage and as guidance for the search process. There are three basic processing steps. In the first step, which processes the list of syntax nodes, the traceback information is stored and the information about the words that leave the syntax node is added both to the list of words and the list of grid points. In the second step, the words on the list of words are processed by performing the word interior time alignment. The result is a temporary list of new generated grid points that is processed in the third step. At this step, a number of additional operations are performed: the pruning of grid points, the check for word boundaries and for

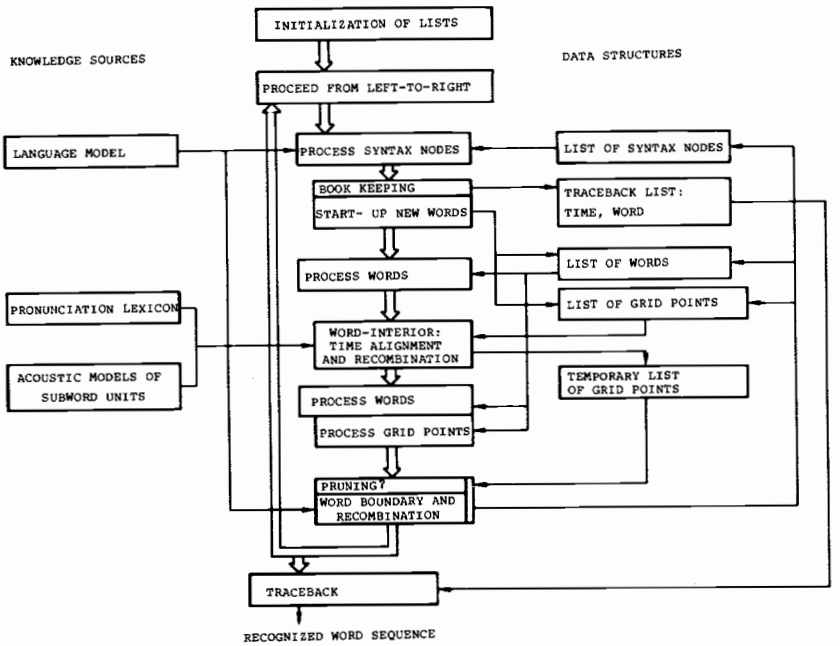

Fig. 2: Flow diagram of the search organization.

recombination at syntax nodes and the up-date of the three lists of syntax nodes, words and grid points. The details of the time alignment are given in Fig.3 The double-boxed array and list are used as temporary storage. During the time alignment, a list of successor points is generated. The recombination of path hypotheses is performed by using the grid lattice array and checking whether the successor point under consideration is active, i.e. whether its score in the grid lattice array is smaller than infinite. Each time a word has been processed the active grid points in the grid lattice array are reset to infinite.

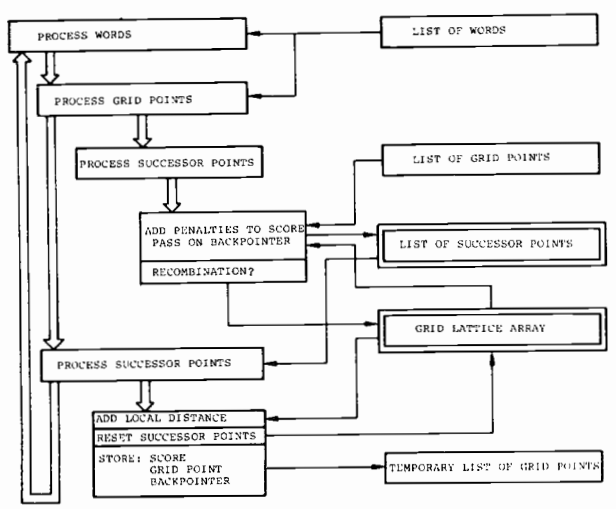

Fig. 3: Flow diagram of the time alignment.

4. EXPERIMENTAL RESULTS

Experimental tests were run on a German speech data base spoken by six speakers. For each speaker, there were three or two (speakers F02 and M10) sessions in each of which the 200 'SPICOS' sentences [9] were spoken in a continuous manner, i.e. with no pauses between the words. The recognition tests were run on 188 of the 200 'SPICOS' sentences, which results in 1292 word recognition tests for each session of each speaker. The task and the language models used are described in more detail in [10]. The training of the speaker dependent phoneme models is described in [9]. The training vocabulary is different from the test vocabulary. The recognized word sequence is automatically compared with the correct word sequence to determine the number of confused, deleted and inserted words. The sum of confusions, deletions and insertions is defined as the word error rate. To measure recognition errors caused by the German-typical word endings, we use a pragmatic definition of a stem error rate [9]. In addition, we give the average number of hypotheses per 10-ms frame of the input utterance for each recognition experiment. For the same speaker, different error rates may be given because the recognition results depend on a number of parameters and in particular on the test session.

Table 1 shows the recognition results of one session of speaker M03 for three different language models. As can be expected, the error rate does not depend very much on the number of word transitions in the language model. The pruning threshold was chosen by trial and error. Fig. 4 shows the word error rate as a function of the number of hypotheses per 10-ms for speaker M03 and language models NET3500 and NET11000. For more than 3000 hypotheses, there is saturation in the recognition performance. Thus no significant improvement can be expected by increasing the number of hypotheses. For language model NET3500, there is a total of 3481*50 grid points for each 10-ms if we assume an average length of 50 10-ms frames for each word. Thus 4000 hypotheses means that only 2.2 % of the total of the possible hypotheses are evaluated. Nevertheless the computational cost is about 50 MIPS for processing 1 sec of speech.

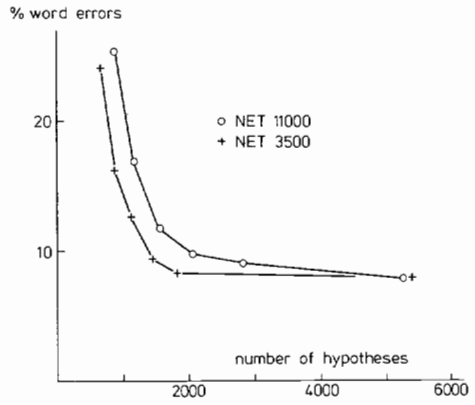

Fig. 4: Word error rate as a function of the average number of hypotheses for session 2 of speaker M03 and two language models.

In order to check whether the globally optimal path was lost during the search, an additional experiment was carried out. For each sentence to be recognized, the score of the path of the correct word sequence was computed and compared with the score of the path found by the search procedure. Table 2 gives these results for the 6 speakers. The column 'Path Missed' gives the number of cases where the score of the correct word sequence was better and therefore the optimal must have been missed. E.g. for speaker F02, the optimal path was not found in 3 out of 376 sentences. On the average, only for 0.5% of the test sentences was it sure that the best path had been missed.

When discussing these recognition experiments, we have not been concerned with the acoustic models and the error rate as such. Nevertheless it should be stressed that the training and test vocabularies were different, that a standard dictionary pronunciation with no variants was used and that the phoneme units were modelled by comparatively simple, stationary models.

5. SUMMARY

In this paper, we have presented a list organization of the DP beam search for large vocabulary, continuous speech recognition. The search is organized in such a way that the computational requirements are determined by the portion of the search space actually searched and are independent of the overall size of the search space. We have studied this search organization in a number of recognition experiments on a speech data base that comprises a total of 20672 words spoken by 6 speakers.

ACKNOWLEDGMENT

The work described was carried out in a joint Siemens-Philips-IPO(Eindhoven) project ('SPICOS') and sponsored by the German Federal Ministery for Research and Technology (BMFT) under grant No. 413-5839-ITM 8401. Only the authors are responsible for the contents of this publication.

Table 1: Effect of different language models on the number of hypotheses and the error rate for session 2 of speaker M03.

LANGUAGE MODELS	NET3500	NET11000	TREE
WORD TRANSITIONS	3481	10952	28101
EMPTY TRANSITIONS	413	1026	0
SYNTAX NODES	368	944	398
PERPLEXITY	58	73	66
HYPOTHESES/10-MS	3100	3300	3800
WORD ERRORS [%]	10.1	13.4	12.7
STEM ERRORS [%]	5.8	8.2	3.7

Table 2: Number of missed paths for language model NET3500.

SPEAKER	HYPOTH./10-MS	ERROR RATES[%]	PATH MISSED
F01	13900	20.7/ 17.8	2/564
F02	6000	9.9/ 8.3	3/376
M01	7600	15.6/ 13.3	9/564
M02	7400	13.1/ 10.9	0/564
M03	8800	7.8/ 6.8	0/564
M10	8600	14.7/ 12.5	3/376
AVERAGE			17/3196 = 0.57 %

REFERENCES

[1] S.E. LEVINSON: "Structural Methods in Automatic Speech Recognition", Proc. of the IEEE, Vol.73, No.11, pp. 1625-1650, Nov. 1985.

[2] J.S. BRIDLE, M.D. BROWN, R.M. CHAMBERLAIN: "An Algorithm for Connected Word Recognition", Proc. 1982 IEEE Conf. on Acoustics, Speech and Signal Processing, Paris, France, pp. 899-902, May 1982.

[3] 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.

[4] J.C. SPOHRER, P.F. BROWN, P.H. HOCHSCHILD, J.K. BAKER: "Partial Traceback in Continuous Speech Recognition", Proc. Int. Conf. on Cybernetics and Society, Boston, MA, pp. 36-42, Oct. 1980.

[5] 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. PAMI-5, No.2, pp. 179-190, March 1983.

[6] B. LOWERRE, R. REDDY: "The Harpy Speech Understanding System", in W. A. Lea, Editor: Trends in Speech Recognition, pp. 340-360, Prentice-Hall, Englewood Cliffs, New Jersey, 1980.

[7] N. J. NILSSON: Principles of Artificial Intelligence. Tioga Publishing Company, Palo Alto, CA,1980.

[8] M. K. BROWN, L. R. RABINER: "An Adaptive, Ordered, Graph Search Technique for Dynamic Time Warping for Isolated Word Recognition", IEEE Transaction on Acoustics, Speech and Signal Processing, Vol. ASSP-30, No. 4, pp. 535-543, August 1982.

[9] A. NOLL, H. NEY: "Training of Phoneme Models in a Sentence Recognition System", Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing, Dallas, TX, April 1987.

[10] D. MERGEL, A. PAESELER: "Construction of Language Models for Spoken Database Queries", Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing, Dallas, TX, April 1987.