S10.1 A NETWORK-BASED FRAME-SYNCHRONOUS LEVEL BUILDING ALGORITHM FOR CONNECTED WORD RECOGNITION

Chin-Hui Lee and Lawrence R. Rabiner

Speech Research Department
AT&T Bell Laboratories
Murray Hill, NJ 07974

ABSTRACT

In this paper, a network-based, frame-synchronous, level building (FSLB) algorithm is described for recognizing continuous speech as a connected sequence of words. Previous frame synchronous approaches to recognition have included the one-pass approach, the one-stage approach, and an unpublished multi-level approach. The proposed algorithm, which has all the features of these earlier methods, as well as several new features, and which is implemented in a network-based approach, is a numerically exact, frame synchronous, implementation of the conventional level building (LB) algorithm. As with some of the earlier methods, the proposed algorithm is highly regular and modularized for distributed computation among several special purpose processors. New features of the algorithm include the capability of determining the best alternative recognition strings (e.g. second and third best strings), at every level, even for very complicated grammar networks, and the capability of efficiently incorporating several word and state duration scoring techniques directly in the forward search, thereby eliminating the need for a postprocessor as required in the direct LB implementation. Word transition rules (e.g. a language model) can also be easily incorporated into the proposed algorithm.

1. INTRODUCTION

The problem of recognizing a fluently spoken sentence (or string of words or subword units) based on concatenating individual word models is extremely important for automatic speech recognition tasks. A wide variety of approaches this problem, all based on the technique of dynamic programming (DP) [1], have been proposed and evaluated [2-8]. The earliest algorithm for connected word recognition was proposed by Vintsyuk [2] who showed how DP techniques could be used to get the optimal sequence of words which match a spoken input. Vintsyuk's procedure processed the speech signal in a frame-synchronous manner, and therefore his pioneering work formed the basis for several DP-based solutions to the speech recognition problems. Vintsyuk also proposed a rudimentary scheme for incorporating syntactic constraints among words in the search (i.e. a grammar).

Since Vintsyuk's work was largely unknown in the US and Japan, two different DP-based search structures were proposed for solving the connected word recognition problem, namely the two-level DP match approach of Sakoe [3], and the level building approach of Myers and Rabiner [4]. These approaches differ from Vintsyuk's method in their flexibility and in the computation and storage requirements, but fundamentally they both were capable of finding the optimal match to a spkoen word string. A rediscovery of Vintsyuk's method was made by Bridle *et al* [5], who proposed a variation on the frame-synchronous method which was subsequently called the one-pass DP approach, and by Ney [6] who proposed the one-stage DP approach. The major innovation in these approaches was the way in which a grammar was integrated into the search procedure. Another innovation was the incorporation of multiple levels into the search so that a given input utterance could be decoded into "optimal" strings of different length (i.e. number of words in the string). Incorporation of a grammar into the level building algorithm was also demonstrated by Myers and Levinson [7]. Most recently, Glinski [8] proposed a frame-synchronous algorithm which could be used with either templates or hidden Markov models (HMM) [9], and which had all the features of the Bridle *et al* and Ney approaches, and it also had the capability of handling mutiple levels in an appropriate manner. A special purpose chip, called the Graph Search Machine (GSM) [8] was designed and shown capable of performing the search operations required in the DP solution.

In this paper we propose a frame-synchronous level building (FSLB) algorithm that preserves *all* the properties of the original level building approach, and that is applicable to solutions based on using either HMM's or templates. At issue here is proper incorporation of word and state duration constraints, as well as proper extraction of multiple candidate strings. None of the earlier DP-based search algorithms handled these problems in an appropriate manner. The proposed FSLB algorithm is also capable of handling word (or subword) transition rules, e.g. a language model, directly in the forward search part of the procedure. There are several ways in which state durational constraints have been incorporated into the HMM scoring, namely Levinson's continuously variable duration HMM [10], and hidden semi-Markov models and expanded state HMM's [11]. Since these explicit duration models significantly increase the computation and storage associated with HMM scoring, a simple alternative is to account for state duration in a post-processor after the level building search is completed [12]. The level building algorithm in [12] not only deals with word and state duration scoring strategies but also handles multiple candidate strings effectively. The technique is not guaranteed to generate the best Q candidate strings; however, it does give a reasonable list of candidates, including the best candidate for every possible string length. The FSLB algorithm proposed here handles both word and state duration constraints properly (in the forward search) and generates the best Q candidate strings at all levels with essentially minimal computation cost.

As part of the development of the FSLB algorithm, we review the similarity between the conventional level building algorithm and the frame-synchronous DP approaches [2, 5, 6, 8] by recasting the speech recognition task as a network search problem. We then present a unified approach to solving speech recognition by using optimal finite-state network (FSN) decoding, and show that the word and state duration scoring schemes in the level building algorithm can be implemented exactly by assigning the appropriate cost to arcs and nodes in the network search.

2. SPEECH RECOGNITION VIA FSN DECODING

As pointed out in [13], most speech recognition tasks can be organized into a hierarchy of networks with a finite number of nodes and arcs corresponding to acoustic, phonetic and syntactic knowledge sources and their interactions. Recognition of a spoken utterance corresponds to finding an optimal path through the finite state network. The idea is applicable for both isolated word recognition and continuous speech recognition [14-15]. The optimal network search can be accomplished by sequential decoding using Dynamic Programming (DP) based on a simple

CH2561-9/88/0000-0410 $1.00 © 1988 IEEE

concept. The concept was stated by Bellman [1] as the principle of optimality in the following terms: "An optimal set of decisions has the property that whatever the first decision is, the remaining decisions must be optimal with respect to the outcome of the first decision." In terms of decoding optimal paths in a finite state network, the principle of optimality enables the decoding to be performed on a frame-by-frame basis, as long as all the information required for the local optimal paths are kept so that the global optimal paths can be found based on the local ones.

For a connected word recognition task, it is instructive to decompose the network into two levels, namely a phrase (grammar) level and an intra-word level. The intra-word level is usually a word model, which could be a whole word template or a hidden Markov model for a word. In this paper, we will focus our attention on the latter. The intra-word nodes are essentially the HMM states, while the intra-word arcs represent state transitions. For a left-to-right HMM, the intra-word node can be reached from only a small number of predecessor intra-word states. In general, the intra-word level uses a sparse network representation for most recognition tasks. As for the inter-word level, it is simply represented by a grammar network, in which the nodes represent level boundaries, as in the conventional LB algorithm, and the arcs represent word models and word transitions. This grammar level network representations range from simple networks with few syntactic constraints to highly constrained, complicated grammar networks.

We will now describe how to associate a stochastic cost (penalty) to different parts of an FSN so that optimal FSN decoding can be performed based on those stochastic penalties. In a speech recognition task, in which the word models are characterized by an HMM, the accumulated cost of a path to any node in the FSN at time t can be defined as the negative of the accumulated likelihood of the path at time t, where likelihood is defined the logarithm of the probability of that path. The cost of staying in an internal state at time t is related to the probability of observing the feature vector in that state at time t, and can be defined as the negative of the logarithm of the state observation probability. The cost of making an internal transition includes the negative of the logarithm of the transition probability, plus some possible state duration penalty. The cost of entering the right grammar node of a grammar arc includes a possible state and word duration penalty. Finally, the cost of leaving the left grammar node of a grammar arc includes a possible word transition penalty. With all the costs assigned properly, the search for the best path in an FSN is essentially the same as finding the minimum cost path through the network or equivalently performing the maximum likelihood network decoding.

3. AN FSLB ALGORITHM FOR SPEECH RECOGNITION

We have discussed several ways to map a speech recognition task into a network decoding problem. The LB algorithm [12] and the FSLB algorithm proposed here are fundamentally identical in that they are solving the same network optimization problem, i.e. finding among all possible segmentations of the input utterance, the most likely state and string sequence that satisfies a given network syntactic constraint. The search strategies are the same at the intra-word level, while the difference lies in the grammar level search. The LB algorithm searches for all possible optimal paths on a level-by-level basis, that is it finds, for all possible ending times, the optimal path at one level: it then uses the set of all possible level ending times from the previous level, as the set of possible starting times of the next level, in order to build optimal paths for the higher levels at all possible ending times. Therefore the computation wavefront progresses from one level to the next. On the other hand, the computation wavefront for the FSLB algorithm progresses on a frame-by-frame basis, i.e. to search for optimal paths to all possible level boundaries at each frame. Since both the LB and FSLB algorithms attempt to solve the same optimization problem efficiently, the amount of search computation is similar; the key difference lies in the way the search computation wavefront progresses, and therefore the storage requirements and the memory management strategies are vastly different.

The FSLB algorithm proposed here is essentially a frame-synchronous implementation of the LB algorithm described in [12], and the optimal search is accomplished by incorporating all the state and word duration scoring schemes on a frame-by-frame basis. Since the optimal path at one level is based on optimal paths to the previous levels, it suffices to solve the level building optimization problem frame-synchronously by searching for optimal paths to all level boundaries at any time t, which in turn can be solved by searching for all optimal paths to all internal nodes between subsequent level boundaries at any time t. Therefore, for each node in the FSN, at any time t, the FSLB searches for the best path arriving at that node at that time, and constructs the optimal path of duration t to that node from all the best paths of duration $(t-1)$. The principle of optimality allows the best path to any node j, at time t, to be determined from the best paths to all nodes i, at time $(t-1)$, plus the best policies (transitions from node i to node j) at time t. Parameters needed for computing the best paths include the accumulated likelihood to any node on the FSN, the path information such as source and destination grammar nodes for each grammar arc on the path, and word and state durations required in order to compute duration probabilities. Based on the above, we now present a grammar-driven FSLB recognition algorithm.

A Syntax-Directed FSLB Algorithm

Step 1. Perform initialization for all nodes and traceback buffers
Step 2. Perform optimal path building frame-by-frame
  For every input speech frame (while not end of utterance)
    Perform feature extraction (**FE**)
    Compute local likelihoods (**LL**) for all states
    For every grammar node in the network (level loop)
      For every predecessor grammar node
        For every arc between them (word model loop)
          Perform local DP for every internal node (**DP1**)
          Update path likelihood and path information
      Perform path merging DP (**DP2**)
      Update accumulated path likelihood and path information
      Update accumulated traceback buffers
Step 3. Perform post-processing and backtracking (**PP/BT**)
  For every terminal node
    Perform duration scoring post-processing
    Perform traceback to identify recognized string
Step 4. Determine the recognized string

The modules of the FSLB algorithm are arranged in the block diagram shown in Fig. 1. Modules FE, LL, PP and BT are essentially the same as those in the LB algorithm. The "Update" module uses the local likelihoods computed in the LL module to update the accumulated likelihoods of the optimal paths to all the internal nodes. We will now describe modules DP1 and DP2 in more detail in the following sections.

3.1 A Modified Intra-word Viterbi Decoding Algorithm

Conventional Viterbi decoding algorithms for finding the best decoded sequence for a given hidden Markov model are readily available in the literature. The duration scoring strategy is often not incorporated into the algorithm. We will now modify the Viterbi decoding algorithm to include all possible state and word duration probabilities and word transition probabilities so that all the probabilities can be incorporated in the forward search scheme either before or after paths are merged at the nodes of the FSN. Depending on the desired duration scoring strategies, we can simply associate proper cost to the arcs of the word-level HMM network. The post-processing duration scoring scheme in [12] can be accomplished in the FSLB by associating a word duration penalty for transversing the arc connecting the rightmost node of the word model and the right grammar node (RG) in the forward search and by assigning a state duration penalty to the same arc and incorporating the state duration penalty at the end of the utterance for post-processing. We will discuss, in more detail, various ways of incorporating duration information into the FSLB algorithm in Section 5. As for the word transition penalty, it can be based on some simple word transition rules or a complex language model [15]. The word transition penalty can be easily incorporated by associating a penalty for transversing the arc connecting the left grammar node (LG) and the leftmost internal node of a word model. This cost

depends on the previously decoded words on the best path entering the left grammar node.

3.2 A Grammar Level Path Merging Algorithm

Grammar nodes on a network allow paths reaching that node to merge so that only a limited number of the paths are allowed to grow to the next level. They also provide branching to the higher levels. If only the optimal path is desired at each of the terminal nodes of a grammar network, then path merging at a grammar node eliminates all non-optimal paths reaching that node. The grammar level path merging algorithm proposed here is similar to eq.(11) in [12] where an optimal path is selected at a level boundary. For a general grammar network, we not only need a backpointer array to indicate how long the best path reaching node g has stayed in the best arc, we will also need a node backpointer array to indicate the left grammar node of the best arc reaching that grammar node.

4. SEARCH FOR THE GLOBALLY SECOND BEST PATHS

By definition, the globally second best path to any grammar node, in a finite state network, is different from the globally best path to that node, in the sense that the decoded strings should be different. Since the search is only focused on the grammar node level, rather than on the internal node level, there is very little additional computation required. Similar to the search strategy for the optimal path, searching for the second best path to any grammar node can be accomplished sequentially by computing the second best path to that grammar node at any time t. The second best path information required is also similar to what is needed for the optimal path.

The algorithm proposed here to search for the globally second best paths to every grammar node in the finite state network can be formulated based on a concept extended from the principle of optimality. It states that the second best path to any grammar node g, at time t, is either path $P_{12}(g)$ or path $P_{21}(g)$ defined as follows. We define 1(g) and 2(g) to be the globally best and second best paths reaching grammar node g, and 1′(g) and 2′(g) to be the locally best and second best subpaths (arcs) on the locally best and second best paths reaching grammar node g. From the principle of optimality, the best path is composed of the locally best subpath 1′(g)=a of duration $bp(a)$ and the best path 1(i) reaching the left grammar node i=LG(a) of arc a at time $t(i)=t-elapse(a)$. Whereas the candidate path $P_{12}(g)$ is composed of the locally second best subpath 2′(g)=b of duration $elapse(b)$ and the best path 1(j) reaching the left grammar node j=LG(b) of arc b at time $t(j)=t-elapse(b)$; the candidate path $P_{21}(g)$ is composed of the locally best subpath 1′(g) and the second best path 2(i) reaching grammar node i at time $t(i)$. Nodes i and j can be the same, and word durations $elapse(a)$ and $elapse(b)$ can also be the same; however the arcs a and b should be different to guarantee the complete second best string be different from the best string.

In the FSLB algorithm proposed in Section 3, we only allow the optimal paths at time $(t-1)$ to be used in constructing the optimal paths at time t. If we also allow other candidate paths to construct paths in the next level, as was performed in the LB algorithm to obtain multiple candidate strings, then the amount of search will increase proportional to the number of candidate paths retained. It turns out that $P_{12}(g)$ is the second best path among all other paths if paths can only be constructed from the local best ones. Similarly, $P_{21}(g)$ is the second best path if other candidate paths are allowed to grow before they are pruned when merging at the grammar node LG(a) at time $t-elapse(a)$. The algorithm proposed here extends the property of the principle of optimality in that only paths $P_{12}(g)$ and $P_{21}(g)$ are compared. As a result, the algorithm requires no additional search computation, and only some additional bookkeeping is required. The traceback information and duration probabilities of the globally second best paths are managed similar to that of the globally best paths.

5. INCORPORATION OF DURATION PROBABILITIES

When intra-word state duration is properly incorporated, the HMM duration models in [10-11] seem to improve performance on some recognition applications. Since those explicit duration models significantly increase the computation and storage associated with scoring, a simple alternative is to account for the state duration in a post-processor after the level building search is completed [12]. In the following, we focus our discussion on how to incorporate both the word and state duration probabilities into the HMM scoring and network search. The reader is referred to the modeling approaches in [10-12] to learn how to model the word and state durations and how to assign weights when these duration penalties are incorporated in the HMM scoring.

We have shown in Section 2 how duration penalties can be represented in the network search by associating proper cost to arcs on a finite state network. In Section 3.1 we have also formulated a general strategy showing how the duration penalties can be incorporated in the modified Viterbi decoding algorithm. The way duration penalties are incorporated depends on the HMM scoring strategy in the network search. If they are to be used before the paths merge at a certain node in the network, then they should be added to the path likelihoods before the paths are compared and pruned. If the scoring scheme requires the duration penalties to be included only at the end of the utterance, then a separate buffer is needed to save the additional duration information so that they can be imposed by a post-processor.

The word duration probability is usually included at the end of a word (strategy W1); therefore candidate paths with unlikely word durations are more likely to get pruned even when the path likelihoods are reasonable. However, we can impose the word duration at the end of an utterance and use a post-processor to handle the additional information before the best candidate path is selected (strategy W2). We can also use normalized word durations or normalized accumulated word durations in the scoring or not use any word duration information at all (strategy W0).

The state duration probability can be included at the end of a state (strategy S1), at the end of a word (Strategy S2), or at the end of an utterance (Strategy S3) by using a post-processor. We also can use either normalized state durations within a word or normalized accumulated state durations within a word. We can also avoid using the state duration information (Strategy S0).

In the following, we compare performance using different combinations of the word and state duration scoring strategies. The task involved is speaker-independent connected digit recognition. The database used in the evaluation is the TI speaker-independent connected digit database. The feature vector used in the experiment is a combination of 12 weighted LPC cepstral coefficients [16], and 12 delta-cepstral coefficients [17]. The state observation densities are characterized by a multivariate mixture Gaussian density [18]. The HMM training is described in more detail in [12], where a segmental k-means algorithm was used to estimate the HMM parameters via a k-means clustering techniques. In this specific experiment, the test vocabulary consisted of 11 words, namely 10 digits plus the word "oh". To characterize each word, we use 3 models per word, 10 states per model, and 9 mixtures per state.

Six different duration scoring strategies were tested. These included the following: (1) W0+S0, where no duration information was used; (2) W1+S0, where state duration was not used while word duration was scored at the end of each word; (3) W0+S2, where word duration was not used but state duration was used at the end of each word; (4) W1+S3, the current post-processing strategy in [9]; (5) W1+S2, where both word and state durations were imposed at the end of each word; and (6) W1+S1, where the word duration was incorporated at the end of each word and state duration was incorporated at the end of each state. The test results are listed in Table 1, where 8,578 variable length strings were tested. The results in the row labelled UL shows the string error rates if the length of the tested string length is unknown, and the results in the row labelled KL gives the string error rate if we know the length of the tested string. As shown in Table 1, there is only a slight statistical difference among all the duration scoring strategies tested.

	W0+S0	W1+S0	W0+S2	W1+S3	W1+S2	W1+S1
UL	3.28	3.21	3.15	3.12	3.12	3.21
KL	1.81	1.71	1.68	1.64	1.65	1.69

Table 1. String error rates for various duration scoring configurations

6. CONCLUSION

In this paper, a frame-synchronous level building (FSLB) algorithm is proposed for recognizing connected sequences of words. The algorithm is essentially a frame-synchronous implementation of the conventional level building (LB) algorithm and therefore the FSLB algorithm yields exactly the same numerical results as the LB algorithm. However, in contrast to the LB algorithm which decodes optimal strings at the end of an utterance on a level-by-level basis, the FSLB algorithm performs the maximum likelihood string decoding in a network on a frame-by-frame basis so that the optimally decoded partial strings are immediately available at any time. The FSLB algorithm also offers a number of advantages over the LB algorithm, namely: (1) The FSLB algorithm is highly regular, which makes it easy to implement on general or special purpose hardware; (2) the FSLB algorithm is highly modularized so that the local likelihood computation and the local Viterbi decoding for each word model can be performed in parallel, which makes it attractive for distributed computation among multiple special-purpose processors; (3) for decoding a given finite state grammar network, at time t, the FSLB algorithm uses only the information about the optimal paths at time $(t-1)$ plus the observation vector at time t to search for the optimal paths and to update the accumulated path likelihood so that the memory requirement is greatly reduced; and (4) multiple candidate strings, e.g. the globally second best strings, can be obtained easily even for a very complicated grammar network. In addition to the simplified search strategy, the FSLB algorithm also has the flexibility that all the word and state duration scoring techniques, e.g. the state duration postprocessing model used in the LB algorithm, can be implemented simply, efficiently and directly in the forward search. Word transition rules, such as a language model, can also be easily incorporated.

REFERENCES

[1] R. Bellman, *Dynamic Programming*, Princeton Univ. Press, Princeton, NJ, 1957.

[2] T. K. Vintsyuk, "Element-wise Recognition of Continuous Speech Composed of Words from a Specified Dictionary," *Kibernetika*, Vol. 7, pp133-143, Mar.-Apr. 1971.

[3] H. Sakoe, "Two-Level DP-Matching - A Dynamic Programming-Based Pattern Matching Algorithm for Connected Word Recognition," Trans. ASSP-27, No.6, pp588-595, Dec., 1979.

[4] C. S. Myers and L. R. Rabiner, "A Level Building Dynamic Time Warping Algorithm for Connected Word Recognition," Trans. ASSP-29, No.2, pp284-297, April, 1981.

[5] J. S. Bridle, M. D. Brown and R. M. Chamberlain, "An Algorithm for Connected Word Recognition," *Proc. ICASSP* 82, pp899-902, Paris, May, 1982.

[6] H. Ney, "The Use of a One-Stage Dynamic Programming Algorithm for Connected Word Recognition," Trans. ASSP-32, No.2, pp263-271, April, 1984.

[7] C. S. Myers and S. E. Levinson, "Speaker-Independent Connected Word Recognition Using a Syntax-Directed Dynamic Programming Procedure," Trans. ASSP-30, No.4, pp561-565, August, 1982.

[8] S. Glinski, et al., "The Graph Search Machine (GSM): A Programmable Processor For Connected Word Recognition and Other Applications," *Proc. ICASSP* 87, pp519-522, Dallas, 1987.

[9] L. R. Rabiner and B.-H. Juang, "An Introduction to Hidden Markov Models," *IEEE ASSP Magazine*, Vol.3, No.1, pp4-16, Jan. 1986.

[10] S. E. Levinson, "Continuously Variable Duration Hidden Markov Models for Automatic Speech Recognition," *Computer, Speech, and Language*, Vol. 1, No.1, pp. 29-45, March, 1986.

[11] M. J. Russell and A. E. Cook, "Experimental Evaluation of Duration Modelling Techniques for Automatic Speech Recognition," *Proc. ICASSP* 87, pp101-104, Dallas, 1987.

[12] L. R. Rabiner, J. G. Wilpon and B.-H. Juang, "A Model-based Connected-Digit Recognition System Using Either Hidden Markov Models or Templates," *Computer, Speech, and Language*, Vol. 1, No. 2, pp. 167-197, December, 1986.

[13] J. K. Baker, "The Dragon System - An Overview," IEEE Trans. ASSP-23, No.1, pp24-29, February, 1975.

[14] P. F. Brown, J. C. Spohrer, P. H. Hochschild and J. K. Baker, "Partial Traceback and Dynamic Programming," *Proc. ICASSP* 82, pp1629-1632, Paris, May, 1982.

[15] L. R. Bahl, F. Jelinek and R. L. Mercer, "A Maximum Likelihood Approach to Continuous Speech Recognition," Trans. Pattern Analysis and Machine Intelligence, Vol. PAMI-5, No.2, pp179-190, March, 1983.

[16] B.-H. Juang, L. R. Rabiner and J. G. Wilpon, "On the Use of Bandpass Liftering in Speech Recognition," Trans. ASSP-35, No.7, pp947-954, July, 1987.

[17] F. K. Soong and A. E. Rosenberg, "On the Use of Instantaneous and Transitional Spectral Information in Speaker Recognition," *Proc. ICASSP* 86, pp877-880, Tokyo, 1986.

[18] B.-H. Juang and L. R. Rabiner, "Mixture Autoregressive Hidden Markov Models for Speech Signals," Trans. ASSP-33, No.6, pp1404-1413, Dec. 1985.

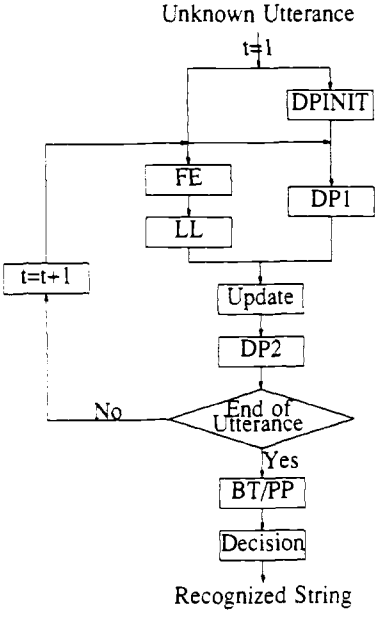

Figure 1. A block diagram of the FSLB recognition algorithm