PROCEEDINGS

ICASSP 91

1991 INTERNATIONAL CONFERENCE ON
ACOUSTICS, SPEECH, AND SIGNAL PROCESSING
MAY 14-17, 1991

IEEE SIGNAL
PROCESSING SOCIETY

IEEE
91CH2977-7

VOLUME 1:

S_1 SPEECH PROCESSING 1

Lennig

ICASSP 91

VOLUME 1

S_1

SPEECH PROCESSING 1

1991
INTERNATIONAL CONFERENCE
ON
ACOUSTICS, SPEECH, AND
SIGNAL PROCESSING

MAY 14-17, 1991
THE SHERATON CENTRE HOTEL & TOWERS
TORONTO, ONTARIO, CANADA

IEEE
91CH2977-7

Sponsored by
THE INSTITUTE OF ELECTRICAL
AND ELECTRONICS ENGINEERS,
SIGNAL PROCESSING SOCIETY

A* - admissible heuristics for rapid lexical access[1]

P. Kenny, R. Hollan, V. Gupta[2], M. Lennig[2], P. Mermelstein[2] and D. O'Shaughnessy
INRS-Télécommunications, Montreal, Quebec, Canada

Abstract

We present a new class of A algorithms for Viterbi phonetic decoding subject to lexical constraints. We show that this type of algorithm can be made to run substantially faster than the Viterbi algorithm in an isolated word recognizer having a vocabulary of 1,600 words and that it runs very quickly on a 60,000-word recognition task. In addition, multiple recognition hypotheses can be generated on demand and the search can be constrained to respect conditions on phone durations in such a way that computational requirements are substantially reduced.*

I. Introduction

The most widely used decoding strategy in speech recognition is the Viterbi algorithm [4]. This algorithm suffers from several disadvantages: it is an exhaustive search, it generates only one recogntion hypothesis (but see [1]) and it does not respect minimum duration constraints on phones (but see [2]).

The principal alternative to the Viterbi algorithm is A* or stack decoding. This type of algorithm was applied to speech recognition by IBM in the 1970's, using the formula dictated by the Bayes decision rule to score recognition hypotheses [4], but in the absence of a good heuristic to evaluate partial hypotheses the Viterbi algorithm has generally been found to be more efficient.

In this paper we will present a new A* algorithm for isolated word recognition *using the Viterbi scoring procedure to evaluate recognition hypotheses* and a class of heuristic functions which allows for a trade-off between the efficiency of the search and the burden of evaluating the heuristic scores. Since the search is guided by an A*-admissible heuristic, it is capable of generating multiple recognition hypotheses on demand. Furthermore, using the Viterbi scoring criterion enables us to incorporate phone duration constraints in a natural way — their use greatly improves the speed of the search. We have tested two versions of the algorithm on an isolated word recognition task with a 1,600 word vocabulary and found that it runs considerably faster than the Viterbi algorithm. We have also found that it runs very quickly on a 60,000-word recognition task.

II. Markov models and phonetic graphs

In Markov modelling, a speech segment is viewed as a doubly stochastic process $(Y_1, \ldots, Y_T, s_0, \ldots, s_T)$; $Y_1, \ldots, Y_T$ is a sequence of acoustic observations and $s_0, \ldots, s_T$ is a Markov chain. Corresponding to each transition in the Markov chain there is a distribution on acoustic observations. For each time $t = 1, \ldots, T$, Y_t is assumed to be generated by random sampling from the distribution associated with the transition $s_{t-1} \rightarrow s_t$. In the general case, there is a starting distribution π such that s_0 is determined by sampling from π. States s which have the property that $\pi(s)$ is positive are known as source states. The state sequence (and hence the observation sequence) terminates when a sink state is reached. Sink states are distinguished by the fact that the probability distributions on transitions out of these states are undefined.

If s is a state of a HMM M and $t = 0, \ldots T$ we define the forward probability $\alpha_t(s)$ to be the joint likelihood $P(Y_1, \ldots, Y_t, s_0, \ldots, s_t)$ where $s_0, \ldots, s_t$ is the sequence of states such that s_0 is a source and s_t is s for which this joint likelihood is maximal. The backward probability $\beta_t(s)$ is defined to be the joint likelihood $P(Y_{t+1}, \ldots, Y_T, s_{t+1}, \ldots, s_T)$ where $s_t, \ldots, s_T$ is the sequence of states such that s_t is s and s_T is a sink for which this joint likelihood is maximal. Both the forward and backward probabilities can be calculated by the Viterbi algorithm. The Viterbi score of the complete utterance $Y_1, \ldots, Y_T$, which we denote by $V(Y_1, \ldots, Y_T|M)$, can be evaluated either by maximizing $\alpha_T(s)$ over all sink states or by maximizing $\beta_0(s)$ over all source states.

We assume that each phone model has a single source and a single sink. Furthermore we will assume that, in each phone model, the source communicates with just one state which we refer to as the initial state and that there is just one state which communicates with the sink; we refer to this state as the final state and we denote the probability of the transition from the final state to the sink in the model corresponding to a phone f by q_f. Two phone models are concatenated by identifying the sink state of the first model with the source state of the second.

For each time $t = 1, \ldots, T$ and each phone f we define an 'impulse response' function h_t^f on the interval $[t, T]$ by setting

$$h_t^f(t') = V(Y_t \ldots Y_{t'}|f)$$

for $t' = t, \ldots, T$. Note that this function can be evaluated for all t' in the interval $[t, T]$ using a single trellis calculation. If f is any phone other than the silence phone we impose maximum and minimum constraints on the duration of f by setting $h_t^f(t')$ to be 0 for all values of t' for which the duration (namely $t' - t + 1$) is inadmissible.

A phonetic graph G consists of a list of nodes and a list of branches. Each branch is a triple (n_1, f, n_2) where n_1 and n_2 are nodes (referred to as the starting node and the target node of the branch) and f is a phone label. A node n with the property that no branch has n as its target is called an initial node. If n has the property that no branch has n as its starting node then it is said to be a final node.

For example, a phonetic transcription $f_1 \ldots f_K$ is a graph having $K + 1$ nodes $n_0, \ldots, n_K$ whose branches are (n_{k-1}, f_k, n_k) for $k = 1, \ldots, K$. There is one initial node, namely n_0, and one final node, n_K. If $\mathbf{f}$ is a phonetic transcription of length K then we set $\mathbf{f}_k = f_1 \ldots f_k$ for $k = 1, \ldots, K$.

If G is a phonetic graph, then a path in G is a sequence of branches $b_1, \ldots, b_K$ having the property that the target node for each branch is the same as the starting node of the following branch. The path is said to be complete if the starting node of b_1 is an initial node and the target node of b_K is a final node. If we are given a lexicon L and a graph G, we say that G generates L if every complete path in G corresponds to a transcription in L and every transcription in the lexicon corresponds to a complete path.

An example of a graph that generates L is the lexical tree whose nodes are sequences of phones which have the property that they can be extended to complete transcriptions in L. If n_1 and n_2 are nodes and

[1] This work was supported by the National Sciences and Engineering Research Council of Canada

[2] also with Bell-Northern Research, Montreal, Canada

CH2977-7/91/0000-0689 $1.00 © 1991 IEEE

f is a phone label, then (n_1, f, n_2) is a branch of the lexical tree if the sequence n_2 is a one-phone extension of n_1 and the phone in question is f. There is one initial node (the empty phone sequence, also referred to as the root node) and the final nodes are phone sequences which are complete transcriptions in L (also referred to as the leaf nodes). The reverse lexical tree corresponding to L is defined similarly: its nodes correspond to sequences of phones which have the property that they can be extended backwards to complete transcriptions in L.

If we are given a graph G and an equivalence relation $\sim$ on the nodes of G we can construct a new graph G^*, called the quotient of G by $\sim$. The nodes of G^* are equivalence classes of nodes of G. A triple (n_1^*, f, n_2^*) is a branch in G^* if there is a branch (n_1, f, n_2) in G such that n_1^* is the equivalence class of n_1 and n_2^* is the equivalence class of n_2. Note that for every path in G there is a corresponding path in G^* whose branches carry the same phone labels. In particular if G generates a lexicon L, every complete transcription in L corresponds to a complete path in G^*. Generally speaking, G^* will overgenerate L but it is easy to construct examples where G^* does not overgenerate L and G^* contains fewer branches than G.

Suppose we are given a phonetic graph G together with a probability distribution π on the initial nodes and, for each node n which is not a final node, a probability distribution $P(\cdot|n)$ on the branches having n as their starting node. In this case we say that G has a probability structure. For example if G is the lexical tree corresponding to L, then there is a probability structure defined on G as follows. The distribution π is concentrated on the root node and probabilities are assigned to each of the branches of G in such a way that the prior probability of any transcription in L is the product of the probabilities on the branches in the corresponding path through G.

If G is a phonetic graph having a probability structure, we can associate a HMM M_G with it as follows. The states of M_G are defined by taking the nodes of G and, for each branch in G, taking a copy of the corresponding phone model with the source and sink states removed. The source states of M_G are the initial nodes of G and the starting distribution is π. The sink states are the final nodes of G.

If n is a node and s is the initial state on a branch b having n as its starting node and f as its phone label, then the probability of the transition $n \to s$ is $P(b|n)$. The distribution associated with the transition is the same as the distribution associated with the transition from the source to s in the f model. If n is a node and s is the final state on a branch b having n as its target node and f as its phone label, then the probability of the transition $s \to n$ is q_f. The distribution associated with the transition is the same as the distribution associated with the transition from s to the sink in the f model.

Now suppose we are given an utterance $Y_1, \ldots, Y_T$. We can find the complete transcription $\mathbf{f}$ in the lexicon generated by G for which

$$V(Y_1, \ldots, Y_T|\mathbf{f})P(\mathbf{f})$$

is maximal by using the Viterbi algorithm to search the Markov model M_G; $\mathbf{f}$ is called the Viterbi transcription of the utterance.

If G does not have a probability structure then the Viterbi transcription of the utterance is defined to be the complete transcription $\mathbf{f}$ in the lexicon generated by G for which

$$V(Y_1, \ldots, Y_T|\mathbf{f})$$

is maximal. This can also be found by the Viterbi algorithm if we set the 'probability' on each branch to be 1 and put $\pi(n) = 1$ if n is an initial node and 0 otherwise.

III. A* algorithms

We are given an utterance $Y_1, \ldots, Y_T$ and a lexicon L and we wish to find the complete transcription $\mathbf{f}$ in L for which the Viterbi score $V(Y_1, \ldots, Y_T|\mathbf{f})P(\mathbf{f})$ is maximal.

Generally speaking, an A* search of the lexicon proceeds as follows. At each iteration of the algorithm, there is a sorted list (or 'stack') of partial transcriptions each with a heuristic score. The partial transcription with the highest heuristic score is expanded, meaning that for each of the one-phone extensions permitted by the lexicon the heuristic score of the extended transcription is calculated and the extended transcription is inserted into the stack at the appropriate position. The algorithm terminates when a complete transcription appears on the top of the stack. If the heuristic satisfies a simple admissibility condition given below, then the entry on the top of the stack when the algorithm terminates is guaranteed to be an optimal complete transcription. Furthermore, if the algorithm is iterated beyond the point when the first optimal complete transcription is found, then each of the complete transcriptions in the lexicon will eventually appear on the top of the stack in order of decreasing likelihood. Hence multiple recognition hypotheses are obtained at little extra cost.

We associate forward and backward probabilities with stack entries as follows. Let G be the lexical tree corresponding to L with the natural probability structure inherited from the prior distribution on complete transcriptions. Each stack entry $\mathbf{f}$ can be thought of as a node in G. Forward and backward probabilities at nodes of G are calculated by treating them as states in the model M_G. We assume that the probability structure on G is taken into account and that duration constraints on phones are imposed.

The *score* of a partial transcription $\mathbf{f}$ (complete or incomplete) is defined as

$$h(\mathbf{f}) = \max_{0 \le t \le T} \alpha_t(\mathbf{f})\beta_t(\mathbf{f}).$$

The object is to find the complete transcription $\mathbf{f}$ for which $h(\mathbf{f})$ is maximal. A *heuristic score* is a function h^* defined on partial transcriptions. The heuristic is said to be admissible if $h(\mathbf{f}) \le h^*(\mathbf{f})$ for all partial transcriptions $\mathbf{f}$, with equality in the case of complete transcriptions. A proof that admissibility guarantees optimality can be found in [6].

Given an equivalence relation on G, we can construct an admissible heuristic as follows. Let G^* be the quotient graph and suppose we calculate the backward probabilities at the nodes of G^* using the Viterbi algorithm. (We do not assume that there is a probability structure on G^*.) If $\mathbf{f}$ is a partial transcription and $t = 0, \ldots, T$, we define $\beta_t^*(\mathbf{f})$ to be the backward probability at time t at the node in G^* corresponding to $\mathbf{f}$ (namely the equivalence class of $\mathbf{f}$). Note that, for $t = 0, \ldots, T$,

$$\beta_t(\mathbf{f}) \le \beta_t^*(\mathbf{f}). \tag{1}$$

For, if $t < T$, there is a path in G starting at $\mathbf{f}$ and ending at a leaf node of G such that

$$\beta_t(\mathbf{f}) = V(Y_{t+1}, \ldots, Y_T|\mathbf{f}')P(\mathbf{f}'|\mathbf{f}) \tag{2}$$

where $\mathbf{f}'$ is the phone string corresponding to the path and phone duration constraints are used in determining $V(Y_{t+1}, \ldots, Y_T|\mathbf{f}')$. Since G^* is a quotient of G, this path projects onto a path in G^* which starts at the node corresponding to $\mathbf{f}$ and ends at a final node of G^*. The corresponding phone string is also $\mathbf{f}'$, but the Viterbi score of $Y_{t+1}, \ldots, Y_T$ on this path in G^* must be greater than or equal to the right-hand side of (2), since G^* does not have a probability structure and minimum duration constraints on phones are not imposed. This implies (1). Now if we define

$$h^*(\mathbf{f}) = \max_{0 \le t \le T} \alpha_t(\mathbf{f})\beta_t^*(\mathbf{f}), \tag{3}$$

it is clear that the admissibility condition is satisfied.

Thus we get an A* admissible heuristic for each equivalence relation on the lexical tree. At one extreme, there is the trivial equivalence relation which identifies all of the nodes in the lexical tree. The graph G^* imposes no phonotactic constraints on phone strings so h^* will tend to seriously over-estimate h, causing a substantial amount of backtracking in the A* search. On the other hand, since the graph G^* is so small, very little work has to be done to evaluate the heuristic.

At the other extreme there are equivalence relations which have the property that the graph G^* does not overgenerate the lexicon. There may be many of these — for a given lexicon it is generally possible to construct graphs which generate it and are smaller than the lexical tree (using well known graph reduction techniques [3]). In this case, if phone duration constraints are not imposed, we have $h^* = h$ so the number of expansions performed by the A* search is minimized.

Although calculating the heuristic function is expensive since it requires an exhaustive search through a relatively large graph G^*, multiple recognition hypotheses can be found at essentially no extra cost. This is the idea behind the Tree-Trellis algorithm of Soong and Huang [8].

For each positive integer k we can define a k-phone equivalence relation on G as follows. Each of the nodes whose distance from the root node is less than k branches is assigned to its own equivalence class. Two of the remaining nodes are equivalent if the phone labels on the first k branches encountered in tracing back to the root node are the same. We have decided to work with the diphone equivalence relation ($k = 2$) since it appears to give a reasonable trade-off between the amount of back-tracking done during the A* search and the amount of computation needed to evaluate the heuristic scores.

In implementing this type of algorithm, stack entries consist of a partial transcription $\mathbf{f}$, the heuristic score $h^*(\mathbf{f})$ and an array of forward probabilities $\alpha_t(\mathbf{f})$, $t = 0, \ldots, T$. When a partial transcription $\mathbf{f}_1$ is extended by adding a phone f to give a new partial transcription $\mathbf{f}_2$, the forward probabilities $\alpha_t(\mathbf{f}_2)$, $t = 1, \ldots, T$, are calculated using precomputed impulse response functions (taking duration constraints into account): for $t = 1, \ldots, T$,

$$\alpha_t(\mathbf{f}_2) = \max_{t' < t} \alpha_{t'}(\mathbf{f}_1) h^f_{t'+1}(t). \tag{4}$$

Once these have been calculated the heuristic score for the extended transcription can be calculated from equation (3).

As it stands, the algorithm requires that a full array of forward probabilities $\alpha_t(\mathbf{f})$ be carried around with each of the partial transcriptions $\mathbf{f}$ on the stack. The computation could clearly be speeded up if the size of this array were reduced by roughly locating the endpoint of the partial transcription $\mathbf{f}$. Suppose $\mathbf{f} = f_1 \ldots f_k$. The endpoint of $\mathbf{f}$ is given by the expression

$$\operatorname*{argmax}_{0 \le t \le T} \alpha_t(\mathbf{f}) \beta_t(\mathbf{f})$$

which cannot be evaluated in practice since the backward probabilities are not known. An approximate endpoint is given by the expression

$$\operatorname*{argmax}_{0 \le t \le T} \alpha_t(\mathbf{f}) \beta^*_t(\mathbf{f}).$$

We have found empirically that this gives an estimate which is accurate to within a few frames[1] most of the time but occasionally makes much larger errors. On the other hand, it is reasonable to expect (and it turns out to be true in practice) that it should be possible to endpoint the phones f_{k-j} for $j = 1, 2, \ldots$ with a high degree of accuracy. We now show how the algorithm we have proposed can be modified to take advantage of this fact.

Looking one phone ahead

Let G^* be the quotient of the lexical tree G corresponding to the diphone equivalence relation and let h^* be the corresponding heuristic function.

We define a new heuristic $h^{(1)}$ as follows. Given a partial transcription $\mathbf{f} = f_1 \ldots f_k$, let n_{k-1} be the node in G^* corresponding to $\mathbf{f}_{k-1}$, and let n_k be the node corresponding to $\mathbf{f}_k$. Let b be the branch (n_{k-1}, f_k, n_k) and let s_i be the initial state of b. For each $t = 0, \ldots, T$ let $\beta^{(1)}_t(\mathbf{f})$ be the backward probability at time t at s_i (obtained by a Viterbi search through G^*). We set

$$h^{(1)}(\mathbf{f}) = \max_{0 \le t \le T} \alpha_t(\mathbf{f}_{k-1}) P(b | n_{k-1}) \beta^*_t(s_i).$$

Note that for $t < T$, $\beta^{(1)}_t(\mathbf{f})$ is the Viterbi score of the observations $Y_{t+1}, \ldots, Y_T$ on the optimal path through G^* which starts at node n_{k-1} and is constrained to satisfy the condition that the first branch taken on the path is b. Hence, if there are no constraints imposed on the duration of f_k, $h^{(1)}(\mathbf{f})$ is identical to $h^*(\mathbf{f})$ and if duration constraints are imposed

$$h^{(1)}(\mathbf{f}) \ge h^*(\mathbf{f}).$$

It follows that

(i) $h(\mathbf{f}) \le h^{(1)}(\mathbf{f})$, since $h(\mathbf{f}) \le h^*(\mathbf{f})$

(ii) $h(\mathbf{f}) = h^{(1)}(\mathbf{f})$ if $\mathbf{f}$ is complete, for in this case there are no duration constraints on f_k (the silence phone) so that $h^{(1)}(\mathbf{f}) = h^*(\mathbf{f}) = h(\mathbf{f})$

and hence that $h^{(1)}$ is an admissible heuristic.

We implemented both the A* algorithm using $h^{(1)}$ as a heuristic and the standard Viterbi algorithm in a speaker-independent phoneme-based telephone speech recognizer having a vocabulary consisting of 1,600 words and phrases on a VAX station 3540.

In order to ensure that the transcription and the corresponding Viterbi score found by the A* algorithm agree exactly with those found by the standard Viterbi algorithm in all cases, we implemented the A* algorithm as follows. Firstly, for each phone (except the silence phone) we imposed a minimum duration of 0 and a maximimum duration of 70 frames. (For most phones these constraints are over-generous but the standard Viterbi algorithm does not know this.) Secondly, for each partial transcription $\mathbf{f} = f_1 \ldots f_k$ on the stack we calculated an estimate $T^*(\mathbf{f})$ of the endpoint of $\mathbf{f}_{k-1}$ by setting

$$T^*(\mathbf{f}) = \operatorname*{argmax}_t \alpha_t(\mathbf{f}_{k-1}) \beta^{(1)}_t(\mathbf{f}) \tag{5}$$

and in using (4) to expand the stack entry we assumed that $\alpha_t(\mathbf{f}_{k-1}) = 0$ whenever $|t - T^*(\mathbf{f})| > 35$. Thus we associated an array of up to 71 forward probabilities with each stack entry. (Since the array index cannot extend beyond the beginning or the end of the utterance the size of the array is less than 71 in some cases.)Furthermore we modified the algorithm slightly so that the Viterbi segmentation of the first complete path appearing on the top of the stack could be generated.

We ran the A* algorithm on 861 words and found that under these conditions the CPU time needed to recognize a word was 3 minutes on average. This figure can be broken down as follows: 74 seconds for the calculation of backward probabilities at each node of the graph G^* by the Viterbi algorithm, 52 seconds to precompute all the impulse response functions and 55 seconds for the A* search itself.

In the Viterbi transcriptions of the words in the test set, there is a total of 12,396 segment boundaries. We found that the estimate given by (5) agrees with the exact endpoint 91% of the time. If the estimate were exact all of the time then it would be possible for us to reduce the number of forward probabilities associated with each stack entry from 71 to 1. Since this is not in fact the case we sought to minimize the number of forward probabilities needed by asking the following question: if we choose the N values of t which give the largest values of $\alpha_t(\mathbf{f}_{k-1})\beta^{(1)}_t(\mathbf{f}_k)$, how large does N have to be to ensure that the exact endpoint of $\mathbf{f}_{k-1}$ is included among the N hypotheses? The first column of the table 1 gives the relative frequency of affirmative answers to this question on this sample for different values of N. This relative frequency can be interpreted as the empirical probability that a segment boundary will be included among the top N hypotheses when we look ahead one phone.

N	SI 1-phone look ahead	SI 2-phone look ahead	SD 2-phone look ahead
1	0.913	0.967	0.999
2	0.947	0.972	1.000
3	0.964	0.986	1.000
4	0.975	0.989	1.000
5	0.980	0.991	1.000
6	0.985	0.994	1.000
7	0.990	0.995	1.000
8	0.992	0.996	1.000
9	0.993	0.997	1.000
10	0.994	0.997	1.000

Table 1. Probability that a segment boundary is included in the top N hypotheses. SI = speaker independent, SD = speaker dependent.

[1] 1 frame = 10 ms.

The speed of the A* search is directly proportional to the number of forward probabilities associated with each of the stack entries. The table indicates this number can be substantially reduced with essentially no loss in recognition performance. Furthermore, using realistic duration constraints will increase the speed of both the A* search and the impulse response calculation by a factor of 2 to 3 (there is good reason to believe that they will increase the recognition rate as well [2, 7]) and the latter calculation can also be reduced by computing the impulse responses on demand. Thus it should be possible to implement the algorithm in such a way that the time taken to recognize a word is only fractionally larger than the time taken to perform a Viterbi search through the graph G^*.

To compare the performance of this algorithm with that of the standard Viterbi algorithm, we first constructed a graph which generates the 1,600 word lexicon and is about half the size of the lexical tree. We found that a standard forward Viterbi search of this graph takes 6 minutes CPU time on average. (To ensure a fair comparison, no attempt was made to prune the stack in the A* search nor to restrict the forward trellis to a beam in the Viterbi search.)

Looking two phones ahead

The results of the last section show that on the 1,600 word application the A* algorithm with $h^{(1)}$ as a heuristic runs substantially faster than the standard Viterbi algorithm, but there is scope for improvement in endpointing partial hypotheses. In this section, we indicate a way of dealing with this problem by constructing a heuristic $h^{(2)}$ which can be used to endpoint the third last phone in each partial transcription and which can be evaluated with the same amount of computation as $h^{(1)}$.

Let G be the lexical tree, as before, and let G' be the reverse lexical tree. The diphone equivalence relation on G' is defined by assigning each of the nodes whose distance from the root node is less than two branches to its own equivalence class and saying that two of the remaining nodes are equivalent if the phone labels on the first two branches encountered in tracing forward to the root node are the same. Let G^* be the quotient of G' by this equivalence relation and suppose that the backward probabilities at the nodes of G^* have been calculated by the standard Viterbi algorithm (without using a probability structure or phone duration constraints).

Suppose we are given a partial transcription $\mathbf{f} = f_1 \dots f_k$ with $k \geq 2$. We associate a node n in G^* with $\mathbf{f}$ by letting n be the equivalence class defined by the pair $f_{k-1}f_k$ and we define $\beta_t^{(2)}(\mathbf{f})$ to be the backward probability at time t at n (calculated by the backward Viterbi algorithm in G^*). Define

$$h^{(2)}(\mathbf{f}) = \begin{cases} \max_{0 \leq t \leq T} \alpha_t(\mathbf{f}_{k-2})\beta_t^{(2)}(\mathbf{f}) & \text{if } \mathbf{f} \text{ is incomplete} \\ h(\mathbf{f}) & \text{if } \mathbf{f} \text{ is complete.} \end{cases} \quad (6)$$

It is easy to see that $h^{(2)}$ is an admissible heuristic. In implementing the A* search, the stack is initialized by entering all diphone sequences which appear in word-initial position as well as all complete transcriptions of length 1 (if any). The heuristic score is calculated using (6) in each case. If a stack entry $\mathbf{f} = f_1 \dots f_k$ is incomplete we associate with it a set of N forward probabilities $\alpha_t(\mathbf{f}_{k-2})$ by taking the N values of t for which $\alpha_t(\mathbf{f}_{k-1})\beta_t^{(2)}(\mathbf{f})$ is maximal. The number N can be determined empirically by collecting segmentation statistics, just as in the previous section. The statistics obtained under exactly the same conditions using a 572 word test set on the 1,600 word speaker-dependent application are given in the second column of table 1. The results show that the heuristic $h^{(2)}$ gives much more accurate segmentations of partial hypotheses, as expected.

The third column gives the result of the same experiment performed in our very large vocabulary speaker-dependent recognizer (clean speech sampled at 16 kHz and a 60,000 word lexicon). The test set consisted of 536 words uttered by a single speaker. The time taken to recognize a word on a DECstation 5000 was 36.8 seconds on average with the A* search taking less than 0.5 seconds. As in the previous section, no attempt was made to optimize the algorithm so these figures are very encouraging.

IV. Discussion

The problem we have addressed is that of finding the Viterbi transcription of an utterance subject to lexical and phone duration constraints. We have shown how to construct an A* admissible heuristic for this problem by using the results of a Viterbi search through a relatively small graph G^* which imposes weak lexical constraints on phone strings and no constraints on phone durations.

We implemented two versions of the algorithm in isolated word recognizers having vocabularies of 1,600 and 60,000 words using graphs G^* which impose triphone phonotactic constraints. We found that it can be made to run substantially faster than the standard Viterbi algorithm (which does not respect phone duration constraints and only generates a single hypothesis for the Viterbi transcription of the utterance).

Our choice of the graph G^* was partly motivated by Soong's results on connected phone recognition in Japanese [7]. By incorporating triphone phonotactic constraints as well as phone duration constraints into his phonetic decoder, he was able to obtain a phone error rate of 4%. If 96% of the phones in the string generated by the phonetic decoder are correct, it seems that it ought not be too difficult to impose lexical constraints on the output of the phonetic decoder. A reasonable strategy for doing this would be to match such phone strings with transcriptions in the lexicon using a string matching scheme such as that used by Levinson et al. [5], thereby generating a list of candidate transcriptions to be scored exactly. However we have found that in our applications it is difficult to guarantee the optimality of such a strategy without making the list of candidate transcriptions rather large. On the other hand, the A* algorithm does this in an optimal way and is only fractionally more computationally expensive than the standard Viterbi search through the graph G^*. This is achieved by using *all* of the data generated by the Viterbi search through G^* in lexical access and not just the phone string it produces.

References

[1] Y.-L. Chow and R. Schwartz, "The N-Best Algorithm: An Efficient Procedure for Finding Top N Sentence Hypotheses", *Proc. DARPA Speech and Natural Language Workshop*, pp. 199–202, October 1989.

[2] V.N. Gupta, M. Lennig, P. Mermelstein, P. Kenny, F. Seitz and D. O'Shaughnessy, "Using phoneme duration and energy contour information to improve Large Vocabulary Isolated Word Recognition", submitted to *Proc. ICASSP* , 1991.

[3] J.E. Hopcroft, "An n log n algorithm for minimizing the states in a finite automaton", *The Theory of Machines and Computation* (Z. Kohavi, ed), pp. 189–196, 1971.

[4] F. Jelinek, "Continuous Speech Recognition by Statistical Methods", *Proc. IEEE*, vol. 64, no. 64, 1976.

[5] S.E. Levinson, A. Ljolje, and L.G. Miller "Continuous Speech Recognition from a Phonetic Transcription ", *Proc. ICASSP*, pp. 93–96 , 1990.

[6] N. Nilsson, "Principles of artificial intelligence ", Tioga Publishing Company ,1982.

[7] F.K. Soong, "A Phonetically Labeled Acoustic Segment (PLAS) Approach to Speech Analysis-Synthesis" *Proc. ICASSP* , pp. 584–587 , 1989.

[8] F.K. Soong and E.-F. Huang, "A Tree-Trellis Based Fast Search for Finding the N-Best Sentence Hypotheses in Continuous Speech Recognition " *Proc. DARPA Speech and Natural Language Workshop*, June 1990.