Fast search strategy in a large vocabulary word recognizer

V. N. Gupta,[a] M. Lennig,[a] and P. Mermelstein[a]
INRS-Télécommunications, Montreal, Quebec H3E 1H6, Canada

(Received 14 April 1987; accepted for publication 20 July 1988)

In this article, a fast search algorithm is presented for generating word hypotheses for a 75 000-word vocabulary, speaker-trained, isolated word recognizer. The algorithm is envisioned as the first pass of a total recognition system generating a small number of hypotheses with rough likelihood estimates, to be followed by more detailed hypothesis evaluation. The possible word choices are restricted by estimating the number of syllables in the unknown word using a hidden Markov model (HMM) for syllables. A heuristic search algorithm then searches through a sequence of syllable networks to find the most likely word candidates. Arcs in the syllable network correspond to phonemes. The assumption that the likelihoods of these phoneme arcs are independent of the phonetic context allows us to convert the search through a large tree into a search through a much smaller network or graph. The computational requirements are reduced by roughly a factor of 70 compared to estimating the exact likelihood scores for the 75 000 words. This fast search algorithm is called the syllabic graph search. The recognition accuracy obtained for the syllabic graph search approaches that obtained using the exact likelihood scores for the phoneme sequences.

PACS numbers: 43.72.Ne

INTRODUCTION

The goal of our 75 000-word recognizer is to transcribe text spoken as a sequence of isolated words. For each spoken word, the recognizer uses acoustic information and rough likelihoods in a fast search algorithm to narrow the possible word hypotheses from the 75 000 words in the total vocabulary to a small list. It then refines the list by computing an exact likelihood score for each hypothesized word. The exact likelihood scores take into account acoustic information, but not the syntactic and semantic characteristics of English. To take these into account, the exact likelihoods are further refined with the aid of a statistical language model to generate the most likely sequence of words. In this article, our focus is on generating the inital short hypothesis list of the most likely words.

A simplistic approach to large vocabulary recognition would compare the unknown (word to be recognized) against all possible phonemic transcriptions for all the words in the vocabulary. To score the 75 000 vocabulary words in real time would require hardware with computing capability on the order of hundreds of millions of floating point operations per second. The cost of the hardware to provide such computing capability would be prohibitive. To reduce the number of candidates for detailed comparison, we require an algorithm to perform a fast search, reducing significantly the size of the list for subsequent detailed search.

The stack decoding algorithm (Jelinek, 1976) may be employed to achieve a heuristic search. Bahl *et al.* (1983) have used the stack decoding algorithm to search a word tree defined by a language model with a vocabulary size of 5000 words.[1] In contrast, we apply the stack algorithm to search a lexical tree where each branch of the tree is labeled with a phoneme and is considered to be *a priori* equally likely. The implementation of the stack decoding algorithm used by Bahl *et al.* is computationally expensive since it performs a heuristic search through a tree that contains all possible valid pronunciations. The major computational load is incurred in computing the branch likelihoods to extend the path on top of the stack.

The phoneme tree for our 75 000-word recognizer is quite large, containing over 445 000 branches. Significant computational savings result if we modify the strategy to search through a much smaller graph[2] instead of a tree and assume that the likelihood of every arc in the graph is independent of its phonetic context. The rough likelihood of any path in the graph can be rapidly computed from the likelihoods of the arcs constituting the path. This permits limiting the search to a graph having less than 2000 arcs. We call this search algorithm the *syllabic graph search*.

Additional computational savings result from restricting the search to a subset of the entire vocabulary. This is accomplished by first estimating the number of syllables in the word and then restricting the search to words with the required number of syllables.

The article is organized as follows. We first give an overview of the word recognition strategy in Sec. I. Sections II–VIII describe various components of the word recognition system. Section II outlines the 75 000-word lexicon. The training and test sets employed for evaluating the recognition algorithm are described in Sec. III. In Sec. IV we give details of the acoustic parameters used for recognition. In Sec. V the syllable networks used in the syllabic graph search are developed in detail. Section VI describes the algorithm for estimating the number of syllables in the word. Section VII outlines the hidden Markov models (HMMs) used in the syllabic graph search algorithm. Section VIII gives details of the syllabic graph search for finding the most likely word candidates.

[a] Also with Bell-Northern Research, Montreal, Canada.

 0001-4966/88/122007-11$00.80

I. OVERVIEW OF THE 75 000-WORD RECOGNIZER

A block diagram of the recognition system is shown in Fig. 1. Recognition is performed in four stages.

The first stage determines the end points of the words using C_0 (see Sec. IV for details on C_0). The temporal sequence of feature vectors between these two end points is employed by the succeeding stages to recognize the unknown. Explicit detection of the end points limits the computations required for the following stages, as opposed to implicit end-point detection, where the best match is evaluated between broader limits, possibly allowing silent segments to be appended to either end. The number of end-point detection errors is less than 1% and has only a minimal effect on the performance of the recognizer.

The second stage of recognition generates a number of hypotheses for the syllable count (total number of syllables) in the unknown. This algorithm is described in more detail in Sec. VI. Estimation of the number of syllables in the word allows us to search a subset of our vocabulary corresponding to the words with the estimated syllable count. For each estimate K of the syllable count, we form a graph consisting of a concatenation of K distinct syllable networks (see Sec. V for details on the syllable networks). Each arc of the graph corresponds to one of 44 allophone models listed in Appendix A, or to a null transition. Every phoneme sequence in the lexicon corresponding to a K-syllable word has a path through this graph. This graph is called the syllabic graph of count K. The syllabic graph is used for rapid scoring of partial phoneme strings.

The third stage of recognition is a syllabic graph search (the focus of our article) that computes the sequence of most likely phoneme strings through the syllabic graph. The syllabic graph search performs a fast search through all possible paths within the syllabic graph using a variation of the *stack algorithm* (Jelinek, 1976), or the *A * algorithm* as it is termed in the artificial intelligence literature (Nilsson, 1980). A partial path is put on the stack only if it corresponds to a node of the lexical tree. The branches of the lexical tree correspond to phonemes and the leaves of the tree correspond to words in the vocabulary. With each branch in the lexical tree we associate tags indicating the possible syllable counts through that branch and the syllable position for the branch. The syllable position indicates the position of the syllable in the word with which the branch is associated. The lexical tree has approximately 445 000 branches. An example of a lexical tree representing only five words is shown in Fig. 2. The output of the syllabic graph search is a list of the N most likely lexically valid phoneme strings and their associated rough likelihoods. The syllabic graph search algorithm is described in detail in Sec. VIII.

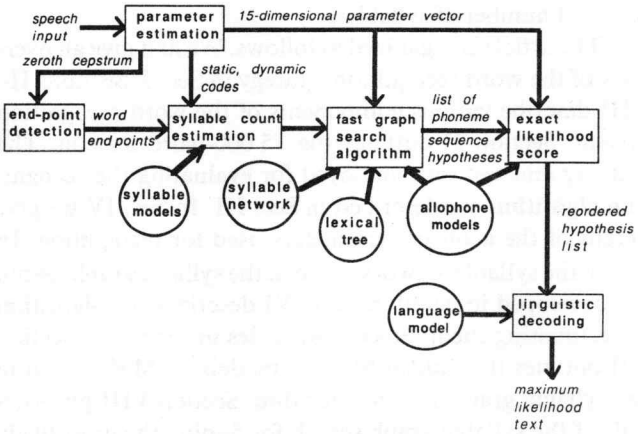

FIG. 1. Block diagram of the large vocabulary recognition system.

The fourth stage of recognition computes the exact likelihood scores for the phoneme strings generated by the syllabic graph search and reorders the phoneme strings based on these scores. The exact likelihood scores provide improved recognition performance over the rough likelihoods computed by the syllabic graph search algorithm. Exact likelihood computation for the hypotheses found most likely by the syllabic graph search requires minimal additional computing.

Although we have explored the use of language models to improve the recognition rates, no language model is included in the experiments reported in this article. The statistical language model we have implemented is based on a trigram language model (Jelinek, 1985). The trigram language model has been shown to perform quite well for a vocabulary size of up to 20 000 words (Averbuch *et al.*, 1987). The word trigram probabilities are estimated using a large text corpus. In our system, trigram probabilities are then used to find the most likely word sequence from the word lattice generated by the first four stages of recognition. To focus on acoustic recognition, all the 75 000 words in the lexicon are considered *a priori* equiprobable in the experiments reported in this article.

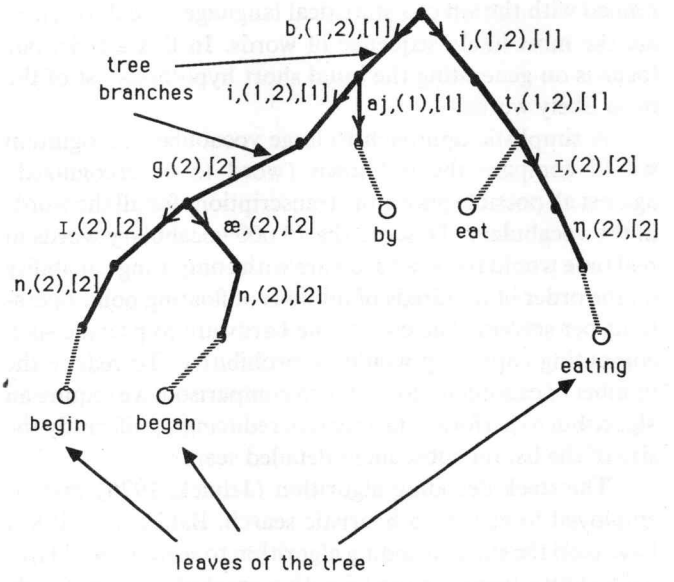

FIG. 2. Example of a lexical tree containing five words: begin, began, by, eat, eating. Each branch has a phoneme label, a (possible syllable counts tag), and a [syllable position tag] associated with it. The possible syllable counts tag shows all the possible syllable counts for words through the branch. The syllable position tag shows the position of the syllable in the word with which the branch is associated.

The accuracy of the recognizer is evaluated using two quantitative measures. One criterion is the percent of words for which the top word choice is correct.[3] The second criterion is the average rank of the correct word in the ordered list of word hypotheses.

II. THE 75 000-WORD LEXICON

All recognition experiments and lexical statistics reported here are based on a lexicon of 76 211 orthographically distinct words. These words correspond to the contents of Merriam Webster's Seventh Collegiate Dictionary (1965; henceforth, W7), together with most of the words in the Brown Corpus (Francis and Kucera, 1979) not in W7, plus approximately 400 common first names of people, and all of the words in the training and test sets used for the recognition experiments.

Words are distinguished orthographically. For example, *book* and *books* are considered two different words in the vocabulary. On the other hand, *bow*(n)/bo/ and *bow*(v) /baw/ are considered to be the same word. Note that W7 does not list regular inflected forms. The frequently occurring inflected words in our lexicon have been taken from the Brown Corpus.

Because some words have multiple pronunciations, the 76 211 distinct words in the lexicon have a total of 126 445 distinct phonemic transcriptions; 73 554 words have at least one phonemic transcription that is different from all other words and 72 709 words have no homophone confusions at all among their phonemic transcriptions. We refer nominally to our vocabulary size as 75 000 even though any of the numbers 72 709, 73 554, 76 211, and 126 445 could be justified according to the various possible definitions.

III. TRAINING AND TEST SETS

The training and test sets consist of different sentences read in a quiet room by the same speaker. All of the training and test texts are read with pauses of at least 150 ms between the words. The texts were selected arbitrarily from magazines, office correspondence, books, and newspaper articles. Sample sentences from the test set are shown in Appendix B. Some additional training words are included to provide extra tokens of phonemes in consonant clusters and in CVC context, where C stands for one of the consonants /ptkbdgrl/. The training set consists of approximately 850 word tokens corresponding to approximately 10 min of speech (not including the silence intervals between words).

The training and test texts were spoken by two male native English speakers. One speaker's phonetic and phonological patterns are characteristic of the Philadelphia speech community, while the other's are characteristic of Montreal speech. Both speakers are researchers associated with our speech recognition project.

IV. ACOUSTIC FEATURES

Speech is captured using a Crown PZM microphone, low-pass filtered at 7.1 kHz and sampled at 16 kHz. A 15-dimensional feature vector is computed every 10 ms from the sampled speech waveform using a 25.6-ms overlapped window. The 15-dimensional feature vector consists of seven mel-based static cepstrum coefficients $(C_1,...,C_7)$ (Davis and Mermelstein, 1980) and eight dynamic parameters $(\Delta C_0,...,\Delta C_7)$.

The static cepstral coefficients $(C_1,...,C_7)$ are computed by first dividing the spectrum between 0 and 8 kHz into 24 channels spaced according to the mel scale of frequency. The center frequencies for the first ten channels are spaced 100-Hz apart, while the remaining 14 channels are spaced logarithmically. The energy in each channel is computed by summing a triangularly weighted spectrum located at the center of the channel. Taking the log of the channel energies yields the log channel energies. The cosine transform of the vector of 24 log channel energies given by

$$C_i = \sum_{j=1}^{24} E_j \cos\left(i(j-0.5)\frac{\pi}{24}\right), \quad i = 1,2,...,7\,,$$

where E_j is the log channel energy in the jth channel, gives us the cepstrum coefficients. Only the first seven static cepstrum coefficients $(C_1,...,C_7)$ are used in recognition. C_0 is computed as a weighted sum of the log channel energies

$$C_0 = \sum_{j=1}^{24} W_j E_j\,,$$

where the weights W_j simulate the perceptual loudness contour.[4] The cepstrum coefficients have been shown to give improved recognition performance compared to a number of other feature parameters used in speech recognition (Davis and Mermelstein, 1980).

The eight dynamic parameters $(\Delta C_0,...,\Delta C_7)$ are obtained by taking signed differences between the corresponding static cepstral values 40-ms apart. The resulting 15-dimensional feature vector $(C_1,...,C_7,\Delta C_0,...,\Delta C_7)$ is computed every 10 ms.

V. THE SYLLABLE NETWORK

The syllabic graph search algorithm searches through all possible phoneme sequences corresponding to valid pronunciations of the words in the vocabulary. How can we represent these phoneme sequences efficiently? We represent the phoneme sequences using a graph, where each arc of the graph corresponds to a phoneme. A path through the graph exists for every phoneme sequence constituting a word in the lexicon. The speed of the search is controlled by the complexity (number of arcs) of the graph.

The complexity of the graph can be reduced by considering the phonetic structure of phoneme sequences within words. A word consists of one or more syllables. Every syllable has a nucleus consisting of a vowel and may also have an onset and a coda arranged in the form

$$(O_1(O_2(O_3)))N(C_1(C_2(C_3(C_4))))\,,$$

where O_i stands for a consonant in the syllabic onset, N for the vowel in the syllabic nucleus, and C_j for a consonant in the syllabic coda. The parentheses indicate optional occurrence. Since the syllabic onset, nucleus, and coda are the subunits of the syllable within which the tightest phonotac-

tic constraints obtain (Selkirk, 1982), we construct our syllable network by concatenating three subnetworks representing these three parts of the syllable. For example, using such a structure, we can easily capture the constraint that all three-consonant onsets end in a liquid (/l/ or /r/) or a glide (/j/ or /w/). Since not every phonotactic constraint can be captured in this way, we allow the syllable network to overgenerate (i.e., generate a superset of) the allowable syllables. Finally, a graph to represent words is obtained by concatenating a number of syllable networks.

One may ask whether a single syllable network suffices to represent all the syllables in the lexicon or whether it is preferable to use a distinct syllable network in each syllable position within the word. To resolve this question, we computed the different syllables that can occur in each syllable position of the words in the lexicon. As shown in Table I, analysis of our 75 000-word vocabulary reveals a significant rapid decrease in the number of alternative syllables that may occupy positions two, three, etc. of polysyllabic words.

The results of Table I suggest the creation of a separate syllable network for each syllable position of the word. This results in smaller word graphs compared to having a single syllable network to represent all the 12 128 syllables in the 75 000-word lexicon. The maximum number of syllables in any word in the vocabulary is ten, resulting in ten distinct syllable networks.

As mentioned above, each of the ten syllable networks has separate subnetworks for the onset, nucleus, and coda since these are the subunits of the syllable with the tightest phonotactic constraints. The subnetwork for onset has separate paths for one- two-, and three-consonant onsets. Similarly, the subnetwork for coda has separate paths for one-, two-, three-, and four-consonant codas. The syllable network for word-initial syllables is shown in Fig. 3. The syllable network representing all possible ninth syllables in the 75 000-word vocabulary is shown in Fig. 4. Notice that the syllable networks representing later syllable positions are more and more restrictive.

In the syllabic graph search, we first estimate the syllable count K of the unknown. We then create a K-syllable word graph by concatenating the syllable networks for syllable positions 1 through K. The syllabic graph search is performed over this graph. The amount of computation required in the syllabic graph search is proportional to the sum of the number of distinctly labeled arcs emanating from each node of the network. We call this sum the *total distinct arcs* in the network. Table II shows the total number of arcs and total distinct arcs for the syllable network corresponding to each syllable position in the word. The network for the first syllable has 504 arcs, while the network for the tenth syllable has only four arcs. The total number of arcs for all ten syllable networks is 1913.

TABLE I. Count of number of phonemically distinct syllables occurring in each syllable position in the vocabulary. The total number of distinct syllables in all positions is 12 128.

Syllable position	Total distinct syllables
first	9350
second	6597
third	3942
fourth	1672
fifth	668
sixth	269
seventh	95
eighth	33
ninth	10
tenth	3

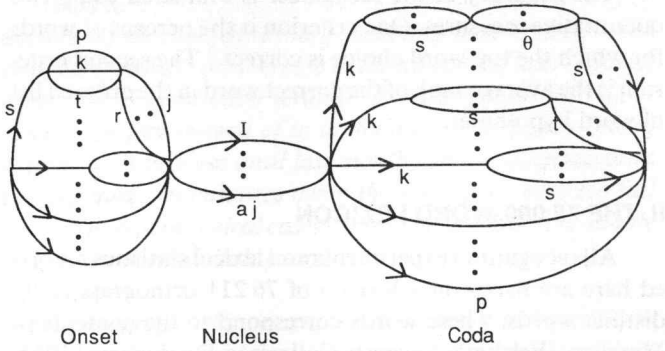

FIG. 3. Syllabic network describing all possible word-initial syllables. The labels on the arcs show examples of possible phonemes.

There are two disadvantages to concatenating syllable networks to obtain graphs with a given syllable count. First, concatenation of syllable networks does not allow imposition of intersyllabic phonotactic constraints, thus giving rise to overgeneration. For example, concatenated syllable networks permit paths between two consecutive vowels that traverse as many as seven consonants (four coda and three onset consonants). However, the lexicon indicates that the maximum number of consonants between any two vowels is five. Additionally, these five-consonant sequences are greatly restricted. Only 24 distinct five-consonant sequences exist within words in our 75 000-word lexicon. In 11 cases, the first three consonants are /nts/ and in only two cases is the third consonant not an /s/.

The second disadvantage of concatenating syllable models is that we cannot take advantage of the fact that the distributions of syllables which occur in word-initial position is strongly influenced by the syllable count of the words. For example, even though 9350 different syllables occur in word-initial position, only 6898 occur in monosyllabic words, 4976 in bisyllabic words, and 3647 in words three syllables long. These numbers decrease rapidly with increas-

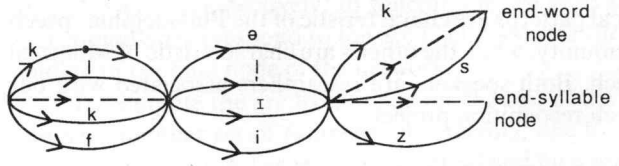

FIG. 4. Syllabic network for the ninth syllable in all the words in the dictionary. The dotted line indicates a null transition. The network has two possible end nodes. The end-word node corresponds to words which are nine syllables long, while the end-syllable node corresponds to the ninth syllable of words ten syllables long.

TABLE II. Total number of arcs for the syllable network corresponding to each syllable position of the words in the vocabulary. Total distinct arcs is the sum, over the nodes, of the number of distinctly labeled arcs emanating from that node.

Syllable position	Total arcs	Total distinct arcs
first	504	341
second	418	284
third	336	229
fourth	250	176
fifth	168	121
sixth	107	75
seventh	73	58
eighth	38	33
ninth	15	13
tenth	4	4
Total	1913	1334

TABLE IV. Distribution of word transcriptions in the lexicon by syllable count. The 76 211 distinct words in the lexicon have 126 445 distinct transcriptions corresponding to all possible pronunciations of the words. In all, 73 554 words have at least one distinct transcription, while 72 709 words have no homophone confusions.

Syllable count of word	Number of word transcriptions	Percentage of lexicon
1	6 894	5.5%
2	33 891	26.8%
3	39 368	31.1%
4	27 845	22.0%
5	12 810	10.1%
6	4 216	3.3%
7	1 142	0.9%
8	249	0.2%
9	25	0.02%
10	5	0.004%

ing syllable count. The same is true of syllables that occur in other syllable positions of the word. Some of these numbers are shown in Table III.

An alternative approach based on the above considerations, which would further restrict the possible productions, would be the construction of separate networks for each possible syllable count, as opposed to the concatenation of syllable networks. However, for considerations of simplicity, this alternative was not implemented.

VI. ESTIMATION OF NUMBER OF SYLLABLES IN THE WORD

We require an estimate of the number of syllables in the unknown to specify which syllabic networks to search. The number of possible choices for the syllable count in our lexicon is ten. If we can specify one or two estimated values for the syllable count of the unknown, the list of candidates can be reduced to less than half of the lexicon. Table IV shows the number of word transcriptions in the dictionary for each possible syllable count. For a monosyllabic word, we have to search through only one-twentieth of the phoneme sequences in the lexicon, while for words with three syllables we have to search through one-third of the lexicon.

To allow the syllabic graph search algorithm to process a graph with a fixed syllable count, the number of syllables in the word has to be estimated first. We have tested two algorithms to estimate the number of syllables in the word. In both cases the overhead for estimating the number of syllables is small, less than 2% of the computation time required for recognition on the VAX 8600.

TABLE III. Total distinct syllables in each syllable position for words with different syllable counts.

Syllable count of word	Total distinct syllables in syllable position				
	first	second	third	fourth	fifth
1	6898	...	...	...	...
2	4976	5217	...	...	...
3	3647	3020	3036	...	...
4	2099	2121	1745	1167	...
5	1247	1153	1161	730	455
6	627	588	599	454	266

The preferred algorithm for estimating the number of syllables uses hidden Markov modeling (HMM). An alternative algorithm, based on the convex hull algorithm (Mermelstein, 1975), was tried as well, but yielded worse results. We will describe in detail only the HMM-based estimation of the number of syllables.

Several algorithms may be considered for carrying out HMM-based estimation of the number of syllables. For word recognition, we have observed that HMMs with Gaussian distributions for the output probabilities give higher recognition performance compared to HMMs trained with vector quantized outputs. Among the different alternatives for HMMs using vector quantized (VQ) outputs, we have found a word-level integration algorithm (Gupta *et al.*, 1987) to give the best performance. However, for estimating the number of syllables in the word, the word-level integration algorithm performs better than using HMMs with Gaussian distributions for the output probabilities.

To estimate the number of syllables in the word using the word-level integration algorithm, we use two HMMs called the *syllable models*. The syllable model is not a model of a specific syllable, but a general syllable model trained on all the syllables in the training set. One syllable model (called the *static syllable model*) is estimated using only the static cepstral parameters, while a second syllable model (called the *dynamic syllable model*) is trained using the dynamic parameters only. To train the syllable models, the 850 words used in the training set (described in Sec. III) are manually segmented into syllables. Each syllable constitutes one observation of the form $O_1^l = O_1,...,O_l$, a temporal sequence of l 15-dimensional acoustic feature vectors. Approximately 1800 observations are used to train the syllable models.

The static syllable model is trained as follows. We train a codebook of size 64 using a sequence of static cepstral parameter vectors corresponding to approximately 2 min of speech. The codebook is generated using the k-means algo-

rithm (Duda and Hart, 1973). All the cepstral parameter vectors in the training set are coded using this codebook. Each observation for training the static syllable model is of the form $S_1^l = S_1, S_2, \ldots, S_l$, a temporal sequence of VQ codes corresponding to the static cepstral parameters for the syllable. The structure of the syllable model is that of a ten-state[5] left-to-right (or Bakis) model (Jelinek, 1976), as shown in Fig. 5. Each state has self-loop, next-state, and skip-transition probabilities. With each transition in the syllable model we associate a transition probability and an output probability distribution on the codes. We employ the forward–backward algorithm to train the cepstral syllable model (Jelinek, 1976; Levinson *et al.*, 1983). The dynamic syllable model is trained similarly using observations of the form $D_1^l = D_1, \ldots, D_l$, a temporal sequence of VQ codes corresponding to the dynamic cepstral parameters for the syllable.

To estimate the number of syllables in the unknown, we compute the likelihood that the unknown is K syllables long by concatenating the syllable model with itself K times. In other words, if the unknown is represented by acoustic feature vectors O_1^l, then we compute the likelihoods $L(O_1^l \mid \text{syllable count} = 1), \ldots, L(O_1^l \mid \text{syllable count} = 10)$. The likelihood of each possible syllable count hypothesis is computed as the product of the likelihood of the syllable count hypothesis using the static cepstral parameters and that of the syllable count hypothesis using the dynamic cepstral parameters. If S_1^l and D_1^l are the temporal sequence of VQ codes corresponding to the static cepstral parameters and the dynamic cepstral parameters for the word, respectively, then

$$L(O_1^l \mid \text{syllable count} = n) \triangleq P(S_1^l \mid \text{syllable count} = n) \times P(D_1^l \mid \text{syllable count} = n) .$$

The syllable count hypotheses are ranked according to their likelihoods. Only the hypotheses having an average per frame likelihood greater than half that of the maximum likelihood hypothesis are retained, up to a maximum of 3.

When tested on 714 words, the top syllable count hypothesis was found correct for 85.5% of the words, the second choice was correct for 13% of the words, and the third choice was correct for the remaining 1.5% of the words. An average of 2.3 syllable count hypotheses were retained for each word. Table V shows the confusion matrix for the top choice for the syllable counts of the test words. Most of the errors are due to a three-syllable word recognized as a two-syllable word or a four-syllable word recognized as a three-syllable word. The words are generally polysyllabic words including an unstressed syllable comprising a vowel alone. Some examples of such erroneous words are *capital* /kæp-ə-tl/, *period* /pɪr-i-əd/, and *immigration* /ɪm-ə-gre-ʃən/.

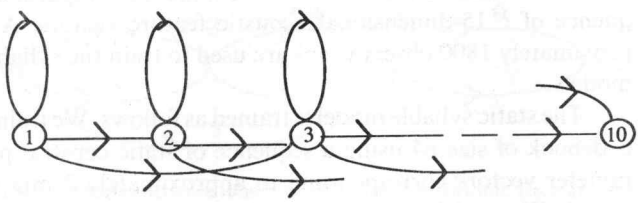

FIG. 5. Structure of the syllable model.

TABLE V. Confusion matrix for the estimation of syllable count in a word. The test set size is 714 words.

Actual syllable count	Estimated syllable count 1	2	3	4	5
1	382	15	0	0	0
2	6	167	6	0	0
3	2	53	50	2	0
4	0	2	16	11	0
5	0	0	0	2	0

One way to eliminate errors in three-syllable words like *capital*, *period*, etc. is to allow the syllabic graph of syllable count two to have paths corresponding to these three-syllable words. Similarly, the syllabic graph of syllable count three would have paths corresponding to four-syllable words like *immigration* (four-syllable words having an unstressed syllable comprising a vowel alone). Syllabic graphs of count K that allow paths for some words containing $(K+1)$ syllables require the construction of a separate syllabic graph for each possible syllable count, instead of forming the syllabic graph by concatenating K distinct syllable networks.

Using a syllable model with Gaussian distribution of output probabilities instead of the VQ-based static and dynamic syllable models gave poorer results, as shown in Table VI. For the Gaussian model, 79% of the top choices, 16% of the second choices, and 5% of the third choices were correct.

We conclude that the syllable models do a reasonable job of estimating the number of syllables in the word. Using VQ-based models, overall results show that the top two syllable count hypotheses include the correct estimate over 98% of the time.

VII. ALLOPHONE MODELS

In Sec. V we described the syllabic graphs whose arcs correspond to phonemes. In the current section, we will describe how the phonemes themselves are modeled.

For the most part, we have used one HMM to represent each phoneme of English. However, in the case of the liquids /l/ and /r/, we use two HMMs each. The phonemes /l/ and /r/ have distinct prevocalic and postvocalic models. (Syllabic [l] and [r] are grouped in the same allophone with postvocalic /l/ and /r/.) We also employ one HMM to represent aspiration at the start of the word and one for the breath noise at the end of the word. Each allophone model is

TABLE VI. Comparison of results for syllable count estimation using VQ-based versus Gaussian-based syllable models. The VQ-based syllable models use a word-level integration algorithm.

Syllable model	Top choice	Second choice	Third choice
VQ based	85.5%	13%	1.5%
Gaussian	79%	16%	5%

trained on a training corpus (described in Sec. III) that has been manually segmented into phones for training. Training is performed using the forward–backward algorithm (Jelinek, 1976; Levinson *et al.*, 1983). We term the HMMs thus trained *allophone models*. Appendix A gives a complete list of the allophone models.

The structure of the allophone models corresponds to the *left-to-right* or *Bakis* model (Jelinek, 1976). With each transition in the model, we associate a mean acoustic feature vector. While we estimate a separate mean vector for each transition, we compute a pooled covariance matrix over all the transitions within a particular model, resulting in a common covariance matrix for each allophone. Generation of one covariance matrix for each allophone model results in a significant reduction in computation and in elimination of singular covariance matrices for some transitions during training.

The number of states varies among the allophone models: Consonant models have between three and six states and vowel models have between five and ten states. The number of states for each allophone model was optimized using a test set different from the one used in this article. In general, long vowels have ten states and short vowels except /ə/ have six states. The number of states used for each allophone model is given in Appendix A.

VIII. SYLLABIC GRAPH SEARCH ALGORITHM FOR DECODING WORDS AS SEQUENCES OF PHONEMES

As mentioned in the Introduction, a simplistic approach to large vocabulary recognition would compare the unknown against all possible phonemic transcriptions for all the words in the lexicon. To compute the exact likelihood scores for the 75 000 word choices requires over 80 min of CPU time on the VAX 8600. The cost of the hardware to provide real-time recognition capability would be prohibitive. To reduce the number of candidates for precise likelihood evaluation, we require an algorithm to perform a fast search through all the possible word candidates. The syllabic graph search algorithm reduces the search time from over 80 min to 1 min on the VAX 8600. For each hypothesized syllable count of the unknown, we search the corresponding syllabic graph created by concatenating the appropriate number of syllable networks, as outlined in Sec. V.

The syllabic graph search, while faster, is less accurate than computing exact likelihoods of all the words in the lexicon. The exact likelihood is the likelihood of the temporal sequence of acoustic feature vectors of the unknown given the sequence of allophone models corresponding to the phonemic transcription of the hypothesized word. The rough likelihood of a path in the syllabic graph, corresponding to a partial or complete allophonic transcription, is computed as a function of the likelihoods of the arcs (arc likelihoods) constituting the path. The arc likelihood is an approximation to the likelihood of the highest exact likelihood path through that arc. The rough path likelihoods used in the syllabic graph search are always greater than or equal to the exact likelihoods.

Recognition results obtained using the syllabic graph search are only slightly poorer than those obtained by using exact likelihoods. For example, on a test set of 714 words, the average rank of the words using the syllabic graph search was 2.8, while the average rank using the exact likelihood scores was 2.3. Retaining several of the most likely hypotheses minimizes the risk of excluding the correct hypothesis. Estimating the exact likelihood scores for the small number of retained word hypotheses further sharpens recognition accuracy. This step is inexpensive in terms of computing requirements because exact likelihoods need to be computed for only 30–90 word hypotheses. Since we can compute the exact likelihood scores for the retained word hypotheses quite inexpensively, we do not incur any significant loss of performance by switching from maximum likelihood decoding to the syllabic graph search algorithm.

The major advantage of the syllabic graph search is that it searches a small graph instead of a large tree. In a tree search, extending each branch of the tree to obtain the likelihood values requires a significant amount of computation. Our search is through the syllabic graph, where each arc of the graph corresponds to an allophone model. The arc likelihoods are assumed to be independent of the preceding or the following phonemic context and can therefore be precomputed.

Figure 6 outlines the difference between a tree search and a graph search. In the tree search, the allophone model for /ɑ/ is used three times to extend the paths for /p/, /t/, and /k/. In the syllabic graph search, the allophone model for the arc labeled /ɑ/ is used only once to precompute the arc likelihood independent of the context. (We compute the likelihood of every arc in the graph only once.) Once the arc likelihoods are known, the likelihoods of paths through the network can be computed rapidly. By employing context-independent arc likelihoods we reduce the search to finding the most likely paths through the graph.

It is desirable to restrict the syllabic graph search to allophone sequences corresponding to phoneme subse-

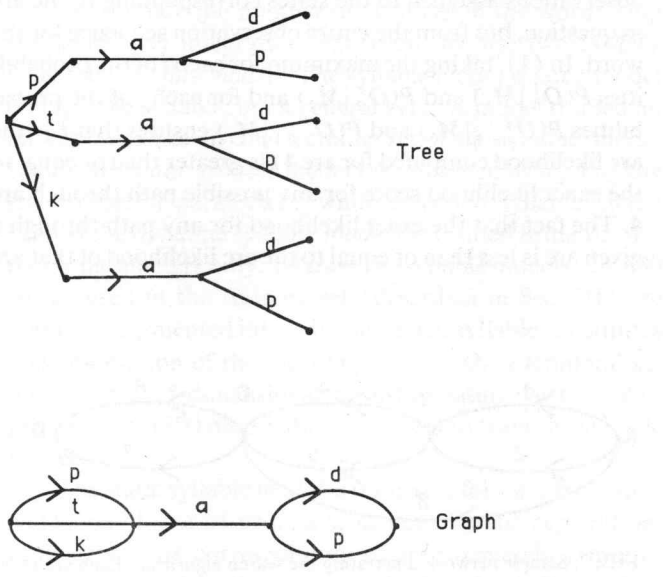

FIG. 6. Comparison of the graph search and the tree search algorithms.

quences of words in the lexicon having the desired syllable count. This is achieved using the lexical tree (see Sec. I and Fig. 2). The lexical tree places three restrictions on partial paths through the syllabic graph. First, a valid partial path through the syllabic graph must have a corresponding partial path in the lexical tree. Second, the partial path in the lexical tree must have the correct associated syllable count tag. Third, the syllable position tag for each branch of the partial path in the lexical tree must correspond to the correct syllable in the syllabic graph with which the branch is associated.

In order to compute rough likelihoods of paths through the syllabic graph, we first calculate all the arc likelihoods. We illustrate the algorithm for computing arc likelihoods using the network of Fig. 7. Consider the computation of the likelihood of arc 4. The subnetwork of Fig. 8 is derived by retaining only the arcs that can form a path through arc 4. Let $O_1^l = O_1,\ldots,O_l$ be the observation sequence of feature vectors corresponding to the input word and let M_1, M_2, M_4, M_6, and M_7 be the allophone models corresponding to the allophones represented by arcs 1, 2, 4, 6, and 7. Using the subnetwork in Fig. 8, the likelihood value for arc 4 is then given by

$$L_4 = \sum_{i,j} \max\left[P(O_1^i|M_1),P(O_1^i|M_2)\right]P(O_{i+1}^j|M_4) \times \max\left[P(O_{j+1}^l|M_6),P(O_{j+1}^l|M_7)\right], \quad (1)$$

where $P(O_m^n|M_k)$ is the probability of observing the feature vectors O_m^n given the allophone model M_k. The arc likelihoods for all the arcs in the syllabic graph can be computed iteratively, as outlined in Appendix C.

In contrast, the exact likelihood score for a path is determined by summing the probabilities of all possible state sequences through the path given the observation sequence. For example, the exact likelihood for the path (2,4,6) is obtained by

$$L_{2,4,6}^M = \sum_{i,j} P(O_1^i|M_2)P(O_{i+1}^j|M_4)P(O_{j+1}^l|M_6) .$$

Note that arc likelihood is computed not only from the observations assigned to the states corresponding to the arc in question, but from the entire observation sequence for the word. In (1), taking the maximum for each i of the probabilities $P(O_1^i|M_1)$ and $P(O_1^i|M_2)$ and for each j of the probabilities $P(O_{j+1}^l|M_6)$ and $P(O_{j+1}^l|M_7)$ ensures that L_4 (the arc likelihood computed for arc 4) is greater than or equal to the exact likelihood score for any possible path through arc 4. The fact that the exact likelihood for any path through a given arc is less than or equal to the arc likelihood of that arc further implies that the minimum arc likelihood along a path is greater than or equal to the exact likelihood of the path. This inequality will be important when we discuss rough path likelihoods below.

Once the arc likelihoods have been computed, the next step is to find the N most likely paths through the syllabic graph ($N \approx 30$). To do this, we need to compute rough likelihoods of partial paths from the arc likelihoods computed above. A simple approximation is to assume that the rough likelihood of the path depends only on the arc likelihoods constituting the path. In choosing the appropriate function of the arc likelihoods, we want the rough likelihood of the path to be as close to the exact likelihood as possible. To make the rough likelihood approach the exact likelihood from above we bound the rough likelihood of a path from below by the exact likelihood score and then minimize it.

Many possible functions are available that guarantee the rough path likelihoods to be greater than the exact likelihoods. Examples of these are the arithmetic mean of the arc likelihoods, their geometric mean, and their minimum along the path. Results obtained with these three functions are compared in the first three rows of Table VII. Of these three, the minimum function gives the best recognition performance. This is because for positive numbers (i.e., likelihoods)

$$\text{arithmetic mean} \geqslant \text{geometric mean} \geqslant \text{minimum} .$$

Since all of these functions give rough path likelihoods greater than or equal to the exact likelihood, it is the minimum function that gives the rough path likelihood closest to the exact likelihood score. This accounts for the result that the recognition rate, using the minimum function to compute the rough path likelihoods, is the highest.

The performance of the syllabic graph search can be further improved if we can compute path likelihoods smaller than the minimum of the constituent arc likelihoods. To find such a function, let us consider Fig. 8 used in computing L_4, the arc likelihood of arc 4. From the discussion of arc likelihoods, we know that L_4 must be greater than or equal to the exact likelihood of any path which includes arc 4. For example, the exact likelihoods for paths (1,4,6) and (2,4,6) must be less than or equal to L_4. Note that if arc 4 has the smallest arc likelihood of all the arcs 1–4 and 6, then the rough likelihoods for paths (1,4,6) and (2,4,6) using the minimum function are equal, even though L_2 may be significantly lower than L_1. We can rectify this inconsistency by multiplying the rough likelihood for path (2,4,6) by the ratio of arc likelihoods L_2/L_1. In general, more than two branches may terminate at node 2. In that case, we take the ratio between L_2 and the arc terminating at node B having the maximum arc likelihood. Similarly, we can multiply the rough likelihood of path (2,4,6) by the ratio of the arc likelihoods of arc 6 and

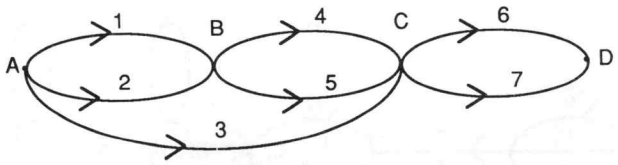

FIG. 7. Sample network illustrating the search algorithm. Each arc represents an allophone model.

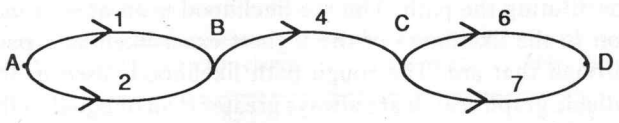

FIG. 8. Subnetwork used to compute the likelihood for arc 4.

TABLE VII. *Percent correct* and *average rank of correct word* for the 75 000-word recognizer on two different speakers using test sets of 714 words for each speaker. We compare different methods for scoring complete paths through the network. All words have equal prior probabilities.

Scoring method	Speaker 1		Speaker 2	
	% correct	average rank	% correct	average rank
arithmetic mean	50%	7.6	52%	11.2
geometric mean	66%	4.0	68%	5.3
minimum	68%	4.1	71%	4.5
ratio likelihood	72%	2.8	77%	2.1
exact likelihood scores	75%	2.3	83%	1.5

the arc emanating from node C having the maximum arc likelihood. In other words, the rough likelihood for the path (2,4,6) is given by

$$L_{2,4,6} = L_4[L_2/\max(L_1,L_2)][L_6/\max(L_6,L_7)], \quad (2)$$

where L_n is the likelihood of arc n. The likelihood function (2) is always less than or equal to the rough path likelihoods computed using the minimum function described above. We call (2) the *ratio likelihood* function.

We have found empirically that the rough likelihoods of complete paths using the ratio likelihood function are always greater than or equal to the exact likelihoods for the same paths; therefore, the reduction of the scores by the ratio of the arc likelihoods is reasonable. The performance of the recognizer using the ratio likelihood function is shown in the fourth row of Table VII and represents a clear improvement over that using the minimum function.

Once the rough path likelihood function for scoring the partial paths has been established, it can be used as the heuristic function in the A^* algorithm to carry out the search. We can verify that the rough likelihood fulfills the admissibility condition for the heuristic function of the A^* algorithm, i.e., the rough likelihood of a partial path is always greater than or equal to the rough likelihood of a complete path containing it. (In fact, the rough likelihood of a partial path is greater than or equal to any extension of that path.)

We will now outline the actual procedure using the network of Fig. 7 as an example. The following steps are performed iteratively for the A^* search.

(1) Put the null path corresponding to node A on the input list, with its likelihood set to 1.0. Initialize the output list to be empty.

(2) Remove the partial path in the input list with the highest likelihood and expand all 1-arc extensions of this partial path.

(3) Check all path extensions with the lexical tree for validity. Put the valid partial paths in the input list and the valid complete paths in the output list.

(4) Check the condition for termination. The search terminates if there are at least N complete paths in the output list, with the likelihood of the Nth path greater than the highest likelihood partial path in the input list. If the terminating condition is not satisfied, go to step (2).

Generally, the recognition performance for words improves with increasing syllable count. Therefore, the total number of word hypotheses N is chosen to be as large as 40 for monosyllabic words and as small as 25 for words four or more syllables long. These values ensure that the likelihood of not having the correct word among the decoded word hypotheses is very small.

TABLE VIII. Computation time for the syllabic graph search algorithm depending on N, the average number of phoneme sequence hypotheses generated for each syllable count hypothesis. The test set size is 714 words.

N	Computation time	Words missed
5	41 s	7.4%
15	52 s	3.9%
30	65 s	1.3%

The computation time for the syllabic graph search algorithm varies with the syllable count of the word and with N, the number of hypotheses generated. We estimated the computation time for generating different numbers of hypotheses on a test set of 714 words, allowing on average 2.3 syllable count hypotheses per word. All the computing times were measured on the VAX 8600, a 4 MIPS machine. The average computation time for generating N hypotheses per syllable count hypothesis is shown in Table VIII. Use of approixmately 30 word hypotheses ($N = 30$) per syllable count hypothesis limits the computing time to 65 s per word, while keeping the number of correct word hypotheses missed by the search algorithm to less than 2%. The computation time of 65 s can be broken down into 7 s for estimating the Gaussian probabilities, 10 s for estimating the arc likelihoods, and 48 s for the graph search to generate the hypotheses for the average 2.3 possible syllable counts for each unknown. Time required to compute the exact likelihood for 70 phoneme sequences per unknown is 2.7 s. In contrast, computing the exact likelihoods for all the 126 445 possible transcriptions in the dictionary would require over 80 min. Thus a total processing time of 68 s for fast search and reduced scope exact likelihoods represents a factor of 70 or better computational savings.

The recognition performance for the syllabic graph search algorithm is compared against the exact likelihood scores in Table VII. The recognition performance improves from 72% to 75% (speaker 1) and from 77% to 83% (speaker 2) when we use the exact likelihood scores for ranking the phoneme sequence hypotheses. The average rank for the correct word improves from 2.8 to 2.3 (speaker 1) and from 2.1 to 1.5 (speaker 2). These results indicate that the assumptions required for the syllabic graph search impose a small performance penalty. If the fast search is followed by a maximum likelihood search over the top candidates, the performance penalty is avoided, yet the advantage due to the reduction in computation is retained.

IX. CONCLUSIONS

We have proposed a new technique to generate a list of word hypotheses from the acoustic information for a spoken word. An algorithm is introduced for estimating the number

of syllables in the word that reduces the lexicon of words to much smaller sublexicons—one for each of the hypothesized syllable counts. A second algorithm implements a graph search through a network appropriate to words with the given number of syllables. The syllabic graph search is made feasible by assigning to each arc of the network a likelihood that is independent of the context. Context independence allows the rough likelihood for a path in the syllabic graph to be rapidly computed as a function of these arc likelihoods, resulting in a fast search. A factor of 70 reduction in computation is achieved. The recognition results for the syllabic graph search algorithm are close to those obtained using the exact likelihood estimates.

ACKNOWLEDGMENT

This work was supported by the Natural Sciences and Engineering Research Council of Canada.

APPENDIX A

The complete list of the allophone models used in this article is as follows. We have used the IPA symbols to represent the phonemes. Numbers in parentheses represent the number of states in the model for that allophone. aj(10), aw(5), ɔj(5), i(10), ɪ(6), e(10), ɛ(6), æ(6), ɑ(6), ʌ(10), ʊ(10), u(5), o(10), ɔ(5), ə(5), $æ^{\perp}$(6) (raised or tense æ), l (5) (includes both syllabic and postvocalic), r(5) (includes both syllabic and postvocalic), m(5), n(5), g(5), tʃ(5), dʒ(3), j(5), l(5) (only prevocalic), r(5) (only prevocalic), w(5), f(5), v(5), θ(5), ð(5), s(5), z(5), ʃ(5), ʒ(5), h(6), p(5), b(4), t(5), d(4), k(5), g(4), *aspiration*(4), *breath*(4).

APPENDIX B

Some examples of texts used as test sentences are as follows. They are shown here as they are spoken. Note that words like *April* (text 4) do not need to be explicitly capitalized since they are always written with a capital letter. The same applies to sentence initial words. However, *Sorrow* in the middle of a sentence (text 4) must be explicitly capitalized: *capital sorrow.*

Text 1:

begin paragraph some airport immigration officers said the real reason for what they described as a lax and haphazard approach is that their superiors are second hyphen guessing politicians and seem more concerned about not being accused of racism toward a particular nationality than about national security period begin paragraph open quote but just wait until something happens and it's traced to the practice then the ellipsis will hit the fan comma close quotes one enforcement officer said period

Text 2:

his future assignments will involve the research of new algorithms and heuristics to model and solve new networking problems introduced by integrated network technologies period his strong background in mathematics is sure to be an asset to the department period

Text 3:

doctors say the most disturbing trend they have noticed in the past few months is the increase in the number of heart hyphen attack victims and other seriously ill patients who turn up in taxis or on foot instead of in ambulances where they belong period that's because until last month comma hospitals were turning away ambulances when their wards became too full comma except in critical cases comma so patients were unable to choose their hospitals period

Text 4:

begin paragraph why are you vexed comma lady question-mark why do you frown question-mark here dwell no frowns comma nor anger comma from these gates capital sorrow flies far colon capital see comma here be all the pleasures capital that fancy can beget on thoughts comma when the fresh blood grows lively comma and returns capital brisk as the April buds in primrose season period and first behold this cordial julep here comma capital that flames and dances in his crystal bounds capital with spirits of balm and fragrant syrups mixed period

APPENDIX C

We will illustrate iterative computation of the arc likelihoods for all the arcs in the syllabic graph using the network of Fig. 7. An example for computing the arc likelihoods is given by Eq. (1), which gives the arc likelihood for arc 4. To compute the arc likelihoods for all the arcs, we first compute an l-dimensional vector for each arc in the network. These vectors are computed sequentially, starting at the initial node and ending at the final node. We present the computational steps in the order of execution.

For the network shown in Fig. 7, we compute seven vectors, one for each branch in the network. The first three vectors are the values of $P(O_1^i|M_1)$, $P(O_1^i|M_2)$, and $P(O_1^i|M_3)$, for $i = 1,...,l$. The next two vectors are the values

$$x(j) = \sum_i \max\left[P(O_1^i|M_1), P(O_1^i|M_2)\right] P(O_{i+1}^j|M_4) ,$$

$$y(j) = \sum_i \max\left[P(O_1^i|M_1), P(O_1^i|M_2)\right] P(O_{i+1}^j|M_5) ,$$

for $j = 1,...,l$. The final two values are

$$\sum_j \max\left[x(j), y(j), P(O_1^j|M_3)\right] P(O_{j+1}^l|M_6) , \quad \text{(C1)}$$

$$\sum_j \max\left[x(j), y(j), P(O_1^j|M_3)\right] P(O_{j+1}^l|M_7) . \quad \text{(C2)}$$

Equations (C1) and (C2) correspond to the arc likelihood of arcs 6 and 7, respectively. In general, the last several values computed correspond to the arc likelihoods for the arcs ending in the final node of the network.

To compute the arc likelihoods for the rest of the arcs, we need another set of l-dimensional vectors, one for each node in the network (except for the initial and final nodes). These vectors are computed starting at the final node and working backward to the initial node. For the network of Fig. 7, we need two vectors. We first compute the vector associated with node C. This vector is given by

$$c(j) = \max\left[P(O_{j+1}^{l}|M_6), P(O_{j+1}^{l}|M_7)\right],$$

for $j = 1,...,l$. The vector corresponding to node B is given by

$$b(i) = \max\left(\sum_j P(O_{i+1}^{j}|M_4)c(j), \sum_j P(O_{i+1}^{j}|M_5)c(j)\right),$$

for $i = 1,...,l$. The arc likelihood for each arc is computed as the inner product of the vector corresponding to the arc with the vector corresponding to the end node of this arc.

[1]In subsequent publications, the IBM group uses a 20 000-word vocabulary (Averbuch *et al.*, 1987).

[2]The terms graph and arc in graph theory (Roberts, 1984) can be equated to the terms network and transition (or branch), respectively.

[3]Since we are not using a language model, homophone confusions are not counted as errors.

[4]The weights W_j, $j = 1,...,24$ are (0.0016, 0.0256, 0.1296, 0.4096, 1.0,...,1.0).

[5]There is no connection between the ten states of a syllable model and the maxium syllable count of ten. We observe a small degradation in recognition performance if we use five states for each syllable model rather than ten.

Averbuch, A., Bahl, L., Bakis, R., Brown, P., Daggett, G., Das, S., Davies, K., Gennaro, S. De, Souza, P. de, Epstein, E., Fraleigh, D., Jelinek, F., Lewis, B., Mercer, R., Moorhead, J., Nadas, A., Nahamoo, D., Picheny, M., Shichman, G., Spinelli, P., Compernolle, D. Van, and Wilkens, H. (**1987**). "Experiments with the Tangora 20 000 word speech recognizer," in *Proceedings of the 1987 IEEE International Conference on Acoustics, Speech, and Signal Processing* (IEEE, New York), pp. 701–704.

Bahl, L. R., Jelinek, F., and Mercer, R. L. (**1983**). "A maximum likelihood approach to continuous speech recognition," IEEE Trans. Pattern Anal. Machine Intell. **PAMI-5**(2), 179–190.

Davis, S. B., and Mermelstein, P. (**1980**). "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," IEEE Trans. Acoust. Speech Signal Process. **ASSP-28**(4), 357–365.

Duda, R. O., and Hart, P. E. (**1973**). *Pattern Classification and Scene Analysis* (Wiley, New York).

Francis, W. N., and Kucera, H. (**1979**). *Manual of Information to Accompany a Standard Sample of Present-day Edited American English, for Use with Digital Computers* (Department of Linguistics, Brown University, Providence, RI).

Gupta, V., Lennig, M., and Mermelstein, P. (**1987**). "Integration of acoustic information in a large vocabulary word recognizer," in *Proceedings of the 1987 IEEE International Conference on Acoustics, Speech, and Signal Processing* (IEEE, New York), pp. 697–700.

Jelinek, F. (**1976**). "Continuous speech recognition by statistical methods," Proc. IEEE **64**(4), 532–556.

Jelinek, F. (**1985**). "Markov modeling of text generation," in *The Impact of Processing Techniques on Communications Series E: Applied Sciences,* edited by J. K. Skwirzynski (Kluwer Academic, The Netherlands), Vol. 91.

Levinson, S. E., Rabiner, L. R., and Sondhi, M. M. (**1983**). "An introduction to the application of the theory of probabilistic functions of a Markov process to automatic speech recognition," Bell Sys. Tech. J. **62**(4), 1035–1074.

Mermelstein, P. (**1975**). "Automatic segmentation of speech into syllabic units," J. Acoust. Soc. Am. **58**, 880–883.

G. & C. Merriam Co. (**1965**). *Webster's Seventh New Collegiate Dictionary.*

Nilsson, N. J. (**1980**). *Principles of Artificial Intelligence* (Tioga, Palo Alto, CA).

Roberts, F. S. (**1984**). *Applied Combinatorics* (Prentice-Hall, Englewood Cliffs, NJ).

Selkirk, E. O. (**1982**). "The syllable," in *The Structure of Phonological Representations,* edited by H. V. Hulst and N. Smith (Foris, Cinnaminson, NJ), pp. 336–383.