Implementation of Viterbi Search Algorithm

Aravind Ganapathiraju

Institute for Signal and Information Processing
Department of Electrical and Computer Engineering
Mississippi State University
Mississippi State, Mississippi 39762
ganapath@isip.msstate.edu

ABSTRACT

Speech Recognition can be treated in a very general sense as a structured search problem. Correct recognition is defined as outputting the most likely word sequence given the language model, the acoustic model and the observed acoustic data. This work involves the implementation of a commonly used search algorithm, Viterbi Search. The implementation uses continuous observation HMMs to represent its word models. The algorithm is provides the most likely word sequence that could have produced the observed acoustic data. The code is object oriented and the structure has been made very generic so as to allow for using other search algorithms such as, Viterbi Beam Search, at a later stage. The design allows for integration of the search engine with various other modules of a speech recognizer including the language model, and the front-end signal processor. For experimentation a small language model has been created and dummy HMM models have been used. The Viterbi algorithm has been found to give the optimum solution to the search problem. It is not efficient in terms of memory. This basic framework will now be used to develop other efficient search algorithms.

1. INTRODUCTION

Considerable progress has been made in the area of speech recognition in the past few years, especially in the area of continuous Large Vocabulary Recognition (LVR). Present day systems can approach performances with word error rate as low as 5%. Their performance is equally commendable even in poor SNR conditions. The key to this improvement can be attributed to developments to all components of a traditional speech recognizer, especially the acoustic modelling. Most LVR systems today are Hidden Markov Model (HMM) [4]based makes the recognition a statistical network searching problem. Some systems also have started using Artificial Neural Networks (ANNs) or a combination of both.

The signal representation of speech has changed and improved over the past decade or so. Systems have now started deviating from the traditional representation of speech as its FFT, to Mel Cepstral based features and Perceptual Linear Prediction (PLP) based features. At the signal processing front sampling frequency, frame durations etc. have also changed from the traditionally used values. At the language modelling front of the speech recognizer, things have become very complex. Recognizers now use Trigram and Bigram models and even as high as 5-gram in some cases. More syntactic and semantic information is being used now. The most important improvements have occurred in the Acoustic Decoding part of the recognizer, also called the Search Engine. The changes to the search engine have made it possible for the speech recognizer to efficiently and accurately decode input acoustic information to its constituent words or sentences. Why is it called the search engine? The recognition process is generally formulated as choosing the most probable path through a large statistical network where the nodes in the network represent a state of the system. The problem with this formulation is that the network is a sparse network, in that there are not many paths in the network which are as probable as the best or the correct path. The network is also a very large network so that exhaustively searching for the best path through the network is impractical. With the advent of better processors we are able to achieve real-time performance by adding some constraints to the search paradigm.

It is this component of the speech recognizer that my current work focuses on. In this paper we discuss the speech recognizer in general. This will be followed by a section on the currently used search algorithms. We then describe the implementation of a specific search strategy, called the Viterbi search which a very simple algorithm compared to its counterparts but, is quite inefficient for real-time large vocabulary recognition purposes. It is a Maximum Likelihood(ML) solution to the problem[1,2].The primarily purpose of this system is to allow using the framework for implementing more efficient algorithms in the future. The implementation makes use of object oriented techniques in order for easy expansion of the system to simultaneously support multiple search algorithms without greatly increasing the memory requirements. This is a necessary feature in systems which are under development so that

performance of the system when various algorithms are used has to be compared before deciding on the final implementation of the speech recognizer. The later part of the paper will focus on the results achieved on the synthesized data.

2. SYSTEM OVERVIEW

The ultimate goal of continuous speech recognition is to correctly decode a set of words or sentences given the input acoustic sequence. evaluation which will correctly achieve this decoding always, is not possible. Therefore the goal of the recognizer is modified to be the most probable word sequence which may have produced the given acoustic data. This leads to the following definition of continuous speech recognition.

$$p(\hat{W}/A) = \max_{W} p(W/A) \qquad (1)$$

where A is the given acoustic sequence, W is the inventory of word models the recognizer has in store and W is the derived word sequence.

By applying Bayes rule to the above equation of speech recognition we can re-write the above equation as

$$P(W/A) = \left(\frac{P(W) \times P(A/W)}{P(A)}\right) \qquad (2)$$

This equation allows us to express $P(W/A)$ in a way we can compute. $P(W)$ is computed from the language model and is the probability of the word occurring in a given context. $P(A/W)$ is the conditional probability of the acoustic sequence A given the word sequence W. This conditional probability is computed from the acoustic models we have for each word. $P(A)$ is the probability of the acoustic sequence A. This probability is not always necessary since this the same for all complete decodings of the A[16].

2.1. Language Modelling

Figure 1 shows the schematic of a top-down approach to speech recognition. In this case the word hypothesizer is guided by the sentence hypothesizer. As a frame is processed, each active sentence hypothesis asks for data as needed. The sequence of data requests typically begins with a request for word hypothesis. These requests for word hypothesis in turn ask for a phoneme hypothesis. The process terminates with a request for a frame of data. Each level applies the constraints of a grammar like structure to the next lower level of data representation. In a simple sense, the recognizer would abandon pursuing a given sentence if the likelihood of the phoneme or word string is getting low. This information is given by the LD. The phoneme or the word strings are given by the AD.

Another approach to the problem is the bottom up approach. In this approach the grammar aids in recognition by not allowing symbol combinations that are not in the language. In this case AD starts hypothesizing phonemes, guided at each step by the appropriateness of the string that is being created. Those strings that are less likely are abandoned before the end of utterance is reached. The main disadvantage with the bottom-up approach is that a sentence cannot be recognized unless each of its symbols is recognized by the AD. The disadvantage of top-down approach is that the language should be highly constrained since all possible sentences need to be hypothesized.

The previous discussion describes a high level viewpoint of the recognition process. The lower level and of the decoding process involves the signal processing and the search. The next section discusses in brief the signal processing side of the problem.

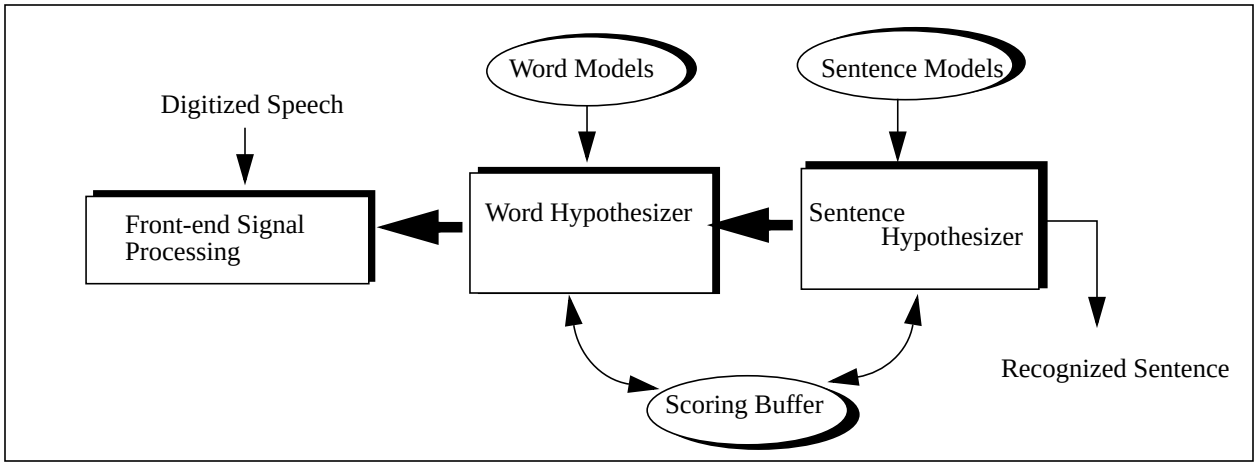

Figure 1. A simple schematic of a speech recognizer based on the top-down approach

2.2. Signal Processing

The most important function of the signal processing front-end is to convert the incoming speech into blocks of data and allow for a compact representation of this data. This form of compact representation results in what is called the feature vector. The input speech is divided into frames of data of duration,typically,10ms. These frames are normally overlapped by windows of duration,typically,35ms. This allows for removing the artifacts of truncation resulting from the conversion into frames. Typically a Hamming window is used. From these frames of data their spectral estimates are derived using Linear Prediction or Fourier analysis. A number of other transformations are also applied to generate the final feature vectors. These transformation procedures include pre-whitening of the features, which does the decorrelation of the features and, channel adaptation which tries to remove the artifacts of the channel from the feature vector. Most systems use the Mel-frequency cepstral coefficients (MFCCs) and their time derivatives. The reason for going for Mel-frequency based features is that research has proved that the human perception of speech follows a logarithmic scale of frequency rather than a linear scale. The fourier spectrum is converted to a Mel-scale by passing the fourier coefficients over a set of triangular frequency bins spaced on a log. scale. This is followed by a DCT operation which has the effect of compressing the spectral information in the lower order coefficients and it also decorrelates them[19]. Some systems also include the energy and difference energy. The need for having the time differences of MFCCs in the feature vector is to accommodate for longer context than just that of the current frame. Since most of the speech recognition techniques assume stationarity of speech in the frame of observation and uncorrelatedness between adjacent feature vectors, which is a poor assumption, the time differences, hopefully, compensate for this assumption. Typically a feature vector has about 40 features. To start with about 60 features are extracted per frame of data.

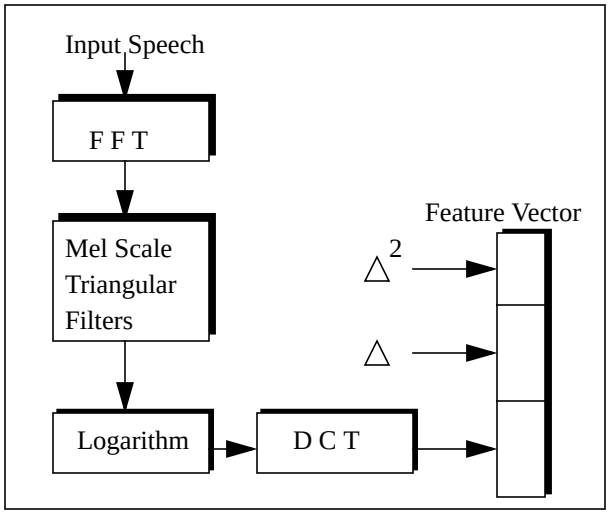

Figure 2 Generation of a feature vector

On these features a principal components analysis is performed to reduce the feature set to about 40 features. A typical procedure for creating the feature vector incorporated in the HTK recognizer is shown in figure 2.

2.3. Acoustic Modelling

The purpose of the acoustic models is to provide a method of calculating the likelihood of any vector sequence **Y** given word **w.**For small vocabulary systems, and digit recognition systems, we can have whole word models and achieve good performance. But, in case of Large vocabulary systems when we speak of several thousand words, each spoken in a number possible ways, it becomes impractical to have a model for each of the words. Thus we go in for modelling sub-word units like phones, phonemes, triphones, etc. There are several advantages and disadvantages of each of these sub-word units. The most commonly used sub-word units in present day systems are the context dependent phonemes, and the triphones. The disadvantage of using sub-word units is that they cannot capture the long term context of the word as in digit recognition.

Early systems used the Dynamic Time Warping techniques for decoding where each word was represented by a template and an optimal alignment was desired for to get the most probable sequence of words. Most of the present day systems use Hidden Markov Models (HMM) for modelling words and sub-word units. In recent years some systems have started using neural networks to model their acoustic models.

The next section discusses the HMM technology in general and also the issues in choosing the appropriate models.

3. HIDDEN MARKOV MODELS

A HMM is used as a model for words or sub-word units in speech recognition. HMM is a doubly stochastic finite state automaton[6,7,24]. What this means is that the transitions from one state to another in the HMM are governed by a stochastic process as also are the output probabilities associated with each state in the HMM. Since the HMMs are finite state in nature, the whole recognition process gets in turn converted to a search problem in a massive network of states. The following example will illustrate the components of a HMM and their use in the speech recognition paradigm.

3.1. Characteristics of an HMM

Why the name ‘Hidden Markov Model’? The HMM can be concisely defined as a doubly stochastic interconnection of states. The present state depends only on the previous state and the present input and need not know anything prior to that. This justifies the name

Markov. The sequence of states is not explicitly seen in an HMM. Theoretically speaking, a HMM can generate all possible state sequences, though with different probabilities. It is the search paradigm which solves this problem by finding the likelihood of a certain sequence given the HMM and an input acoustic sequence. Figure 3 shows the structure a simple 5 state HMM There are explicit start and stop states in this model which need not be the case in practice. At each observation, state transition is assumed to occur. This transition is governed by a state transition probability a_{ij}. There is associated with each state an output density function $b_j(y_t)$, which gives the probability of an output vector being emitted at that state. There is also an initial state probability vector. A matrix can be created with the transition probabilities from one state to another called the State Transition Matrix. Taken together, the state transition matrix and the initial state probability vector specify the probability of residing in any given state at any time.

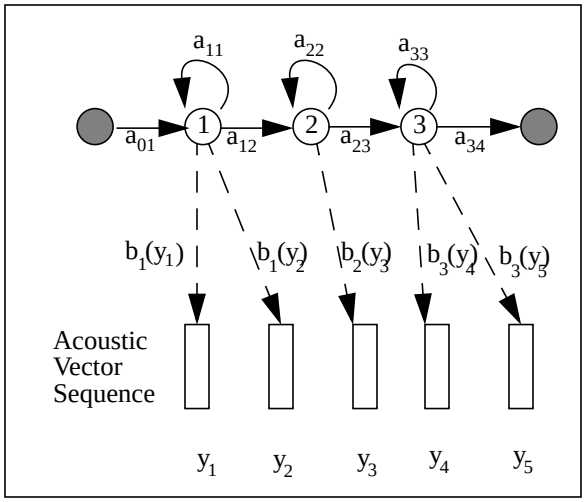

Figure 3 A simple HMM

Different topologies are in use for HMMs. The choice of the topology depends primarily on the type of acoustic data we want to recognize. The above example is a left-to-right model with no skip states.

HMMs can also be characterized as Continuous or Discrete depending on the type of output densities they are associated with. Discrete HMMs were in use for most of the 80's but now the focus has shifted to Continuous HMMs since they have been found to model phonemes better than Discrete HMMs. The penalty one pays for this shift, is the large increase in the number of parameters the recognition has to deal with during the training phase as well as the recognition phase. In continuous density HMMs each state has a mean vector and a Covariance matrix associated with it. The mean vector and the covariance matrix together are used to describe the probability of an output governed by a Multivariate Gaussian distribution. Equation 3. describes the output distribution.

$$\langle f_{\underline{y}/\underline{x}} \rangle \langle \xi | i \rangle = N(\xi; \mu_i, C_i) \qquad (3)$$

In the above equation $\underline{y}$ is the observation vector and $\underline{x}$ is the state sequence. In general single Gaussian pdf is not sufficient to model the observation at a state. Most systems now have a Gaussian mixture density instead, so as to better model the modalities of input speech. Some of such modes could be a male/female distinction, and distinction based on dialects. This topic will be later addressed in the paper.

3.2. Training the HMM

We have to train the HMM to correctly represent its designated word or other utterance like the phoneme or triphone etc. This is more difficult than the actual recognition of an utterance. The performance of the recognizer is very heavily dependent on the training process. There are two main procedures for this purpose. The Forward-Backward (Baum-Welch) algorithm and the Viterbi algorithm[19-21]. The training of the HMMs is a training-by-recognition process. The HMM models are initialized by some seed models. The recognizer is then forced to recognize the known utterance by reestimating its parameters. The use of reestimation formulae will ultimately lead to a Model which represents a local maximum of the likelihood $P\langle y|M \rangle$. However, finding a good local maximum depends rather critically on the initial estimate of the matrix parameters for each state[19]. This problem can be circumvented by using good seed models which are got by actually hand-excising a representative token and creating one state for each frame in the token. Transition probabilities are initialized to favor a path through the model consisting entirely of progressing transitions. The representative token is typically selected based on its duration. A standard thumb of rule being that the number of states should be equal to the average duration in frames of the recognition unit[19].

4. SEARCH ALGORITHMS

Having talked about the basic components of the Speech Recognizer we now introduce the various available search techniques. The search process is the most important and challenging part of the speech recognizer. Most of today's speech research is aimed at achieving better performances by formulating more efficient search techniques. In continuous speech recognition the likelihood of the observed data is computed by scoring it on all the feature models. The search paradigm then chooses a speech pattern with the highest likelihood. The number of possible hypotheses grows exponentially

with the number of feature HMMs and imposes heavy constraints on the computation and storage requirements. Therefore intuitively obvious techniques such as an exhaustive search are not at all practicable and strategies that save on computation by modifying the search space are vital. These may sometimes cause the system to make suboptimal choices but the available suboptimal algorithms are known not to significantly affect the accuracy of recognition. The following section briefly discusses the various commonly used search paradigms.

4.1. Viterbi Search

The Viterbi algorithm is an efficient algorithm for finding the optimal solution [2]. It is based on the DP principles postulated by Bellman. [25,33] and has been extensively used in DP based speech recognition. It imposes the restriction that the cost, or probability of any path leading can be recursively computed as the sum of the cost in making a transition from the previous state to the current state. This constraint goes well with the time constraint imposed by the Hidden Markov.

The Viterbi search is a time synchronous search strategy. That is, at a given time, each partial solution W' accounts for the same portion of the acoustic sequence, namely y^i_1. Thus partial hypothesis can be directly compared without any complicated evaluation functions.

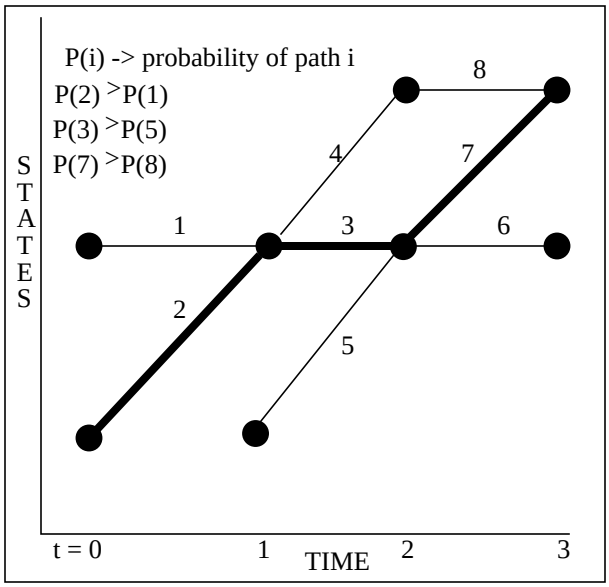

Figure 4. The shaded path is the best partial path through the network of states according to the Viterbi algorithm.

In the Viterbi search, both within-word and between word transitions are considered in a unified framework. This has the advantage of efficiently evaluating partial solutions due to ita dynamic programming characteristics. The partial solutions, in this case, are recombined using the Max. function instead of the Sum function[9]. This is computationally efficient because Max. evaluation is faster than the Sum function evaluation. However it is inconsistent with the summing operation used in the forward-backward algorithm.

4.2. Viterbi Beam Search

The Viterbi search takes advantage of the dynamic programming techniques that reduce the problem size considerably compared to an exhaustive search. However, the problem is still too large when complex finite state grammars or large vocabulary systems are considered. By observing that most partial hypotheses at a state have zero or near-zero probability at a given time, we can modify the viterbi algorithm to consider only states which have a probability of ε less than the best scoring state at the time in consideration, where ε defines the *beam* width of the search. Only a minor modification to the Viterbi algorithm results in the Beam search[36,37].

The clear advantage of the Viterbi beam search is that the problem size is reduced and the path merging function is the *Max* function which has significant computational savings over the *Add* function when applied to log probabilities[9]. The advantage of a dynamic beam heuristic is that it will consider only those hypothesis which are good relative to the best hypothesis. This heuristic therefore allows the search to consider many hypotheses when there is no clear best hypothesis. Alternately, when there is a clearly best hypothesis, only a few alternate hypothesis need to be maintained by the search engine.

4.3. Stack Decoding

The stack decoding algorithm is similar to the A* search used in artificial intelligence[5]. Stack decoding is a state-synchronous approach as compared to the Viterbi algorithm which is a time-synchronous approach. Stack decoding constructs a search tree from state graph S that is described by the language model. The states in the graph represent the abstract states in the language model, and the branches in the search tree are the words that cause transition from one language state to another.

An important advantage of the stack decoding algorithm is its consistency with the forward-backward training algorithm. The disadvantage with the stack decoding algorithm is that an extra function is required for the comparison of hypotheses of different lengths. The basic stack decoding algorithm[11,17,34] can be summarized as follows [35]:

1. Pop the best partial hypothesis from the stack.

2. Apply acoustic and language model fast matches to shortlist the candidate next word.

3. Apply acoustic and language model detailed matches to candidate words.

4. Choose the most likely next word and update all hypotheses.

5. Insert surviving new hypotheses into the stack.

This approach suffers from speed, size, accuracy and robustness, but efficiently combines all information into a one-pass paradigm.

4.4. N-Best Search

The optimal N-best decoding algorithm is quite similar to the Viterbi search[15,26]. The Viterbi search is a simple case of N-best in that it is inherently a 1-best approach. In N-best search all hypotheses within the specified beam are found and all hypotheses with different path histories at each state are kept track of. It then allows only the N-top scoring hypotheses to propagate to the next state. This state dependent pruning is independent of the global Viterbi beam threshold. There are different sources of information which can be used for the purpose of recognition nut each of which is associated with a different cost. A hypothesis which scores the highest given all the knowledge sources gives the optimal solution. This is impractical though because of the large search space. It is therefore advantageous to use the most efficient knowledge sources to find a list of N-top scoring hypotheses. These hypotheses can then be re-evaluated using more complex and expensive knowledge sources.

The N-best paradigm as described above has the disadvantage that it is more partial towards shorter hypotheses. Thus an exact N-best search will require a very large N to find the correct long sentence. Some of the variations of the N-best search are the Lattice N-Best Algorithm, Word-dependent N-Best search and the Forward-Backward search.

In the *Lattice N-Best Algorithm*, a time synchronous 1-best forward-pass search algorithm is used within words and at each frame all the theories and their respective scores are stored in a traceback list. The best score of this frame is sent forward along with a backpointer to the saved list. The N-best sentences are obtained by recursive search through this list. This algorithm is extremely fast but often underestimates or misses high-scoring hypotheses[29].

In the *Word-dependent N-Best search* the algorithm differentiates between hypotheses based on the previous word rather the whole preceding sequence of words.

In the *Forward-Backward search* an approximate time-synchronous search is done in the forward direction to facilitate a more complex and expensive search in the backward direction[30,31]. A simplified acoustic model and a simple language model like the unigram is used for the forward pass. Then a normal within-word beam search is performed in the backward pass to generate the N-best hypotheses list. The backward search scores high on a hypothesis only if there also exists a good forward path leading to a word ending at that time. Figure 5 explains this in a simple schematic.

5. IMPLEMENTATION SPECIFICS

After the brief survey of various search algorithms one can see that Viterbi search is the most simple to implement and efficient too, when dealing with small vocabulary systems like digit recognizers, where we are talking of about 12 word models. The following sections will describe in detail our implementation of the Viterbi algorithm. The main motivation for this implementation is to use this framework to build other more efficient algorithms like the Beam search in the near future. The implementation has kept in mind this fact and also other factors like extensibility of the code to accommodate for various ramifications in HMM based speech recognition technology. The implementation will allow the search engine to be tested in isolation from other components of the speech recognizer as well as in combination with

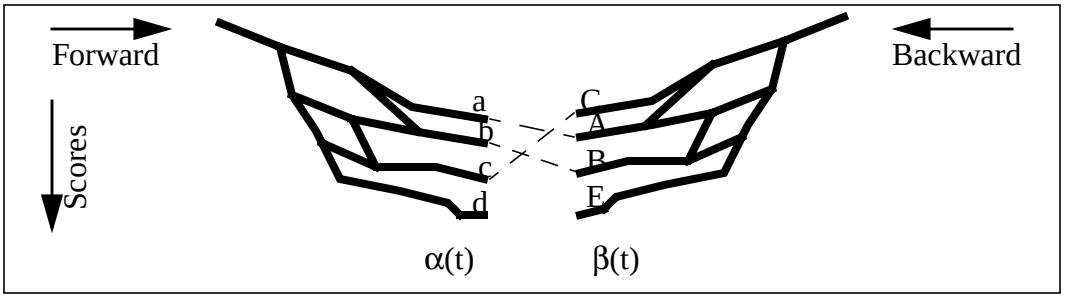

Figure 5 Forward-backward search. Forward and backward scores for the same state and frame are combined to predict final score for each hypothesis

these components.

5.1. The Viterbi Algorithm

The Viterbi algorithm we have used can be summarized as follows:

1. Read the acoustic model specifics and create HMMs to represent each of the word in the vocabulary

2. Read the language model specifics and create a state machine which represents the language model

3. Initialize the system to the start state of the language model at t=0

4. Depending on the language model create a slot in the scoring buffer for the initial state of the words to which the start state can transition to according to the language grammar

5. For each word model that is initialized, look-ahead in the input data and build the trellis by accumulating scores. Constrain the length of a word to safe number of frames (we use 9)

6. At t=1 check if any possible end state occurs. If an end state occurs create a score slot for all initial states of words that the word represented by the end state with the best score, can transition to.

7. Repeat steps 5 and 6 till end of input data is reached at t = T.

8. For all possible end states at t=T, find the end state with the highest score and backtrace through the trellis and find the most probable word sequence.

Figure 6. is shows a schematic of the above described implementation.

For the purpose of implementation of the Viterbi search algorithm we have used an object oriented design methodology. The next few sections deal with objects we created and their features as also their significance to the recognition problem.

5.2. The Datastructures

Most of present day research in speech recognition is driven primarily because of lack of really large portable memory (of the order of 2 GB) which has fast random access. Algorithms and implementations are sought which can achieve decent performance with average

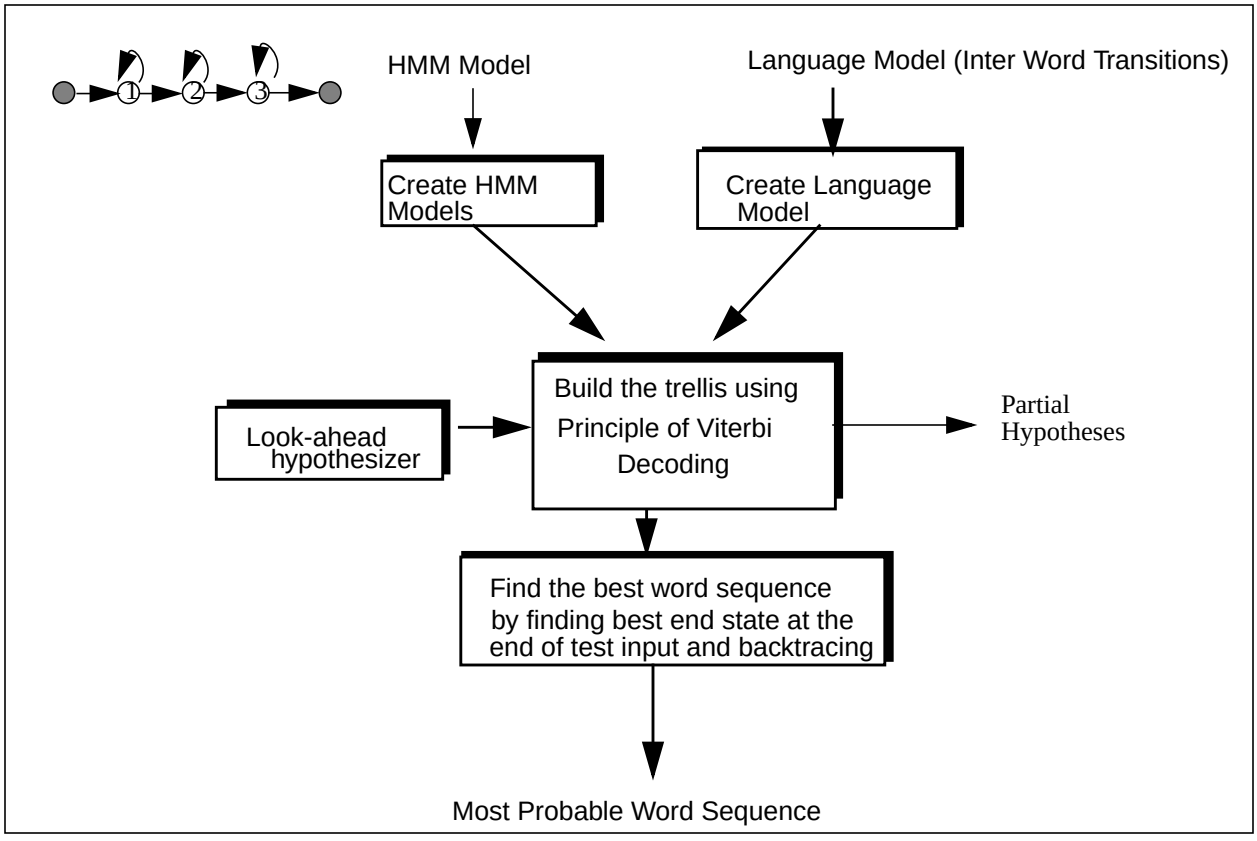

Figure 6. Implementation schematic of a Viterbi decoder for search in speech recognition

memory requirements. This has lead to the importance of choosing the right datastructures in the search process which represent information in a very compact way as well allow for reusability of memory and code structures[8][13].

For the purpose of keeping track of the scores accumulated at each state at each time instant we create an object called the Score_Hyp which carries the following information which can be used in the decoding process.

a) log-likelihood score b) reference time

c) end of word flag d) end of sentence flag

e) back-pointer to parent node and

f) a word history for path or theory identification.

The reference time data helps us in the reuse of the structure. Suppose a Score_Hyp has a time stamp of t. At time t+8 when we need to store score information in a Score_Hyp we could do so by using the Score_Hyp created at time t if the path on which this Score_Hyp falls is found to be inactive. To facilitate easy searching of Score_Hyps we found it more appropriate to have them as a doubly linked list.

The HMM is a stochastic interconnection of states. With each state is associated a mean vector and a covariance matrix when we are dealing with continuous density HMMs. There is also a transition probability associated with each state pair. We found it appropriate to build a HMM also as a Linked list to have a more intuitive representation of the structure. For future applications and development to the system it is necessary to provide for mixture densities too. In most of present day systems, mixtures have become a common practice.

The language model can also be represented as a state machine and hence we have designed the language model also as a linked list of states. Figure 7 shows an example of an HMM and its representation as described above. This representation has the advantage that it is very flexible as far as number of states present in an HMM and also the different transitions. This representation is very intuitive too.

5.3. The Experimental Setup

For the purpose of experimentation with the search engine we designed our own very small vocabulary language model as well as acoustic models. Care has been take to design these test models so as to verify the performance of the search engine in an accurate way.

The vocabulary was chosen to be consisting of four

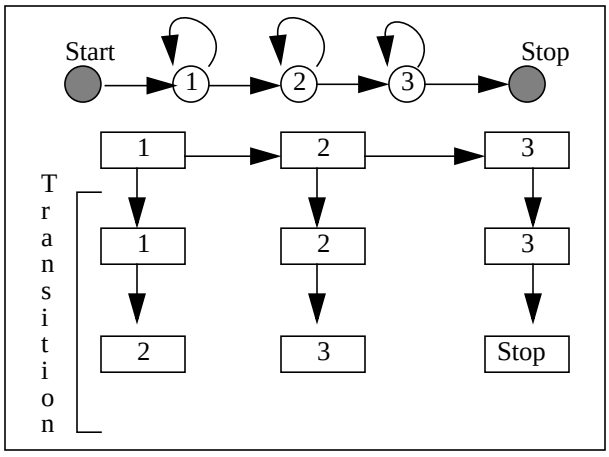

Figure 7. Representation of an HMM as a Linked List

been chosen arbitrarily. There are cases of multiple transitions from one state to another which causes confusion to the search engine during the decoding process. For each word a model has been created. Each word is represented by a three state left-to-right topology without skip states which is one of the most commonly used systems in present day systems. Equal transition probabilities have been chosen.

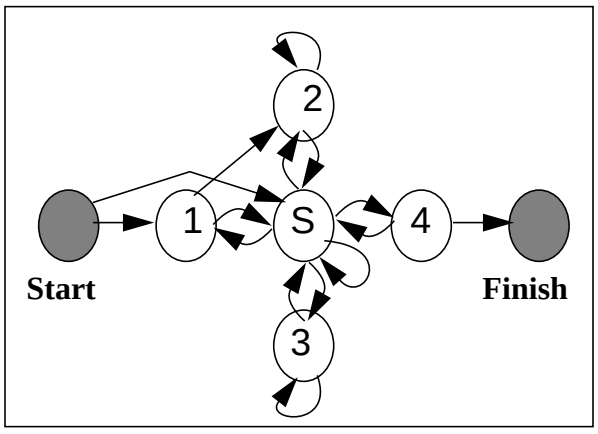

Figure 8 State diagram representation of the experimental language model

3-dimensional mean vectors have been used at each state. For the purpose of scoring we used the log-likelihood probabilities and a euclidean metric has been chosen for scoring the output at each state owing to the simplicity of the computation. In practice this part results in a matrix multiplication and addition process.

Synthetic data is created by adding random noise to the mean vectors at each state in each model and test vectors corresponding to a given model are varying in length too to account for possible self transitions within the model. The maximum duration of a word is assumed to be 9

- utterance constrained to start with silence
- synthetically generated data (with option to add noise)
- Euclidean distance as model scoring criterion
- premise that a word occupies a maximum of 8 frame durations

Reference sentence

<Silence> one two four three <Silence>
6 6 4 5 8 3 (duration in frames)

Recognized Sentence Hypotheses

<Silence> <Silence> one two four <Silence> three <Silence>
Score: 114.37

<Silence> <Silence> one two four <Silence> three <Silence> <Silence>
Score: 111.87

<Silence> <Silence> one two four <Silence> three <Silence>
Score: 115.36

Figure 9. Experimental results on the search engine. Note that the most probable sequence is the one with the highest score. This matches very well with the input synthesized data

frames. Figure 9 shows results of this experimental setup.

6. SALIENT FEATURES OF THE IMPLEMENTATION

The above discussed design has incorporated many of the features which are in use in most present day systems.

Object Oriented Design

The implementation is data driven for most part. Making algorithms datadriven has been a thrust area in speech recognition research for many years now due to the size of the search space. In order to make the recognizer perform in real-time importance has to be given in choosing the data structures for the implementation[8,13]. Most of the present implementation is based on linked lists for more flexibility in the search process [9,13,22,25].

Data driven Implementation

The implementation is made modular to allow for testing various HMM-based applications. The search engine can be easily integrated with other modules of the recognition system (front-end,language-model). The key feature in the design is that the user has control over almost the whole process of the search. The HMM-topology is specified by the user. There is no limit on the size and shape of the structure. The language model is also specified by the user. Incorporating features such as tying of states and mixture distributions is easily done.

7. SUMMARY AND FUTURE RESEARCH

Efficient search techniques and better acoustic models are vital for the improved performance of LVR systems. Most systems still continue to be HMM based though ANN based systems have made inroads into this area. ANN can model non-linearities better than HMMs but these systems have only performed atmost as best as HMM based systems. Most of the techniques discussed in the present work represent state-of-art techniques. Importance has been given to the software structure for efficient memory management which is a core issue in speech recognition. Also, catering to the long term goal of integrating the present search engine with other modules to form a comprehensive LVR system, software has been made very flexible and extensible.

Future research in this area will be based on this existing frame work. Our immediate goal is to extend the system to a Viterbi beam search paradigm. Work towards implementing the Forward-Backward algorithm using the N-best paradigm is under progress. Once this algorithm is functional, results from the different algorithms will be compared. Table 1. shows the search strategies employed by some of the present day LVR systems. The code will allow for the simultaneous use of

these different algorithms. This system will then be integrated with the other modules and testing on real data will be done. Various topologies, tied states, mixture densities etc.[22] will be tested on real data. Acoustic models representing different sub-word units such as phonemes, tri-phones etc. will be tested on.Once this initial testing phase is complete, the recognition performance of the system will be compared with other LVR systems on similar tasks. The recognizer will be initially used for digit-recognition purposes.

LVR System	Search Strategy
BBN/BYBLOS	Multi pass forward-backward search
Dragon Systems	Initial fast match, followed by detailed match: shortlist of words used at each time instant
HTK	Time-Synchronous decoding
MIT Lincoln	Stack Decoder paradigm
IBM-SPHINX	Multi-pass stack decoding

Table 1: Search methods in present day LVR systems

8. ACKNOWLEDGEMENTS

First, and foremost, I thank Dr. Joseph Picone, for all the help, both in terms of patience and time, he has extended throughout this effort. This work was a difficult one especially considering the fact that the system was being built from scratch without the use of any canned speech recognition tools. I also thank, Mr. Neeraj Deshmukh, my colleague at ISIP, for all the support he has extended.

REFERENCES

1. L.R.Bahl, F.Jelinek and R.L.Mercer,"A Maximum Likelihood Approach to Continuous Speech Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Vol. 5, No. 2, pp. 179-190, March 1993.
2. A.Viterbi, "Error Bound for Convolutional Codes and An Asymptotically Optimum Decoding Algorithm," *IEEE Transactions on Information Theory,* Vol. 13, No. 2, pp. 260-269, April 1967.
3. C.H. Lee and L.R. Rabiner," A Frame Synchronous Network Search Algorithm for Connected Word Recognition", *IEEE Transactions on Acoustics, Speech and Signal Processing,* Vol. 37, No. 11,pp.1649-1658, November 1989.
4. B.H.Juang," On The Hidden Markov Model and Dynamic Time Warping For Speech Recognition-A Unified View", *AT&T Technical Journal,* Vol. 63, No. 7, pp. 1213-1243, September 1984.
5. F. Jelinek, "A Fast Sequential Decoding Algorithm Using A Stack", *IBM Journal for Research and Development,* Vol. 13, pp. 675-685, November 1989.
6. L.R. Rabiner and B.H. Juang," An Introduction To Hidden Markov Models", *IEEE Acoustics Speech and Signal Processing Magazine,* Vol.3, No. 1, pp. 4-16, January 1986.
7. L.R. Rabiner," A Tutorial On Hidden Markov Models And Selected Applications in Speech Recognition, *Proceedings of The IEEE*, Vol. 77, No. 2, pp. 257-285, February 1989.
8. H.Ney, D. Mergel, A. Noll, and A. Paeseler, "A Data-Driven Of the Dynamic Programming Beam Search For Continuous Speech Recognition", in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 833-836, Dallas, Texas, April 1987.
9. K.F. Lee, *Large-Vocabulary Speaker Independent Continuous Speech Recognition*: The SPHINX System, Ph.D. Dissertation, Computer Science Department, Carnegie Mellon University, 1988.
10. L.R. Rabiner, J.G. Wilpon, and F.K. Soong, "High Performance Digit Recognition, Using Hidden Markov Models", in *Proceedings IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 119-122, New York, April 1988.
11. L.R. Rabiner, J.G. Wilpon, and F.K. Soong, "High Performance Digit Recognition, Using Hidden Markov Models", I*EEE Transactions On Acoustics, Speech, and Signal Processing*, Vol. 37, No. 8,pp.1214-1225, August 1989.
12. L.R. Bahl, P.F. Brown, P.V. De Souza, and R.L. Mercer, "Acoustic Markov Models Used In The Tangora Speech Recognition System", *IEEE International Conference On Acoustics, Speech, and Signal Processing*, pp. 497-500, April 1988.
13. H. Ney," Dynamic Programming Speech Recognition Using A Context-Free Grammar", *IEEE International Conference On Acoustics, Speech, and Signal Processing*, pp. 3.2.1-3.2.4, April 1987.
14. L.R.Rabiner, B.H.Juang, S.E. Levinson, and M.M. Sondhi, "Recognition Of Isolated Digits Using Hidden Markov Models With Continuous Mixture Densities", *AT&T Technical Journal*, No. 64, pp.21-31, 1985.

15. R. Schwartz,and Y.L.Chow, "The Optimal N-Best Algorithm: An Efficient Procedure For Finding Multiple Sentence Hypothesis", *IEEE International Conference On Acoustics, Speech, and Signal Processing*, pp. 81-84, April 1990.
16. J.W. Picone, "Signal Modelling Techniques in Speech Recognition," *Proceedings of the IEEE*, vol. 81, no. 9, pp. 1215 - 1247, Sept. 1993.
17. D.B.Paul," An Efficient A* Stack Decoder Algorithm for Continuous Speech Recognition with a Stochastic Language Model," *International Conference On Acoustics, Speech, and Signal Processing* 1992, vol. 1, pp. 25 - 28.1992.
18. Code of "*Hidden Markov Model for Automatic Speech Recognition*", Cambridge University Speech Group,U.K.
19. J.W. Picone," Continuous Speech Recognition Using Hidden Markov Models," *IEEE Acoustics, Speech, and Signal Processing Magazine*, pp. 26-41, July 1990.
20. J.R. Deller, J.G. Proakis, and J.H.L. Hansen," *Discrete Time Processing of Speech Signals*" Macmillan Publishing, 1993, New York.
21. L. R. Rabiner, B. H. Juang, "*Fundamentals of Speech Recognition*." New Jersey: Prentice Hall, 1993.
22. J.K. Baker, "The Dragon System - An Overview", *IEEE Transactions On Acoustics, Speech, and Signal Processing*, Vol. 23, pp.24-29, Feb. 1975.
23. L.R. Rabiner, J.G. Wilpon, and B.H.Juang, "A Performance Evaluation Of a Connected Digit Recognizer", in *Proceedings International Conference On Acoustics, Speech, and Signal Processing* 1987, pp. 101-104, Dallas, Texas, 1987.
24. S.E. Levinson, "Continuously Variable Duration Hidden Markov Models For Automatic Speech recognition", *Computer, Speech, Language*, Vol.1, No.1, pp. 29-45, March 1986.
25. C.S. Myers, and S. E. Levinson, "Speaker Independent Connected Word Recognition Using a Syntax Directed Dynamic Programming Procedure," *IEEE Transactions Acoustics, Speech, and Signal Processing*, Vol. 30, pp.561-565, August 1985.
26. Schwartz R.M. and Austin S.,"Efficient, High-Performance Algorithms for N-Best Search", in *Proceedings of the DARPA Speech and Natural Language Workshop*, pp. 6-11, June 1990.
27. L.A. Liporace, "Maximum Likelihood Estimation for Multivariate Observations of Markov Sources", *IEEE Transactions On Information Theory*, Vol. 28, No. 5, pp. 729-734, 1982.
28. L.R. Bahl, P.F. Brown, P.V. De Souza, R.L. Mercer, "A Tree Based Statistical Language Model for Natural Language Speech Recognition., *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. 37, No. 7, pp. 185-189, 1989.
29. L.R. Bahl, R.L. Mercer, S.V. De Gennaro, P.S. Gopalakrishnan," A Fast Approximate Acoustic Match For Large Vocabulary Speech Recognition", I*EEE Transactions Speech and Audio Processing*, Vol. 1, No. 1, pp. 59-67, 1993.
30. R.M. Schwartz and S.Austin. "The Forward-Backward Search Algorithm for Continuous Speech recognition, *Proceedings International Conference on Acoustics, Speech, and Signal Processing*, S10.3, Toronto, 1991.
31. L.Gillick, and R. Roth, "A Rapid Match Algorithm For Continuous Speech Recognition," *Proceedings of the DARPA Speech and Natural Language Workshop, Hidden Valley*, Pennsylvania, pp. 170-172, June, 1990.
32 L.R. Bahl, R.L. Mercer, and P.S. Gopalakrishnan," A tree Search Strategy For Large Vocabulary Continuous Speech Recognition", *Proceedings International Conference on Acoustics, Speech, and Signal Processing*, Vol. 1, pp. 572-575, Detroit, 1995.
33. R.E. Bellman, *Dynamic Programming*, Princeton University Press, Princeton, New Jersey, 1957.
34. D.B. Paul, "The Lincoln Large-Vocabulary Stack Decoder Based Continuous Speech Recognizer", *Proceedings International Conference on Acoustics, Speech, and Signal Processing*, pp. 374-379, April 1993.
35. N. Deshmukh and J.W. Picone, "Methodologies For Language Modeling And Search In Continuous Speech Recognition", *Proceedings IEEE Southeastcon 95*, pp. 192-198, 1995.
36. R. Haeb-Umbach, and H. Ney, "Improvements In Time-Synchronous Beam Search for 10000-Word Continuous Speech Recognition", *IEEE Transactions Speech and Audio Processing*, Vol. 2, pp. 353-356, 1994.
37 V. Steinbiss, "Sentence-Hypothesis Generation in a Continuous Speech Recognition System", *Proceedings Conference On Speech Communication and Technology*, Vol. 2, pp. 51-54, Paris, 1989.