EFFICIENT SEARCH ALGORITHMS FOR LARGE VOCABULARY CONTINUOUS SPEECH RECOGNITION

Neeraj Deshmukh

Institute for Signal and Information Processing (ISIP)
Department of Electrical and Computer Engineering
Mississippi State University, Mississippi State, MS 39762
deshmukh@isip.msstate.edu

ABSTRACT

Automatic speaker-independent speech recognition has made significant progress from the days of isolated word recognition. Today state of the art systems are capable of performing large-vocabulary continuous speech recognition (LVCSR) over complex domains such as news broadcasts and telephone conversations. A significant contribution to this advancement in technology is due to the development of search techniques that support efficient, sub-optimal decoding over large search spaces and complex statistical models. Moreover, these decoding strategies are capable of dynamically integrating information from a number of diverse knowledge sources to determine the correct word hypothesis.

In this project we propose to implement two major classes of such efficient evaluation algorithms viz. multipass N-best search and search using acoustic models that employ a weighted mixture of Gaussian probabilities as density functions. We have implemented these algorithms to function as a standalone decoding engine and evaluated them in isolation using statistical measures and synthetic data. This search engine will later be integrated with other software modules implementing a language model and a speech signal-processing front-end to build a complete speaker independent LVCSR system. The performance of this LVCSR system will be evaluated on speech data available in the public domain and compared with that of other recognizers as a benchmark. The final software will be placed in the public domain at the Institute of Signal and Information Processing (ISIP).

26. INTRODUCTION

Speech is one of the most natural means of exchanging information for humans, and this has spawned a growing interest in developing machines that can accept human speech as input and act appropriately on the information conveyed in it. Enlisting the possible applications of such a system capable of understanding natural human speech is a task limited only by human imagination. The aim of a continuous speech recognizer is, therefore, to provide an efficient and accurate mechanism to transcribe speech into text. To maximize the benefits of such a system and to make it universally applicable, it is desirable that it have the capability to handle a large vocabulary and be independent of speaker characteristics like accents, speaking styles and dysfluencies, as well as different grammatical structures and noise environments.

Even though human communication through speech appears to be extremely easy, replicating the situation artificially has proved to be one of the biggest challenges of current technology. Many of the fundamentals of the speech communication process are still not understood clearly, and the dimensionality and complexity of creating an expert system based on such limited knowledge is staggeringly high. It appears that a statistical approach to speech recognition is the most promising one. A stochastic approach circumvents the need for encoding extraordinary amounts of complex information into building a deterministic system.

27. THE SPEECH RECOGNITION PROBLEM

The statistical framework for the speech recognition problem is as follows. If a sequence of words

$$W = w_1, w_2, \ldots, w_N \tag{37}$$

is spoken, and if A is the acoustic evidence (or the observation) that is provided to the system to identify this sequence; then the recognizer should decide in favor of a word string $\hat{W}$ that maximizes the probability that the word string W was spoken given that the data A was observed.

$$p\langle \hat{W} | A \rangle = \max_{W} p\langle W | A \rangle \tag{38}$$

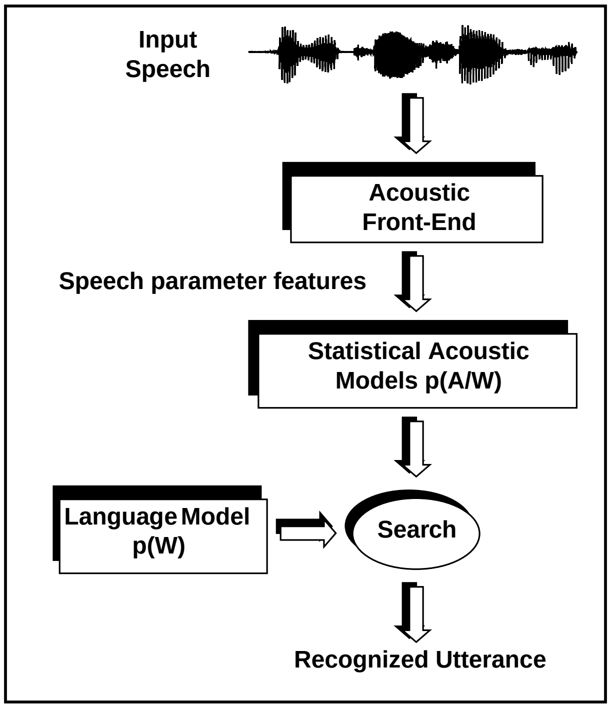

Figure 1: Overview of a Statistical Speech Recognition system

It is difficult to directly compute all the possible *a posteriori* likelihoods $p\langle W|A\rangle$ because of the infinite number of observations possible. This problem can be simplified by applying Bayes formula to finding $\hat{W}$ such that

$$\hat{W} = \underset{W}{\arg max}\, p\langle W\rangle p\langle A|W\rangle \qquad (39)$$

The probability $p\langle A|W\rangle$ that the data A was observed if a word sequence W was spoken is given by a *statistical acoustic model*. The likelihood $p\langle W\rangle$ that enumerates the *a priori* chances of the word sequence W being spoken is determined using a *statistical language model*. Scores for various likely word sequences or hypotheses are generated using the acoustic model scores and the probability of the word sequence given by the language model. The process of combining the two scores and weeding through all the hypotheses to select the one with maximum score is called decoding or search. Figure 1 illustrates the basic schematic structure of a stochastic speech recognition system. We present a brief discussion on each of the main components of this system next.

27.1. Acoustic Front End

A key assumption in stochastic speech processing is that the speech signal is stationary over short intervals of time. The acoustic front end takes in the human speech input and converts it into a signal format. It then divides this signal into small blocks of time (frames) and from each frame derives an estimate of the spectrum of the signal. The spectral variation over different frames is the information used by further stages of the recognizer.

Typically, the signal is divided into 10 ms to 20 ms frames and the frames are overlapped to give a longer analysis window (popular window durations are 25 ms and 30 ms). For better spectral estimates a tapered window function (Hamming) is used. Often, the signal is preemphasized to compensate for high-frequency attenuation caused by lip-radiation.

The spectral features can be extracted using a multitude of techniques. They can be broadly classified as linear prediction based and Fourier analysis based approaches. The popularly used spectral features are signal energy, mel-spaced cepstral coefficients (which can be derived by linear prediction, filter-banks or Fourier analysis) and their temporal derivatives of the first and second order [1, 2, 3, 4, 5, 6, 7]. The output of the front-end is a multidimensional feature vector for each frame of input speech data.

27.2. Statistical Acoustic Models

If a sequence of acoustic feature vectors Y is obtained from the front-end, the acoustic models need to provide a likelihood score for any such Y given a word sequence W. It is impractical to do this calculation for every possible word sequence in case of large vocabulary sizes and hence word sequences are decomposed into basic sound units called *phones*.

The earlier approaches to continuous speech recognition used the technique of Dynamic Time Warping (DTW) [8]. However, it was found to be impractical even for moderately large-vocabulary tasks as it places very high requirements on both memory and computational capacity for implementation. Moreover, it suffers from problems of robustness across multiple speakers, interpolation of parallel hypotheses and inappropriate modeling of word-durations.

A Hidden Markov Model (HMM) is used to model each phone (or in many systems now, a context-based group of phones). An HMM is a doubly stochastic state machine that has a Markov distribution associated with transitions between various states, and some probability density function that models the output for every state. Depending on the complexity of the recognition problem, this distribution can be modeled as a discrete-valued or continuous-valued process [9, 10].

In speech recognition applications the choice of this

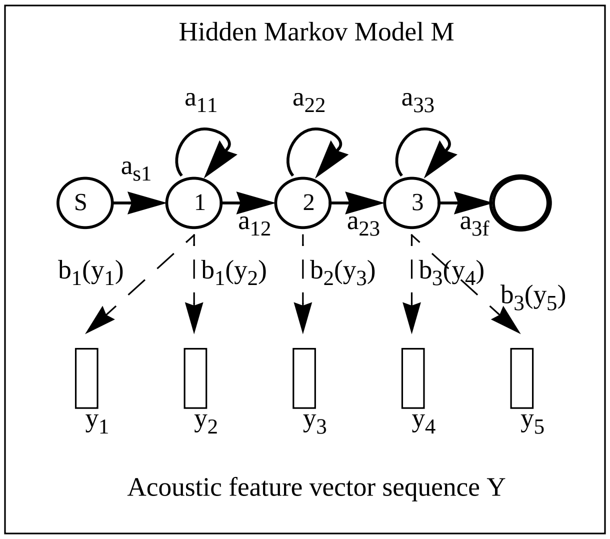

Figure 2: HMM-based acoustic model

output probability function is crucial as it must model all of the intrinsic spectral variability of real speech. Most current state of the art systems use a multivariate Gaussian distribution to model context-dependent sequences of three phones (triphone models).

The prime motivation behind HMM-based acoustic modeling is the availability of algorithms to train them in a fairly efficient manner. Viterbi [11] and Baum-Welch [12] are two commonly used techniques in HMM acoustic model training. The Baum-Welch forward-backward training algorithm recursively re-estimates the HMM parameters using the joint probability computation for each state sequence and the observed output sequence. The Viterbi algorithm does the same by finding a state-sequence that maximizes the output sequence probability.

The HMM can be thought of as a vector sequence generator, where at every time unit instant t the model makes a transition to a new state j and outputs and acoustic speech vector y_t with an output probability $b_j(y_t)$. The transition from state i to state j is also governed by a probability a_{ij}. Figure 2 has an illustration of a simple HMM topology which is commonly used in phone models in current state of the art systems. Here the HMM moves through the state sequence *S, 1, 1, 2, 3, 3, F* to generate the corresponding acoustic vector sequence y_1 through y_5.

We will discuss some aspects of acoustic modeling in detail in the next section.

27.3. Language Models

A language model provides constraints on the occurrence of particular words and word sequences; in particular it provides a mechanism to estimate the probability of some word w_k in a word sequence W given the preceding words $W_1^{k-1} = w_1 \dots w_{k-1}$. A good language model should be able to incorporate grammatical constraints imposed by the structure of the language, topical dependencies and singularities like awkward phrasing, abbreviated word forms etc.

A simple but effective way of language modeling is to treat word sequences as nth order Markov chains.

$$p\langle W\rangle = \prod_{i=1}^{N} p\langle w_i | w_1, w_2, \dots, w_n\rangle \qquad (40)$$

This gives rise to the notion of n-grams [13] where the probability of occurrence of a word depends only on its n predecessors.

$$p\langle w_k | W_1^{k-1}\rangle = p\langle w_k | W_{k-n+1}^{k-1}\rangle \qquad (41)$$

n-grams simultaneously encode syntax, semantics and pragmatics and they concentrate on local dependencies. Also, n-gram probabilities can be directly computed from text data and therefore do not require explicit linguistic rules like a formal language grammar.

Most systems use a trigram back-off language model, though there are a few systems that have ventured as far as four-grams. Apart from such static models there are other techniques like long-range n-grams [14], triggers [15, 16, 17, 18], word caches [19, 20, 21] and class grammars as well as decision-tree clustered grammars [22].

28. ADVANCED ACOUSTIC MODELS

The simple model described in Figure 2 is found to be inadequate for modeling continuous speech as contextual effects cause large variations in the way the same phoneme may sound. a number of modifications have been made to the basic HMM phone model to make it more amenable to continuous speech.

28.1. Context-dependent Models

To achieve good phonetic description between sounds that differ only because of the context in which they appear, different HMMs need to be trained for each of such contexts. A simple and effective way to do this is to use *triphone* models, where every phone has a different HMM corresponding to every unique pair of left and

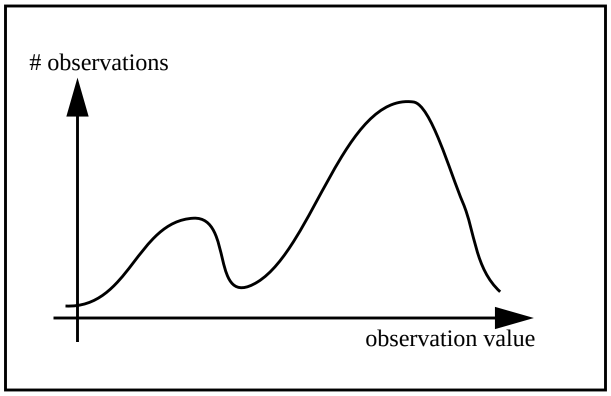

Figure 3: Multiple modalities of a single sound

right neighbors [23].

Triphone models can be trained to encompass word boundaries. Such cross-word triphones offer maximum modeling accuracy but offer a number of complications for the implementation of the decoding strategies. On the other hand, word-internal triphones (with an optional back-off to diphone models) are simplistic from an implementation and training perspective. However, these lack the ability to model contextual effects at word boundaries and hence add to the recognition error.

28.2. Mixture Distributions

The triphone models account for a significant amount of contextual variation of the phones. However, even within the same immediate context there are a number of modalities associated with each sound.These exist due to speaker characteristics like speed, accent and dialect; as well as long-term contexts within and across words. For instance, the phone “t” has 32 different modalities of pronunciation. A lot of these are taken care of by creating different triphones for “t”, but even a single such triphone may have a number of distinct modalities. Figure 3 shows a simple illustration of two modalities for a single phone.

A single Gaussian distribution cannot model such a state output. Therefore, a linearly weighted sum of different Gaussian densities is used to model the form of the state output probability distribution.

$$f_{Y/X}\langle\xi|i\rangle = \sum_{m=1}^{M} c_{im} \aleph(\xi, \mu_{im}, C_{im}) \qquad (42)$$

where

$$\sum_{m=1}^{M} c_{im} = 1 \qquad ,1 \le i \le S \qquad (43)$$

and $f_{Y/X}\langle\xi|i\rangle$ models the output density distribution $b_i(y_t)$.

Detailed reestimation equations for the mixture parameter mean and covariances as well as the weights have been derived. Suppose that we define

$$v(i;t,m) = p\langle \underline{x}(t) = i | \underline{y}(t) \rangle \qquad (44)$$

such that $y(t)$ was produced according to the mixture component m. Then

$$v(i;t,m) = \frac{\alpha(y_1^t, i)\beta\langle y_1^T | i\rangle}{\sum_{j=1}^{S} \alpha(y_1^t, j)\beta\langle y_1^T | j\rangle} \times \frac{c_{im} \aleph(\xi, \mu_{im}, C_{im})}{\sum_{l=1}^{M} c_{il} \aleph(\xi, \mu_{il}, C_{il})}$$

(45)

where $\alpha(y_1^t, i)$ and $\beta\langle y_1^T | i\rangle$ are the Baum-Welch probability terms for the partial forward and backward sequences for time *t*. Also define the sum of all such terms in Equation 8 as

$$v(i;*,m) = \sum_{t=1}^{T} v(i;t,m) \qquad (46)$$

so that the reestimation equations can be written as

$$\bar{c}_{im} = \frac{v(i;*,m)}{\sum_{m=1}^{M} v(i;*,m)} \qquad (47)$$

$$\bar{\mu}_{im} = \frac{\sum_{t=1}^{T} v(i;t,m) y(t)}{v(i;*,m)} \qquad (48)$$

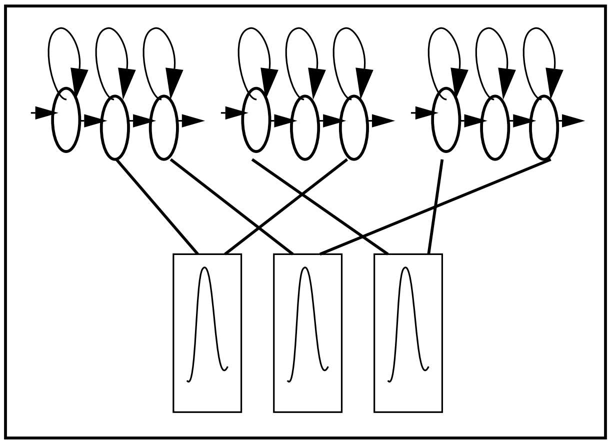

Figure 4: State-tying across different triphone models

$$\bar{C}_{im} = \frac{\sum_{t=1}^{T} v(i;t,m)[y(t)-\mu_{im}][y(t)-\mu_{im}]^T}{v(i;*,m)} \tag{49}$$

Equation (11) represents a ratio of the expected number of times the system is in state *i* and uses the *m*th mixture component to generate the observation vector, to the number of times the system is in state *i*. Equation (12) is a weighted temporal average of the observation vectors and the covariance in Equation (13) is also a similar weighted computation.

The use of Gaussian mixture densities also puts severe requirements on the computational complexity and memory requirements of the system. Instead of one mean vector and covariance matrix per state now there are an M number of such parameters. Since typically there are about 60,000 triphones to model and approximately a 40-dimensional observation vector, the number of system parameters to estimate easily runs into a million or so. A truly huge amount of training data is required to estimate all these parameters.

28.3. State and Mixture Tying

To alleviate the problem of too many parameters and too little training data, it is customary to allow states in different models that display similar characteristics to share the same output distribution. This is called state-tying [24, 25, 26, 27]. Similarly if two HMM states share some common modalities, the mixture components corresponding to those modalities may also be tied together. This leads to a tied-mixture system [28, 29]. Since good smoothing techniques have been developed for continuous density distributions, current systems use typically 8 to 10 mixtures per state and then tie them up.

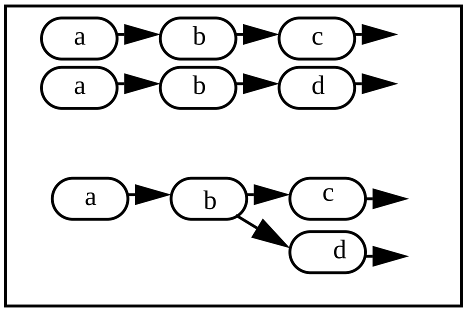

Figure 5: Optimized hypothesis generation

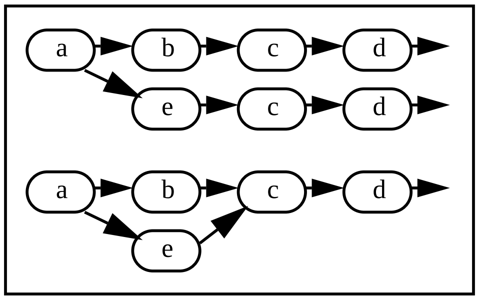

Figure 6: Problem space reduction by merging

The choice of which states to tie can be made using a phonetic decision tree [30, 31]. A simple illustration of state-tying is displayed in Figure 4.

29. SEARCH TECHNIQUES

A decoding strategy is required to find the most likely word sequence given the acoustic models and the language model constraints, and the spoken utterance (or acoustic data). It combines the scores obtained on the acoustic and language models and generates the possible word sequences (or hypotheses) from which we need to find the most likely for recognition.

An intuitive and straightforward solution appears to be to simply combine the two scores and generate all possible hypotheses and select the one with the largest score. However, even for a small vocabulary task the enumerative search will fail to terminate in a practical amount of time, since the number of likely word sequences rises exponentially with the length of the sequence.

Therefore we need to restructure our decoding problem to restrict the search space in some meaningful fashion. Some popular techniques for restructuring the search space [32] are as follows:

• *Optimized Hypothesis Generation* involves merging common partial hypotheses. A set of all partial sequences (hypotheses) is constructed in the form of a tree where common portions of the hypotheses are tied together. Refer to Figure 5 for an illustration.

• *Problem-space Reduction* involves transforming the state space of the problem to make the search more efficient. This is done by merging common states in different hypotheses so that they need not be evaluated again and again. Figure 6 has an example.

• *Search Reduction* entails pruning away hypotheses that have partial evaluation scores less than some complete evaluation or some pre-determined threshold.

• *Knowledge Application* makes use of expert information to improve the efficiency of the search. A tight constraint in the knowledge base translates directly into smaller search space.

Use of such approximations forces the decoder to make sub-optimal choices, but it has been observed that this does not significantly affect the recognition error rate. Based on these modifications a number of search techniques have been evolved.

29.1. Viterbi Search Techniques

Viterbi search and its variants form what is known as the breadth-first search techniques. Here all hypotheses are pursued in parallel and gradually pruned away as the correct hypothesis emerges with the maximum score.

The recognition system can be treated as a recursive transition network composed of the states of HMMs in which any state can be reached from any other. The Viterbi search algorithm [11] builds a breadth-first search tree out of this network in the following fashion:

```
• If N is the duration of the utterance, N number of state lists S are generated. These lists are initialized by setting the probability of the initial state as 1 and the others 0.

• For each state s in S(t)

    For each possible transition from s to some state s' in S(t+1)

    - Compute the transition score p(s'/s)

    - If s' is uninitialized, initialize it with a score p(s'/s) and a backpointer to s.

    - Else update score of s' only if this transition gives a better score.

• Go to step 2 with t = t + 1

• If t = N backtrack to get best path
```

Viterbi search is time-synchronous; i.e. at any stage all partial hypotheses correspond to the same portion of the utterance and hence can be directly compared. However, a complete Viterbi search is impractical for even moderate-sized tasks because of the size of the state space. A Viterbi *beam* search is used to reduce the search space.

In *Viterbi beam search* [33, 34, 35, 36] only the hypotheses whose likelihood falls within a fixed radius of the most likely hypothesis are considered. It is a dynamic programming technique that exploits the observation that many states in the state lists have zero or near-zero scores and therefore need not be considered towards a solution. The best beam size can be determined empirically or adaptively. The advantage of the dynamic beam heuristics is that it allows the search to consider many good hypotheses in absence of a clearly dominant solution. Conversely, in case of a clear best hypothesis few others need to be maintained. The main problem with this strategy is that the same state occurring in different paths needs to be recomputed every time adding to the computation cost.

Many variations of Viterbi beam search have been proposed to improve upon its performance. The state space can be partitioned into subsets that are subject to different beam widths [37]. If there is more information in the form of a larger number of contextual states a tighter pruning threshold is applied. A maximum of path scores may be taken when they merge at word boundaries and a sum when the merging is within a word. In another modification, additional pruning is performed at the frame level to evaluate only a few best-scoring states [38]. This pruning is typically done only at the few initial frames as almost 95% of hypotheses are generated here. In very large vocabulary problems, a tree structured network in which the states corresponding to common initial phones are shared by different words can be used [39]. This uses the fact that the uncertainty about the identity of the word is much higher at its beginning than at the end and therefore more computation is required at the initial phones than the later ones.

29.2. Stack Decoders

Stack decoding search [40] is similar to the A* search in artificial intelligence [41]. It is a depth-first technique in which the most promising hypothesis is pursued until the end of the speech data is reached. It constructs a search tree from the language model state graph where the states correspond to abstract states in the language and the branches represent transitions between these states. The basic stack decoder paradigm [42, 43] can be summarized as:

• Pop the best partial hypothesis from the stack

• Apply acoustic and language model fast matches (computationally cheap methods for reducing the number of word extensions which need to be checked by the more accurate but computationally expensive detailed matches) to shortlist the candidate next word.

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

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

• Insert surviving new hypotheses into the stack.

The stack decoding algorithm requires an evaluation function to compare hypotheses of different lengths. The evaluation uses only the forward algorithm to evaluate partial scores, and is therefore unsuitable as it causes the search to always prefer shorter hypotheses. This is avoided by making the evaluation function normalizing and discriminating, so that it compensates for the path length and favors the optimal path more and more with time.

The A* stack decoder suffers from problems of speed, size, accuracy and robustness. However, an important advantage of the stack decoder is its consistency with the forward-backward algorithm. Therefore several variations that use weaker and cheaper initial acoustic and language models to produce a list of likely hypotheses that is later refined using more detailed and expensive models have been proposed that improve on its performance. An important emerging stack decoding technique is the envelope search.

30. MULTIPASS SEARCH

This class of search strategies have found widespread use in modern-day LVCSR systems. An approximate and efficient search is used to generate a subset of hypotheses that are more likely than others, and in subsequent passes of more detailed decoding over this reduced search space the correct hypothesis is found. The advent of N-best search has been instrumental to the advancement of multipass search techniques.

30.1. N-best Search

The optimal N-best decoding algorithm [44] is quite similar to the Viterbi search. However, while Viterbi decoding is inherently 1-best, N-best search finds all hypothesis sequences within the specified beam and keeps track of hypotheses with different histories at each state. It then allows only N top-scoring hypotheses to propagate to the next state. This state-dependent pruning is independent of the global Viterbi beam threshold.

The sources of information on speech used for recognition purposes can be extremely diverse and are correspondingly associated with different costs in terms of computation and memory requirements. A hypothesis that scores the highest given all these knowledge sources will be an optimal solution to the recognition problem. But this typically requires an impractically large search space. It is advantageous to use a strategy in which the most efficient knowledge sources are used first to generate a list of top N hypotheses. These hypotheses can later be re-evaluated with other, more expensive knowledge sources to arrive at the best hypothesis. N-best search provides an efficient method of integrating different knowledge sources and makes the search process more modular. The scores from different knowledge sources can be combined using weights chosen to minimize the recognition error [45].

The N-best paradigm as described above has the problem of being partial towards shorter hypotheses. In other words, if we consider the probability of error in recognition of a single word being roughly independent of its position in the sentence, then a longer sentence will have more errors and therefore will be pushed down in the rank of correct hypotheses. Thus an exact N-best search will require a very large value of N to find the correct answer for a long sentence.

A number of modifications have been proposed to overcome this problem and to make N-best search more accurate and efficient. These modifications allow for some approximations to generate the list of sentences with much less computation. Such approximations are justified as long as the correct hypothesis is assured to be in this list. Even if it does not hold a very high rank in this preliminary list, the correct hypothesis can be found later by rescoring on other knowledge sources.

30.2. Lattice N-best Search

An initial pass of the recognition system is used to build a lattice of word (or phoneme or syllable etc.) hypotheses which is searched through by subsequent passes to generate the correct hypothesis. A time-synchronous one-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 at this frame is sent forward along with a backpointer to the saved list [46]. The N-best sentences are obtained by recursive search through this traceback list. This algorithm is extremely fast but often underestimates or misses high-scoring hypotheses.

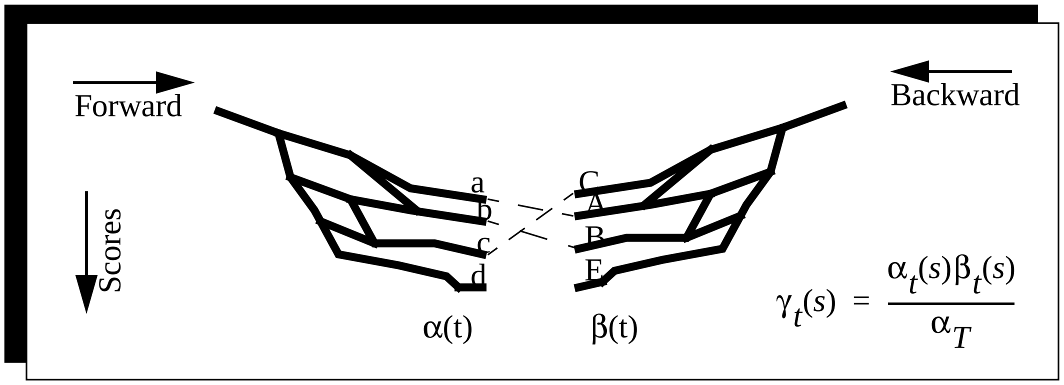

Figure 7: Forward-backward search: The combined score is the normalized product of the forward and backward path scores

A progressive search [47] can be used to avoid this problem. Here a lattice of all sentence hypotheses is maintained instead of evaluating independent sentence hypotheses. This lattice is treated as a grammar and used to rescore all the hypotheses.

30.3. Word-dependent N-best Search

This algorithm differentiates between hypotheses on the basis of only the previous word instead of the whole preceding sequence [46]. The probability for each of the different preceding words is stored within the word at each state. At the end of the word the score for each hypothesis and the name of the previous word are recorded. A recursive traceback is used at the end of the sentence to derive the list of the most likely sentences.

30.4. Forward-Backward Search

Forward-backward search algorithms use an approximate time-synchronous search in the forward direction to facilitate a more complex and expensive search in the backward direction [46, 48, 49, 50]. This generally results in speeding up the search process on the backward pass as the number of hypotheses to be explored is greatly reduced by the forward search.

A simplified acoustic or language model is used to perform a fast and efficient forward-pass search in which the scores of all partial hypotheses that fall above a pruning beamwidth are stored at every state. Then a normal within-word beam search is performed in the backward direction 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 7 describes the forward-backward search in detail.

Similar to the Baum-Welch training algorithm we combine the scores on the forward and backward passes to compute the overall score at each state s of the HMM at time t. Thus

$$\gamma_t(s) = \frac{\alpha_t(s)\beta_t(s)}{\alpha_T} \tag{50}$$

where α and β are the partial path scores on the forward and backward search passes. The N-best sentences thus obtained are rescored using more sophisticated acoustic and language models to obtain the best sentence hypothesis.

Since the forward-backward search allows use of different models on the two passes, a complex model can be used on the backward pass to come up with extremely accurate results [51]. The forward scores, though not exact, are good enough estimates of the word end scores and can be further modified by normalization relative to the highest score in each frame. The time-synchronous nature of both passes allows them to have different normalized scores without loss of accuracy.

Forward-backward search algorithms have greatly facilitated real-time handling of large-scale tasks. The backward pass search is fast enough to be performed without any perceptible delay after the forward search. The forward search can be made more approximate and hence efficient as the scores need not be very accurate on the forward pass.

A variation of the forward-backward N-best search is a *tree-trellis based fast search* algorithm [52] that uses a modified Viterbi beam algorithm in the forward pass and an A* stack decoder search on the backward pass. The partial hypothesis map prepared in the forward trellis search is used by the backward search to estimate the incomplete portion of the partial hypothesis.

31. SYSTEM IMPLEMENTATION

In the course of this project we tried to implement a

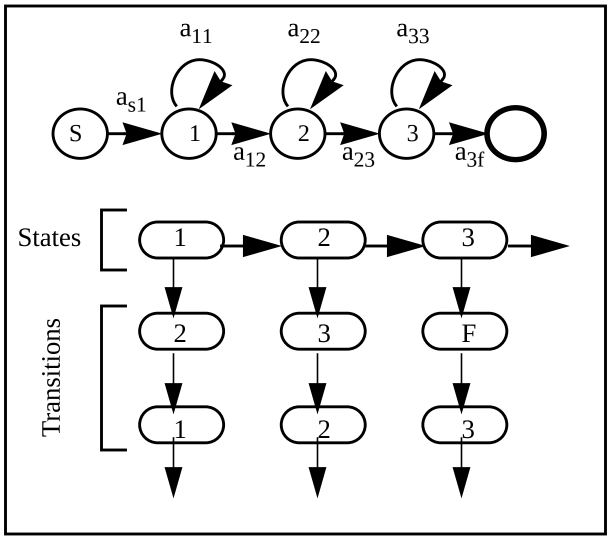

Figure 8: Linked list structure of an HMM

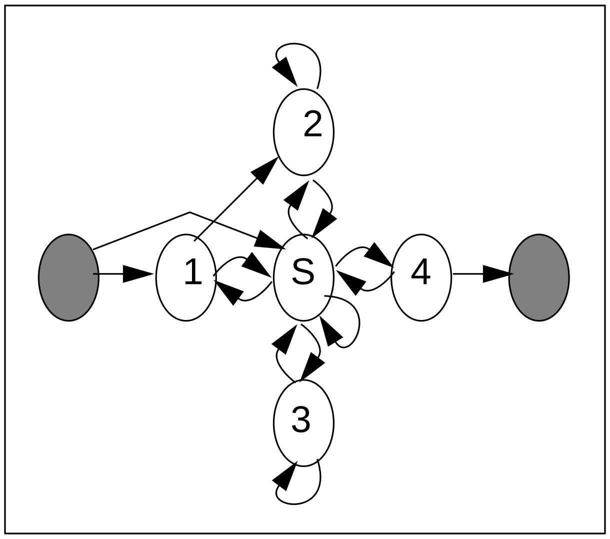

Figure 9: The language model for evaluation

HMM-based speaker independent LVCSR system. A key component of this system was the search module that used acoustic models built with Gaussian mixture distributions and employed a multipass N-best strategy. We will now concentrate on describing the implementation and evaluation of this search engine.

The search algorithms were implemented with an object-oriented thrust. we designed the module to function as a standalone unit that can be plugged into any system that employed HMM-based technology. The next step is to combine this module with an acoustic front end and a language model to build a public domain HMM-based LVCSR system.

31.1. Structure of the Software

Present-day research in LVCSR technology is severely constrained by the limitations on the amount of easily accessible portable memory. This makes it imperative to use well-designed data structures that optimize memory usage and allow for its reusability.

In order to provide a lot of flexibility to the user to set the topology of the models and the component structure of mixture distributions most of the implementation is based on linked lists. Figure 8 shows the implementation of an HMM as a linked list.

The hypotheses are kept track of through another linked list of a scoring structure associated with each HMM state. Each hypothesis is stored using back-pointers to the previous time-frame score structure. Score structures which have not grown for the last two frames are deleted from the list as dead hypotheses (this is the pruning action). The final hypotheses are found by tracing through these back-pointers starting at the head of the linked list at the time of end of data.

31.2. Data Driven Implementation

The software is capable of adapting to a variety of HMM-based recognition applications. There are no constraints on the topology of the HMMs, as well as the number of mixture components at any state of any model. The user can define the number and structure of the models, as well as the grammar that governs the language model.

32. EXPERIMENTS AND RESULTS

We evaluated the performance of the search algorithm in isolation on synthetically created data for a simple experimental setup that closely simulates the working of a real recognition system. We created a grammar structure for a digit-string recognition application and some dummy acoustic word models for the digits.

32.1. Experimental Setup

The experiment for evaluation of the search algorithm consisted of a digit-string recognition problem for the digits “one” through “four” and silence.

The acoustic models were given a fixed topology as that displayed in Figure 8. The state transitions were set to be equiprobable. Tri-dimensional Gaussian mixtures were used to model the output distributions of each HMM state. Different states had different number of mixture components (we experimented with at most three mixtures per state). The mean vectors were chosen to allow a reasonable amount of confusibility between

Reference Sentence

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

Recognized Sentence Hypotheses

Score: 111.87 <Silence> <Silence> one two four <Silence> three <Silence>
Score: 111.87 <Silence> <Silence> one two four <Silence> three <Silence> <Silence>
Score: 115.36 <Silence> <Silence> one two four <Silence> three <Silence>

Figure 10: Sample results for the evaluation test

certain words. A duration constraint was imposed to allow a maximum stay of 9 frames in any single HMM. A Euclidean distance measure was used to compute the likelihood scores on each acoustic model.

The grammar allowed for multiple transitions between words as well as self-loops (thus allowing for digit strings of arbitrary length). The grammar transition probabilities were chosen arbitrarily [Figure 9].

The test data was synthetically generated by passing a test sequence of words and durations to a data generator program (that we developed for this testing) and getting the corresponding sequence of vectors as the output with user-defined amount of White Gaussian noise added to the feature vectors.

32.2. Results

We ran a number of tests constraining the grammar so that any utterance of the digit string would start only with a silence, and observed that the best sequences output were typically variants of each other in one or two places at most, or differed in durations and locations of silences.

A sample of the system output for the aforementioned setup is shown in Figure 10. Further experiments are currently underway for more detailed analysis of the algorithms.

33. SUMMARY

Statistical speaker-independent large vocabulary continuous speech recognition systems have enjoyed significant progress with the advent of high-power efficient algorithms for search and detailed acoustic modeling techniques. Current state of the art machines are capable of real-time recognition of vocabulary sizes ranging more than 40,000 words. The algorithms implemented in this work represent the leading technology to date, with systems like BBN's Byblos and Cambridge University's HTK system using similar techniques.

Data driven systems allow the user flexibility to adapt the application to a wide variety of problem environments. We have a commitment to produce modular object-oriented systems that will find a multitude of applications in the public domain. This software was created as part of a larger project dedicated to build a flexible freeware LVCSR system. Our future efforts will be dedicated in this regard.

The focus of our research will now be to complete the implementation of the backward pass and integrate the search modules with the other components to complete the recognition system. We will train and test the system on real speech data and compare its performance with other LVCSR systems on similar tasks.

34. ACKNOWLEDGEMENTS

We would like to express our heartfelt thanks to Dr. Joseph Picone of the Institute for Signal and Information Processing (ISIP), Mississippi State University for his guidance and support. His insightful comments on the LVCSR technology and patience in dealing with our (mostly) naive problems have been a tremendous contribution to this project. We are also grateful to Aravind Ganapathiraju of ISIP for many interesting and stimulating discussions and support.

REFERENCES

95. J. Picone, "Signal Modeling Techniques in Speech Recognition", in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, Vol. 81, No. 9, pp. 1215-1247, 1993.

96. S.B. Davis and P. Mermelstein, "Comparison of

Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences”, in *IEEE Transactions ASSP*, Vol. 28, No. 4, pp. 357-366, 1980.

97. G.F. Chollet and C. Gagnoulet, “On the Evaluation of Speech Recognizers and Data Bases Using a Reference System”, in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 2026-2029, Paris, 1982.
98. D.B. Paul, “Algorithms for an Optimal A* Search and Linearizing the Search in the Stack Decoder”, in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 693-696, Toronto, 1991.
99. B.S. Atal, “Effectiveness of Linear Prediction Characteristics of the Speech Wave for automatic Speaker Identification and Verification”, in *the Journal of the Acoustical Society of America*, Vol. 55, No. 6, pp. 1304-1312, 1974.
100. S. Furui, “Cepstral Analysis Technique for Automatic Speaker Verification”, in *IEEE Transactions ASSP*, Vol. 29, No. 2, pp. 254-272, 1981.
101. H. Hermansky, “Perceptual Linear Predictive (PLP) Analysis of Speech”, in *the Journal of the Acoustical Society of America*, Vol. 87, No. 4, 1990.
102. J.R. Deller, J.G. Proakis and J.H.L. Hansen, *Discrete-Time Processing of Speech Signals*, Macmillan Publishing, New York, 1993.
103. L.R. Rabiner, “A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition”, *in Proceedings of IEEE*, Vol. 77, No. 2, pp. 257-285, 1989.
104. J.W. Picone, “Continuous Speech Recognition Using Hidden Markov Models”, in IEEE ASSP, pp. 26-41, 1990.
105. A.J. Viterbi, “Error Bounds for Convolutional Codes and an Asymptotically Optimal Decoding Algorithm”, *in IEEE Transactions on Information Theory*, Vol. IT-13, pp. 260-269, April 1967.
106. L.E. Baum, “An Inequality and Associated Maximization Technique in Statistical Estimation for Probabilistic Functions of Markov Processes”, in *Inequalities*, Vol. 1, pp. 1-8, 1972.
107. F. Jelinek, “Up From Trigrams!”, Eurospeech 1991.
108. X.D. Huang, F. Alleva, H.W. Hon, M.Y. Hwang, K.F. Lee and R. Rosenfeld, “The SPHINX-II Speech Recognition System: An Overview”, in *Computer, Speech and Language*, 1992.
109. R. Rosenfeld and X.D. Huang, “Improvements in Stochastic Language Modeling”, in *Proceedings DARPA Speech and Natural Language Workshop*, February 1992.
110. R. Rosenfeld, “A Hybrid Approach to Adaptive Statistical Language Modeling”, in *Proceedings of DARPA Human Language Technology Workshop*, pp. 76-81, March 1994.
111. R. Rosenfeld, “Adaptive Statistical Language Modeling: A Maximum Entropy Approach”, Ph.D. Thesis Proposal, Carnegie Mellon University, September 1992..
112. R. Lau, R. Rosenfeld and S. Roukos, “Trigger-Based Language Models: A Maximum Entropy Approach”, in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, Vol. 2, pp. 45-48, April 1993.
113. J. Kupiec, “Probabilistic Models of Short and Long Distance Word Dependencies in Running Text”, in *Proceedings ARPA Workshop on Speech and Natural Language*, pp. 290-295, February 1989.
114. F. Jelinek, B. Merialdo, S. Roukos and M. Strauss, “A Dynamic LM for Speech Recognition”, in *Proceedings ARPA workshop on Speech and Natural Language*, pp. 293-295, 1991.
115. R. Kuhn and R. de Mori, “A Cache Based Natural Language Model for Speech Recognition”, in *IEEE Transactions on PAMI*, Vol. 14, pp. 570-583, 1992.
116. L. Bahl, P.F. Brown, P.V. de Souza and R.L. Mercer, ``A Tree-Based Statistical Language Model for Natural Language Speech Recognition”, in *IEEE Transactions ASSP*, Vol. 37, No. 7, pp. 1001-1008, 1989.
117. S. Young, “Large Vocabulary Continuous Speech Recognition: a Review”, to appear in *IEEE Transactions ASSP*, 1996.
118. X.D. Huang, H.W. Hon, M.Y. Hwang and K.F. Lee, “A Comparative Study of Discrete, Semi-Continuous and Continuous Hidden Markov Models”, in *IEEE Computer Speech and Language*, Vol. 7, No. 4, pp. 359-368, 1993.
119. S.J. Young, “The General Use of Tying in Phoneme-based HMM Speech Recognizers”, in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, Vol. 1, pp. 569-572, San Francisco, 1992.
120. M.Y. Hwang and X. Huang, “Shared Distribution Hidden Markov Models for Speech Recognition”, in *IEEE Transactions on Speech and Audio Processing*, Vol. 1, No. 4, pp. 414-420, 1993.
121. S.J. Young and P.C. Woodland, “State Clustering in HMM-based Continuous Speech Recognition”, in *IEEE Computer Speech and Language*, Vol. 8, No. 4, pp. 369-384, 1993.

122.V. Digalakis, P. Monaco and H. Murveit, “Generalized Mixture Tying in Continuous Speech HMM-based Recognizers”, in *IEEE Transactions on Speech and Audio Processing*, 1995.

123.V. Digalakis and H. Murveit, “High Accuracy Large Vocabulary Speech Recognition using Mixture Tying and Consistency Modeling”, in *Proceedings of Human Language Technology Workshop*, pp. 313-318, Morgan Kaufman Publishers Inc., 1994.

124.L.R. Bahl, P.V. de Souza, P.S. Gopalakrishnan, D. Nahamoo and M.A Pichney, “Context Dependent Modeling of Phones in Continuous Speech Using Decision Trees”, in *Proceedings of DARPA Speech and Natural Language Processing Workshop*, pp. 264-270, 1991.

125.S. Browning, M. Russell and S. Downey, “Phoneme Decision Tree Construction for Automatic Speech Recognition”, DRA Memorandum No. 4666, Defence Research Agency, 1993.

126.K.F. Lee and F. Alleva, “Continuous Speech Recognition”, *Advances in Speech Signal Processing*, edited by S. Furui and M. M. Sondhi, pp. 651-699, Marcel Dakker Inc., 1992.

127.B. T. Lowerre, “The HARPY Speech Recognition System”, Ph.D. Thesis, Carnegie Mellon University, 1976.

128.Y.L. Chow, M. Ostendorf-Dunham, O.A. Kimball, M.A. Krasner, G.F. Kubala, J. Makhoul, S. Roukos and R.M. Schwartz, “BYBLOS: The BBN Continuous Speech Recognition System”, in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 89-92, April 1987.

129.K.F. Lee and H.W. Hon, “Large-Vocabulary Speaker-Independent Continuous Speech Recognition”, in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 123-126, April 1987.

130.H. Ney, D. Mergel, A. Noll and A. Paeseler, “A Data-Driven Organization of the Dynamic Programming Beam Search for Continuous Speech Recognition”, in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 833-836, April 1987.

131.F. Alleva, H. Hon, X. Huang, M. Hwang, R. Rosenfeld and R. Weide, “Applying SPHINX-II to the DARPA Wall Street Journal CSR Task”, in *Proceedings of DARPA Speech and Natural Language Processing Workshop*, pp. 393-398, 1992.

132.N. Deshmukh, J. Picone and Y.H. Kao, “Efficient Search Strategies in Hierarchical Pattern Recognition Systems”, in *Proceedings of 27th IEEE Southeastern Symposium on System Theory*, pp. 88-91, 1995.

133.J.J. Odell, V. Valtchev, P.C. Woodland and S.J. Young, “A One Pass Decoder Design for Large Vocabulary Recognition”, in *Proceedings of DARPA Speech and Natural Language Processing Workshop*, pp. 380-385, 1992.

134.R.L. Bahl, et al., “Large Vocabulary Natural Language Continuous Speech Recognition”, in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 465-467, May 1989.

135.N.J. Nilsson, *Problem Solving Methods in Artificial Intelligence*, McGraw-Hill, New York, 1971.

136.D.B. Paul, “An Efficient A* Stack Decoder Algorithm for Continuous Speech Recognition with a Stochastic Language Model”, in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 405-409, March 1992.

137.D.B. Paul, “The Lincoln Large-Vocabulary Stack Decoder Based HMM CSR”, in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 374-379, April 1993.

138.Y.L. Chow and R.M. Schwartz, “The N-Best Algorithm: An Efficient Procedure for Finding Top N Sentence Hypotheses”, in *Proceedings of DARPA Speech and Natural Language Processing Workshop*, pp. 199-202, October 1989.

139.A. Kannan, M. Ostendorf and J.R. Rohlicek, “Weight Estimation in N-Best Rescoring”, in *Proceedings of DARPA Speech and Natural Language Processing Workshop*, pp. 455-456, February 1992.

140.R.M. Schwartz and S. Austin, “Efficient, High-Performance Algorithms for N-Best Search, in *Proceedings of DARPA Speech and Natural Language Processing Workshop*, pp. 6-11, June 1990.

141.H. Murveit, J. Butzberger, V. Digalakis and M. Weintraub, “Progressive-Search Algorithms for Large-Vocabulary Speech Recognition”, in *Proceedings of DARPA Speech and Natural Language Processing Workshop*, March 1993.

142.L. Nguyen, R. Schwartz, F. Kubala and P. Placeway, “Search Algorithms for Software-Only Real-Time Recognition with Very Large Vocabularies”, in *Proceedings of DARPA Speech and Natural Language Processing Workshop*, pp. 91-95, March 1993.

143.L. Nguyen, R. Schwartz, Y. Zhao and G. Zavaliagkos, “Is N-Best Dead?”, in *Proceedings of DARPA*

Speech and Natural Language Processing Workshop, pp. 386-388, March 1994.

144.J.K. Chen and F.K. Soong, “An N-Best Candidates-Based Discriminative Training for Speech Recognition Applications”, in *IEEE Transactions on Speech and Audio Processing*, Vol. 2, No. 1, Part II, pp. 206-216, January 1994.

145.R. Schwartz and S. Austin, “A Comparison of Several Approximate Algorithms for Finding Multiple (N-Best) Sentence Hypotheses”, in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 701-704, 1991.

146. F.K. Soong and E.F. Huang, “A Tree-Trellis Based Fast Search for Finding the N Best Sentence Hypotheses in Continuous Speech Recognition”, in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 705-708, 1991.

147.The WWW Resources

148.The com.speech FAQ