SEARCH STRATEGIES IN SPEECH RECOGNITION

Swapna Gogineni

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

ABSTRACT

This paper is an overview of various search strategies in speech recognition systems. Search problem or decoding plays a crucial role in correct recognition. Search can be defined as, the estimation of the most likely hypothesis for a sequence of words, given the speech signal, acoustic models and the language models. The complexity of the search increases due to the large vocabulary size, imposing constrains on the computation and storage capability of the system. This paper presents various Search Techniques evolved for efficient search, reducing the complexity of the systems and increasing the performance. A brief introduction on the acoustic and language models have also been outlined.

1. INTRODUCTION

The speech recognition task can be referred to as efficiently transcribing speech into text. Since obtaining exact results are almost impossible, a statistical approach is used to find the most likely word sequence. Development of a good speech recognition system enhances the rate of communication with machines, influencing the importance of speech recognition. The selection of an efficient algorithm is very crucial to reduce the complexity and the computations required. It is also important to look for an algorithm that performs with betterefficiency and also in real-time. A number of algorithms have been proposed to perform search efficiently, a few will be discussed in this paper.

Let us assume a word sequence W = w_1, w_2,...., w_n has caused the acoustic data A, where all the words in the word sequence W belong to a known vocabulary V. Then the word sequence $\hat{W}$ corresponding to the highest probability that the word sequence W was spoken and the acoustic data A was observed.

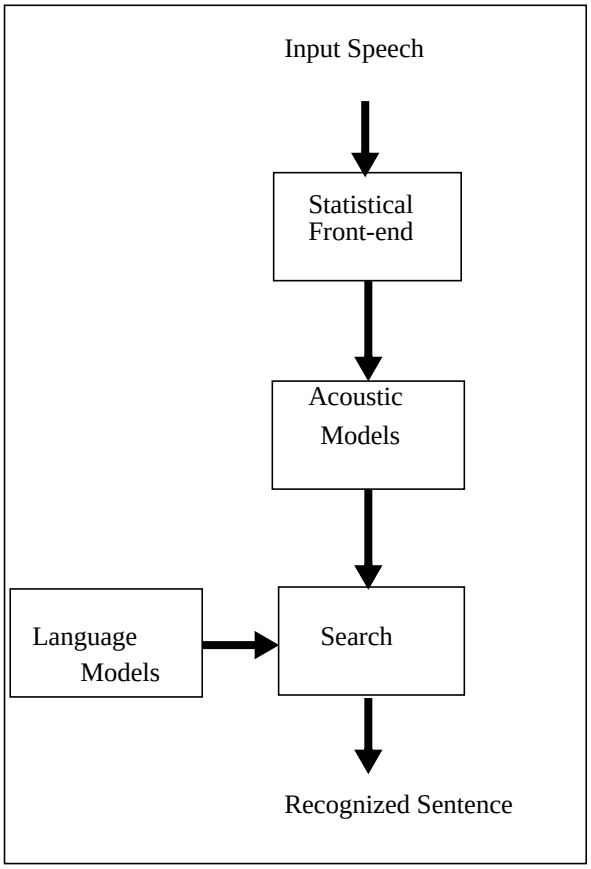

Figure 1: Speech recognition system overview

$$\hat{W} = \arg max P(w/A) \quad (1)$$

where P(W/A) is the probability that the word sequence W was spoken given that the acoustic A is observed. By applying Bayes's formula we can re-write equation (1) as

$$\hat{W} = \arg max P(A/W)P(W) \quad (2)$$

The search task is to evaluate (2) for the likely word sequences and selecting the word sequence with the maximum likelihood.

2. ACOUSTIC MODELS

In speech processing, the acoustic front-end converts the speech signal to a sequence of feature vectors referred to as signal modeling which are used for recognition. Briefly describing, 10 msec frames of data are overlapped to obtain an analysis window of duration 25 msec from which 12 Mel-frequency cepstral coefficients (MFCCs) and power coefficients along with their time derivatives are used to generate the acoustic feature vector.

After obtaining the acoustic feature vectors from the front-end, the acoustic models provide the method to calculate the likelihood of the vector for a given word sequence. If the vocabulary system is small the above mentioned method is feasible. On the other hand, as the vocabulary size increases it is highly impossible to find the likelihood score for all the possible word sequences. For this reason, words are represented as sub-word units like phones, phonemes, triphones, etc. Earlier, 'Dynamic Time Warping' was used to solve this problem. Present day systems use Hidden Markov Models (HMM) to model the sub-word units. Neural Networks is an alternative approach to model the sub-word units.

The HMMs are a set of states connected by transitions based on the Markovian assumption that only the last state is relevant in determining its future behavior. Figure 2. shows a 5 state HMM topology with different transition probabilities p_{ij}. Each state is associated with an output density function from which the acoustic vectors are obtained.

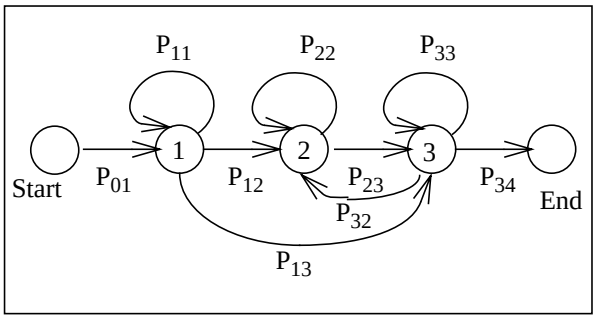

Figure 2: Hidden Markov Models

3. LANGUAGE MODELS

Language models provide us with the probability of a word in a given word sequence. The absence of definite word boundaries increases the importance of language models in large vocabulary systems. A given word sequence can be hypothesized in a number of different ways.

Let us consider a word sequence is a neat person which can be hypothesized as is an eat person or is an eat per son. To overcome all such problems language models have been introduced. They impose grammatical constraints[1] on the word sequence.

The language models provide us with a prior information of the word sequence

$$P(W) = \prod_{i=1}^{n} P(w_i/(w_1, \ldots, w_{i-1})) \quad (3)$$

where $w_1, w_2, \ldots, w_{i-1}$ is the history of the word wi. In reality, it would require large memory to store long histories, so unigrams, bigrams, trigrams, n-grams, etc. have been introduced. Most systems use a trigram language model.

4. SEARCH ALGORITHMS

Having talked about the importance of search we would like to introduce the various search techniques used in speech recognition. The challenges of a search technique is to produce accurate hypothesis with high performance and flexibility. We also need to reduce the search space size and memory usage. As the number of possible hypotheses increases exponentially with the length of the word sequence we follow different approaches for the system to operate in real time. Common techniques used are merging common hypothesis, pruning away unlikely hypothesis and applying external knowledge sources. A few popular search techniques are described briefly here:

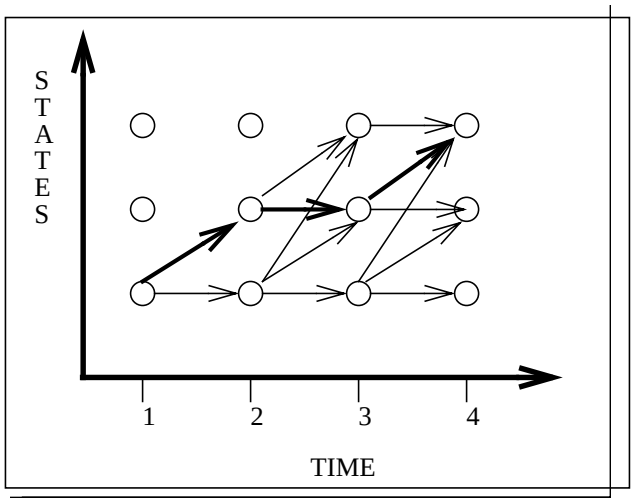

Figure 3: Viterbi Search

4.1. Viterbi Search

Viterbi search is an efficient and most widely used algorithm to find the optimal solution[2]. It is a breath-first search technique where, all hypothesis are computed and pruned away gradually emerging with the maximum score. In the Viterbi search algorithm, the speech signal is divided into frames which are represented by states of HMMs[1]. The transition probabilities of all the possible transitions from state s to s', p(s'/s) are calculated. The one with the highest probability is kept and the remaining are pruned off. This process is repeated incrementing time, till the end of the frame is reached. Once the end of the frame is reached the best path is traced back with the help of back-pointers[3].

Figure 3, shows the best path according to the Viterbi search algorithm. The arrows show all the possible transitions and the bold line shows the best path. The Viterbi search is a time-synchronous algorithm, i.e. it processes all states at time t and then goes to time t+1. The Viterbi algorithm can find the most likely sequence in N^2T computations, where N is the total number of states and T is the total duration[4]. The main drawback of the Viterbi search is that, it requires a large state space, which can be reduced by using Viterbi beam search.

In the Viterbi beam search all the hypothesis that fall below a chosen *beamwidth* [5] of the highest path probability are pruned away. Let P_{max} be the highest path probability and the beam-width (B < 1), then all the paths with probabilities less than P_{max} * B are pruned away. The Viterbi beam search overcomes the disadvantage of the Viterbi search reducing the search space. The best beam size is determined empirically or adaptively. To improve the performance of the Viterbi beam search alternative approaches have been proposed like partitioning the state space into subsets and subjecting them to different beam-widths. Eventually, 95% of the hypotheses are generated at the initial frames, so larger beam-widths are applied for the initial frames to prune away more hypotheses.

4.2. Stack Decoders

The stack decoding search is a depth-first technique, where hypotheses is carried until the end of the speech data is reached[1]. This algorithm is a start-synchronous search and similar to the A* search in artificial intelligence. It constructs a search tree from the language model state[1] where states in the graph are abtract states in the language model, and branches correspond to transitions between states.

To explain the stack decoding search in brief, the best hypothesis is popped from the stack to which acoustic and language model fast matches are applied. The fast matches[5] are computationally cheap methods for reducing the number of word extensions which are further checked by more accurate, but computationally expensive detailed matches. After applying the acoustic and language model detailed matches, the most likely hypothesis is identified and the hypothesis is updated accordingly and pushed into the stack. This process is repeated until the end of the sentence is reached. So it is very essential to have a flag to identify the end of the sentence.

They are many disadvantages of the stack decoding. It requires an extra function for comparing hypotheses of different lengths. It also suffers with problems like speed, size, accuracy and robustness. Though the A* search reduces the size of the stack it still grows exponentially. Pruning can be applied to minimize this problem.

4.3. N-Best Search

The Viterbi search algorithm retains only a single path at any frame to find the best hypothesis failing to take into consideration other hypotheses for future

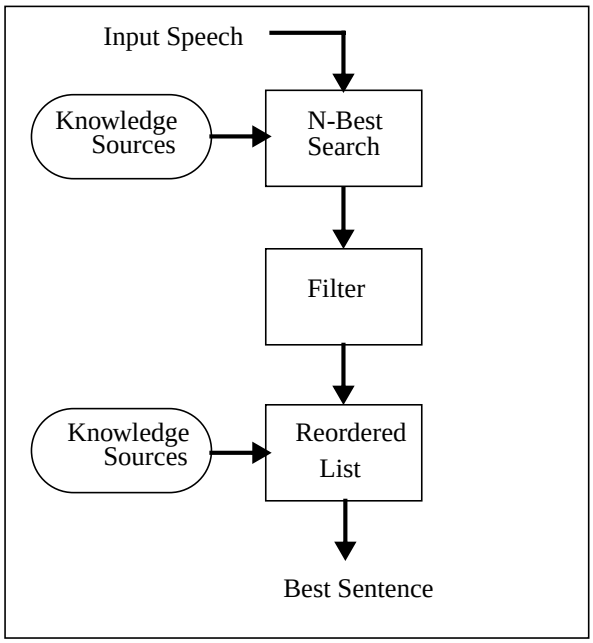

Figure 4: N-Best Search

evaluations. The simplicity of the Viterbi search enables it to perform with less computations and with less memory requirement for storage. The N-best search on the otherhand allows N-top scoring hypothesis[6] to propagate to the next state. As the number of hypotheses that are propagated to the next state increases the search space also increases. We also need large amount of memory to store the histories of all the hypotheses making it very inefficient. To overcome this knowledge sources[7] are used to obtain the best hypothesis. The most powerful knowledge sources are used initially to get a scored list of all the likely sentences. This list is then rescored again using the remaining knowledge sources to get the best hypothesis.

The N-best search is more efficient for short sentences, the recognition error increases as the size of the sentence increases. A larger N is required to obtain the same performance as a short sentence. Apart from these drawbacks it was also observed that the hypotheses in the initial stage did not differ much. A number of alternate methods have been proposed to overcome the flaws in the N-best search, which perform more accurately with fewer computations, memory requirement and with increased speed. We shall discuss about a few here.

Lattice N-Best Algorithm - The lattice N-best algorithm[7] is a time-synchronous one-best forward-pass algorithm. This algorithm is used within words and at each frame. All the scores for the frame are stored in a traceback list and sent to the next frame with a backpointer to the scores of the previous frame. The N-best sentences can be obtained by making a recursive search through the traceback list. Though the algorithm performs with high speed it suffers due to underestimates or misses high-scoring hypothesis[3].

Word-Dependent N-Best Search - The starting of any word depends only on the previous word, therefore a Word-dependent N-best search differentiates between hypotheses based on the previous word rather than the whole sequence. For each word n possible hypotheses are stored along with the name of the word and passed on to the next word with a backpointer. Once the end of the sentence is reached a recursive search is made to obtain the most likely sentence. The Word-dependent N-best search is more efficient for long sentences compared to lattice N-best algorithm. It also enables to derive the best path apart than the most likely path.

Forward-Backward Search - The forward-backward search[8,9] is a time-synchronous search which is mathematically similar to the Baum-welch forward-backward training algorithm. The algorithm uses a simplified forward pass followed by a complex backward pass. The forward search helps in increasing the efficiency and speed of the backward search. Typically a Viterbi search is used in the forward direction and N-best search in the backward direction. The Viterbi search in the forward direction outputs one hypothesis with few computations using simplified acoustics and language models. To perform the second pass (backward search) we either need to reverse the utterance and trace the state and grammar transitions from the backward or reverse the utterance along with the hidden markov models and grammar. Since the forward search is performed in real-time the time taken by the backward search is highly reduced with the help of the information that has been generated in the forward pass, increasing the overall speed of the search compared to the N-best search.

4.4. Frame-Synchronous Viterbi Search

In a frame-synchronous Viterbi search (FSVS) pruning is applied at the end of each frame allowing only a few hypotheses to propagate to the next frame. The pruning level is set according to the frame level and the application. The frame-synchronous Viterbi search differs from the Viterbi beam search where a pruning threshold is applied and hypotheses that fall in the chosen beamwidth are allowed to propagate. In the FSVS all the hypotheses are sorted in descending order of the path scores and pruned allowing a few hypotheses with top scores to propagate to the next frame or set a fixed number of top hypotheses to propagate to the next frame. Since the threshold plays a vital role in pruning at each frame, proper care should be taken in selecting the value to avoid pruning of the correct hypothesis.

5. CONCLUSION

We have considered several approaches to find the sequence of the most likely words. The performance of the system depends on the search technique, acoustic and language models. The reasons that make the search so difficult can be summarised as the lack of proper word boundaries, enormous vocabulary size, ambiguity of the acoustics of the words, presence of noise in the speech data and the influence of sound produced earlier on the current word. To overcome the effects we need to have sophisticated language and acoustic models.

The speed and memory required also play a vital role. Better recognition results can be obtained reducing the speed of the system. It is easy to improve the recognition speed and reduce memory requirement by trading away with the accuracy of the system. We can also reduce the memory requirement at the expense of the computation time. Techniques like pruning and path merging have been introduced to reduce the complexity of the search. To learn more about these techniques, we recommend to refer [10].

REFERENCES

[1] N. Deshmukh, "Efficient Search Algorithms for Large Vocabulary Continuous Speech Recognition", http://www.isip.msstate.edu/ publications/courses/ece_8463/projects/1996/ conference/, Mississippi State University, Mississippi, USA, May 15, 1996.

[2] A. Ganapathiraju, "Implementation of Viterbi Search Algorithm", http://www.isip.msstate.edu publications/courses/ece_8463/projects/1996/ conference/, Department of Electrical and Computer Engineering, Mississippi State University, Mississippi, USA, May 15, 1996.

[3] J. Picone, W.J. Ebel and N. Deshmukh, "Automated Speech Understanding: The Next Generation", Digital Signal Processing Technology, vol. CR57, pp. 101-114, January 1995.

[4] S. Renals and M. Hochberg, “Start-synchronous search for large vocabulary continuous speech recognition”, IEEE Transactions on Speech and Audio Processing, Vol 7, No 5 1999.

[5] R. K. Mosur, “Efficient Algorithms for Speech Recognition”, School of Computer Science, Computer Science Division, Carnegie Mellon University.

[6] S. Austin, R. Schwartz and P. Placeway, “The Forward-Backward Search Algorithm”, IEEE International Conference On Acoustics, Speech, and Signal Processing, Vol. 1, pp. 697-700, May 1991.

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

[8] R. Schwartz and S. Austin, “A Comparison of Several Approximate Algorithms For Finding Multiple (N-BEST) Sentence Hypotheses”, IEEE International Conference On Acoustics, Speech, and Signal Processing, Vol. 1, pp. 701-704, May 1991.

[9] P. B. Douglas, “Algorithms for an Optimal A* Search and Linearizing the Search in the Stack Decoder”, IEEE International Conference On Acoustics, Speech, and Signal Processing, Vol. 1, pp. 693-696, May 1991.

[10]N. Deshmukh, J. Picone and Y.H. Kao, "Efficient Search Strategies in Hierarchical Pattern Recognition Systems", in Proceedings of the 27th IEEE Southeastern Symposium on System Theory, pp. 89-91, Mississippi State, Mississippi, USA, March 1995.