Seminar report On

Hidden Markov Model and Speech Recognition

by

Nirav S. Uchat

Roll No: 06305906

under the guidance of

Prof. Sivakumar

Department of Computer Science and Engineering
Indian Institute of Technology, Bombay
Mumbai

Contents

Abstract

Modeling signal model for speech recognition is challenging task. It gives us great deal of information about problem being modeled. In this seminar we will see how Hidden Markov Model is used to model speech recognition application. We start with mathematical understanding of HMM followed by problem faced by it and its solution. Then we move to block diagram of speech recognition which include feature extraction, acoustic modeling and language model, which works in tandem to generate search graph. Use of HMM in acoustic modeling is explained. At the end we will look at isolated word recognizer using HMM.

1 Introduction

Real-world processes generally produce observable outputs which can be characterized as signals. The signals can be discrete in nature or continuous in nature. The signal source can be stationary (i.e., its statistical properties do not vary with time), or non-stationary (i.e., the signal properties vary over time). The signals can be pure or can be corrupted from other signal sources or by transmission distortions.[2]

A problem of fundamental interest is characterizing such signal in terms of signal model, signal model gives us

- Theoretical description of a signal processing system which can be used to process the signal and so as to provide desired output.
- It helps up to understand great deal about signal source without having to have source available.

One can classify signal model in to two types

- Deterministic Model : It exploit some known property of signal (like Amplitude of wave).
- Statistical Model : It takes statistical property of signal in to account. Example of this type of model is Gaussian Model, Poisson Model, Markov Model and Hidden Markov model.

Speech Recognition : Speech recognition is a process of converting speech signal to a sequence of word. Various approach has been used for speech recognition which include Dynamic programming and Neural Network.

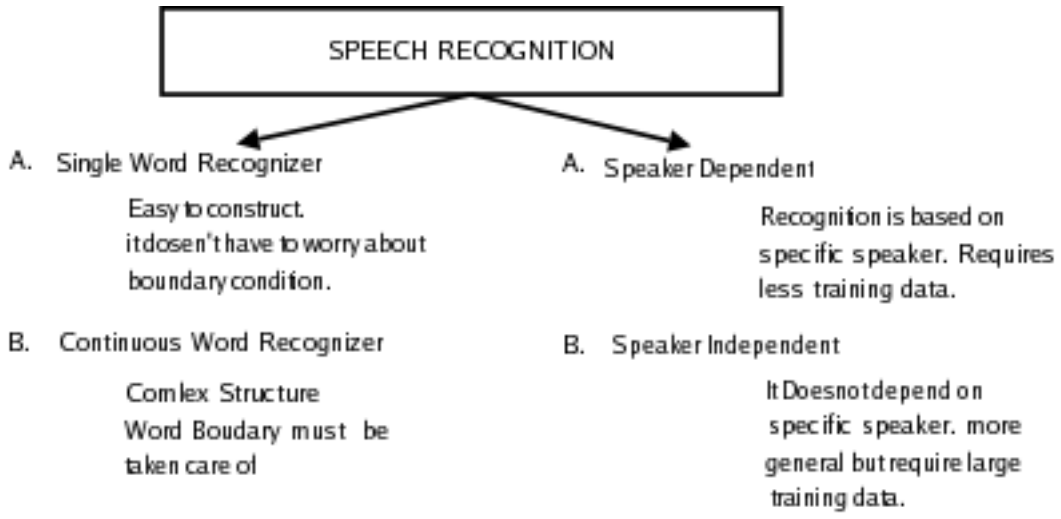

Figure 1: Classification

In this seminar we will try to bridge speech recognition and HMM and figuring out how HMM can be effectively used in speech recognition problem.

Section 2 gives mathematical understanding of Hidden Markov Model. It also focuses on three fundamental problems for HMM,namely:the probability of observation sequence given the model i.e.,$P(O/\lambda)$; the determination of single best state sequence, given the observation sequence $O = O_1, O_2, \cdots, O_T$; and the adjustment of model parameter $\lambda = (A, B, \pi)$ to maximize recognition probability. It also describe the method to efficiently solve this problems.

Section 3 explains block diagram of speech recognition system. We start with acoustic model design using vector quantization which is used to convert feature vector to symbol. It also explains how the algorithms described in 2^{nd} section are used to solve the problem associated with speech recognition. This section discuss in detail about evaluation problem and viterbi algorithm for finding "single best" state sequence. Finally bi-gram language model is explained.

In Section 4, we will apply all technique discuss in previous section to understand the working of isolated word recognizer.

2 Mathematical Understanding of Hidden Markov Model

Why Hidden Markov Model for Speech recognition ?

- HMM is very rich in mathematical structure and hence can form the theoretical basis for use in a wide range of application.
- HMM model, when applied properly work well in practice for several important application.

2.1 Discrete Markov Process

Consider a system which may be described at any time as being in one of the state of set of N distinct state, $S_1, S_2, S_3, \cdots, S_N$,. At regularly time interval system undergoes a change of state (possibly back to same state) according to set of probability associated with the state. we denote time associated with state change as t =1,2 .., and we denote actual state at time t as q_t. A full probabilistic description of above system, would in general require specification of current state and predecessor states. For the special case of discrete, first order , Markov chain, this probabilistic description is truncated to just current and the predecessor state i.e

$$P[q_t = S_j | q_{t-1} = S_i, q_{t-2} = S_k, \cdots] = P[q_t = S_j | q_{t-1} = S_i]$$

Furthermore we consider those processes in which the right hand side of above equation is independent of time , thereby leading to the set of state transition probabilities a_{ij} of the form

$$a_{ij} = P[q_t = S_i | q_{t-1} = S_j] \qquad 1 \leq i, j \leq N$$

with the state transition coefficients having the properties

$$a_{ij} \geq 0$$
$$\sum_{j=1}^{N} a_{ij} = 1$$

The above process could be called as an observable Markov Model, since the output of the process is the set of states at each instant of time, where each state correspond to a observable event.

2.2 Extension to Hidden Markov Model

In above model each state correspond to an observable event. This model is too restrictive to be applicable to many problem of interest. we now extend this idea where observation is probabilistic function of state. To get clear understanding we take coin tossing example.

Coin Toss Models: Assume the following scenario. You are in a room with a barrier through which you cannot see what is happening. On the other side of the barrier is another person who is performing a coin (or multiple coin) tossing experiment. The other person will not tell you anything about what he is doing exactly, he will only tell you the result of each coin flip. Thus a sequence of hidden coin tossing experiments is performed, with the observation sequence consisting of a series of heads and tails, for e.g., a typical observation sequence would be

$$O = O_1\ O_2\ O_3 \cdots O_T(Head\ Tail\ Head \cdots Tail)$$

to model this coin tossing event with HMM, the fundamental question is

- What the state in model correspond to
- How many state in the model should there be

To understand this in better way lets take three different HMM

1. Single biased coin with two state - each state representing head or tail
2. Two biased coin with two state - each state representing single coin
3. Three biased coin with three state - with each coin representing single coin

When single bias coin is being tossed. In this case we could model the situation with a 2-state model where each state corresponds to a side of the coin (i.e., heads or tails). This model is depicted in fig 2. In this case the Markov model is observable, and the only issue for complete specification of the model would be to decide on the best value for the probability of heads and tails.

In case 2 there are 2 states in the model and each state corresponds to a different biased coin being tossed. Each state is characterized by a probability distribution of heads and tails, and transitions between states are characterized by a state transition matrix.

Third form of HMM for explaining the observed sequence of coin toss outcomes is given in Fig. 2. This model corresponds to using 3 biased coins, and choosing from among the three, based on some probabilistic event.

Given the choice among the three models shown in Fig.2 for explaining the observed sequence of heads and tails, a natural question would be which model best matches the actual observations. It should be clear that

- simple 1-coin model has only 1 unknown parameter

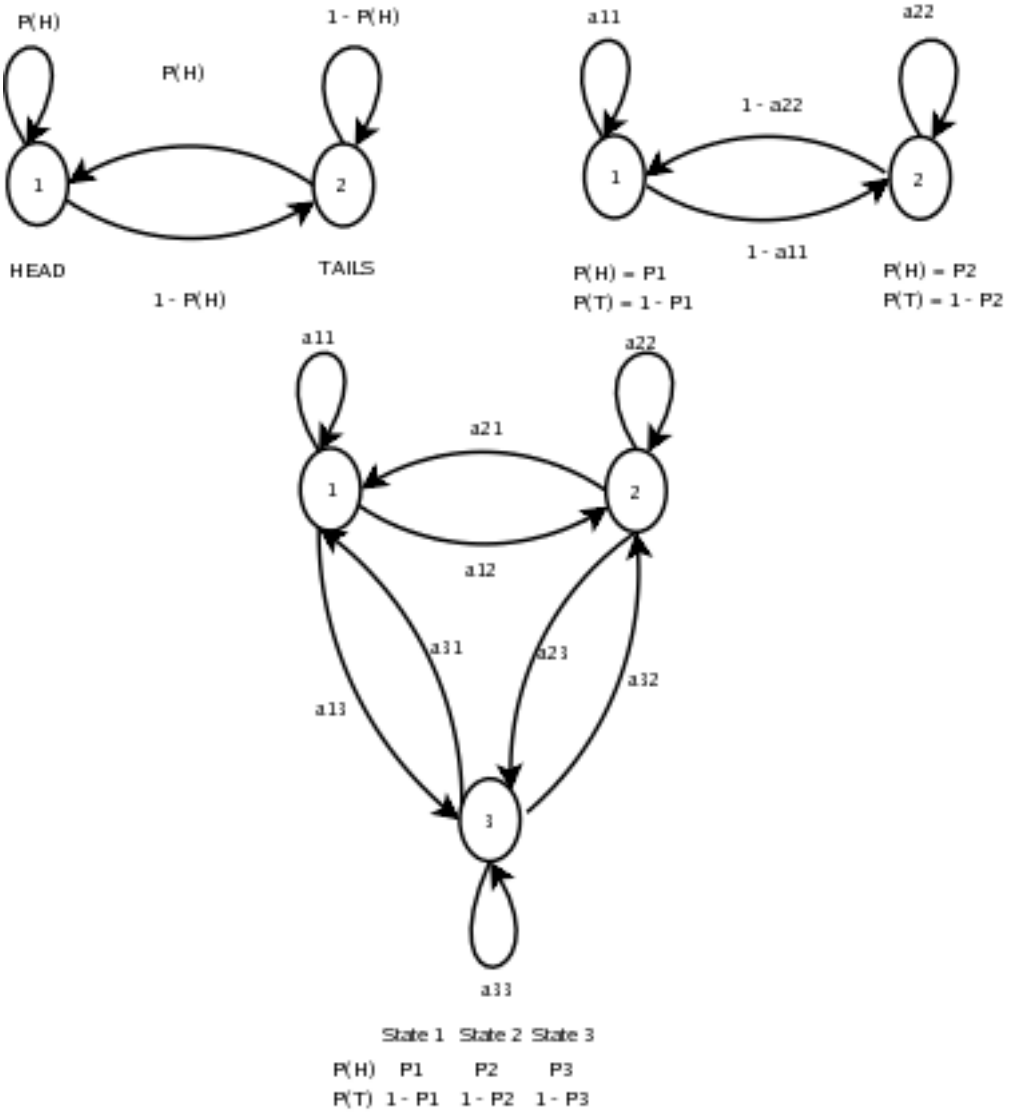

Figure 2: Three possible Markov model which can account for coin tossing experiment

- 2-coin model has 4 unknown parameters
- 3-coin model has 12 unknown parameters

Thus, with the greater degrees of freedom, the larger HMMs would seem to more capable of modeling a series of coin tossing experiments than would equivalently smaller models, but it might just be the case that only a single coin is being tossed. Then using the 3-coin model would be inappropriate, since the actual physical event would not correspond to the model being used-i.e., we would be using an underspecified system.[2]

2.3 Elements Of Hidden Markov Model

We now define elements Of HMM, HMM is characterized by following

1. Number of state N
2. Number of distinct observation symbol per state M, $V = V_1, V_2, \cdots, V_M$

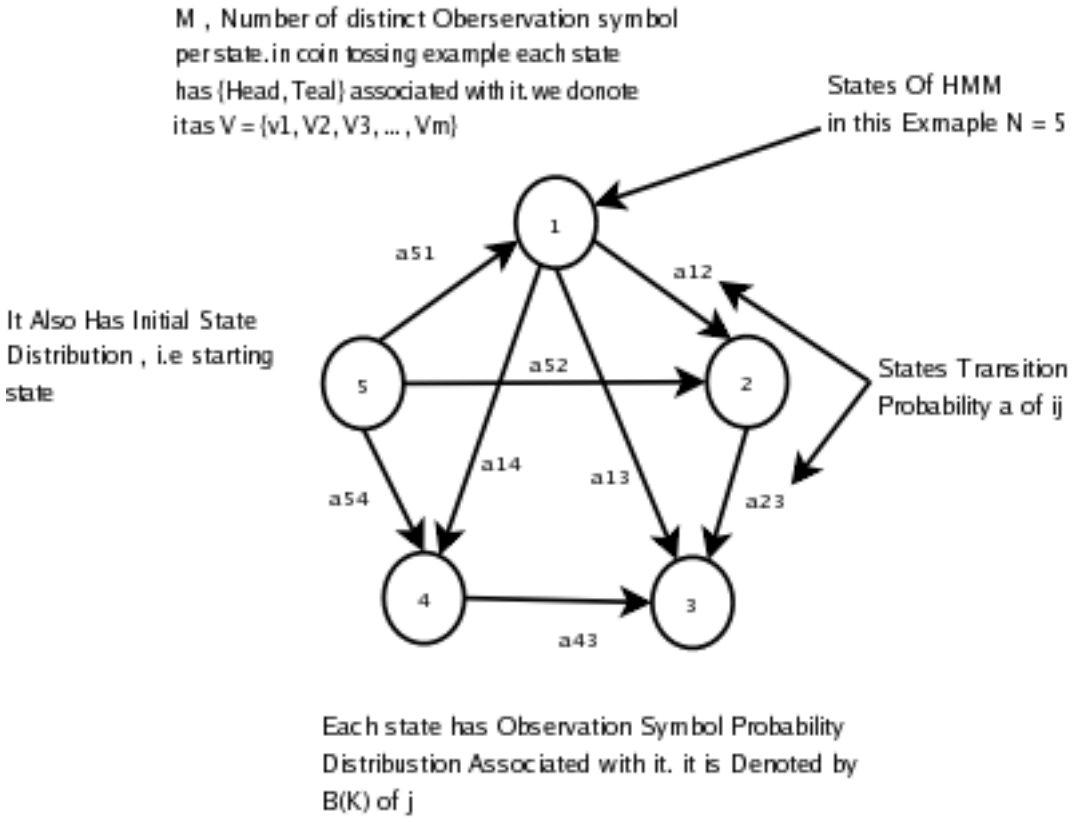

Figure 3: Elements of HMM

3. State transition probability, $a_{ij} = P[q_{t+1} = S_i | q_t = S_j],\ 1 \leq i, j \leq N$

4. Observation symbol probability distribution in state j,$B_j(K) = P[V_k \ at \ t | q_t = S_j]$

5. The initial state distribution $\pi = \pi_i$ where $\pi_i = P[q_1 = S_i] \qquad 1 \leq i \leq N[2]$

Given appropriate value of N,M,A,B and π, HMM can be used **as generator** to give an observation sequence

$$O = O_1 \ O_2 \ O_3 \cdots O_T$$

2.4 The Three Basic Problem for HMMs

Problem 1 : Evaluation Problem
Given the observation sequence $O = O_1\ O_2\ \cdots\ O_T$, and model $\lambda = (A, B, \pi)$, how do we efficiently compute $P(O|\lambda)$, the probability of observation sequence given the model.
problem 1 is evaluation problem i.e given observation sequence and model, how do we compute probability that observed sequence was produce by the model. we can think it as scoring problem. if you have to choose between many competing model then one with maximum probability will give better result.

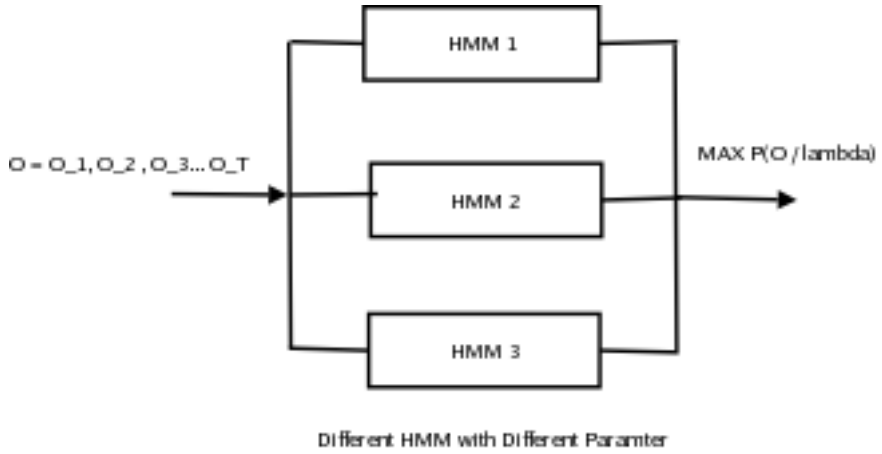

Figure 4: Evaluation Problem

Problem 2 : Hidden State Determination (Decoding)
Given the observation sequence $O = O_1\ O_2\ \cdots\ O_T$, and model $\lambda = (A, B, \pi)$, how do we choose corresponding state sequence $Q = q_1\ q_2\ \cdots\ q_T$ which is optimal in some meaningful sense.
Problem 2 is one which attempts to uncover the hidden part of the problem. There is no correct solution to this problem, in practice we usually use optimal criterion to find best possible solution. For coin tossing example with observation sequence 0 = H H T T H corresponding state sequence in each of the model (in Fig 2) can be

O = H H T T H
Model 1 : S = 1 1 2 2 1
Model 2 : S = 2 1 1 1 2
Model 3 : S = 3 1 2 3 2

Problem 3 : Learning
How do we adjust the model parameter $\lambda = (A, B, \pi)$ to maximize $P(O|\lambda)$. Problem 3 is one in which we try to optimize model parameter so as to best describe as to how given observation sequence comes out. The observation sequence used here are called "training" sequence since it is used for training HMM. Training is one of the crucial element of HMM. It allows us to adjust model parameter as to create best model for given training sequence.[2]

2.5 Solution to Three Problem

Given this problem, it's vital to solve them using efficient algorithm.

2.5.1 Forward Algorithm for Evaluation Problem $[P(O|\lambda)]$

we want to find $P(O|\lambda)$, given the observation sequence $O = O_1, O_2, O_3, \cdots, O_T$. The most straight forward way to find the solution is enumerating every possible state sequence of length T. Consider one such state sequence $Q = q_1, q_2, q_3, \cdots, q_T$ such that q_1 produces O_1 with some probability, q_2 produces O_2 with some probability and so on. so using chain rule

$$P(O|\lambda) = \sum_{q_1, q_2, \cdots, q_T} \pi_{q_1} b_{q1}(O_1) a_{q_1 q_2} b_{q_2}(O_2) \cdots a_{q_{T-1} q_T} b_{q_T}(O_T)$$

but the order of chain rule is N^T, since at every $t = 1, 2, \cdots, T$, there are N possible states which can be reached. This is clearly and inefficient algorithm, to over come this forward algorithm is used.

Forward Algorithm: Forward algorithm is a dynamic algorithm which uses forward variable $\alpha_t(i)$ defined as

$$\alpha_t(i) = P(O_1, O_2, \cdots, O_i, q_t = S_i|\lambda)$$

i.e., the probability of partial observation sequence, O_1, O_2 till O_t and state S_i at time t given the model λ, as we are taking partial observation in to account, we can solve $\alpha_t(i)$ inductively as

Initialization:

$$\alpha_1(i) = \pi_i b_i(O_1), \qquad 1 \leq i \leq N.$$

Induction:

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j(O_{t+1}), \qquad 1 \leq t \leq T-1, \quad 1 \leq j \leq N$$

Termination:

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i)$$

since $\alpha_t(i)$ is the probability of partial observation sequence O_1 to O_t and the state at time t is S_i, then the product $\alpha_i(t)a_{ij}$ is the probability of joint event that the observation O_1 to O_t are observed and state S_j is reached at time $t+1$ via state S_i at time t. Summing over all N possible state S_i $(1 \leq i \leq N)$, at time t, results in the probability of S_j at time $t+1$. By multiplying this quantity with $b_j(O_{t+1})$, we can find $\alpha_{t+1}(j)$. This computation is performed for all state $j(1 \leq j \leq N)$ for a given t and it is iterated through $t = 1, 2, 3, \cdots, T-1$. Finally the required $P(O|\lambda)$ is sum of the terminal forward variables $\alpha_T(i)$, this is true because

$$\alpha_T(i) = P(O_1, O_2, \cdots, O_T, q_T = S_i|\lambda)$$

results shows that this algorithm requires of the order of N^2T calculation rather then N^T in case of chin rule.[1][2]

2.5.2 Viterbi Algorithm for Decoding Hidden State Sequence $P(Q, O|\lambda)$

There are various way in which optimum solution can be calculated. The difficulty lies in the definition of optimum state sequence. One way to find optimum state sequence is to choose the states q_t which are individually most likely. But this has serious flaw in sense that if two states

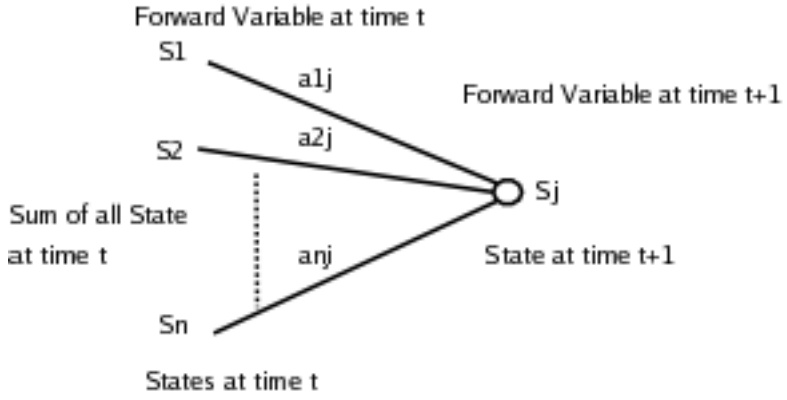

Figure 5: Forward variable calculation

i and j are selected such that $a_{ij} = 0$, then despite of most likely state at time t and $t + 1$, it is not valid state sequence. So deciding optimum criteria is very crucial. There is a way to find the "single best" state sequence is using dynamic programming algorithm, called as viterbi algorithm.

Viterbi Algorithm: It says that to find single best state sequence, $Q = q_1, q_2.q_3, \cdots, q_t$, (which produces given observation sequence) for a given observation sequence $O = o1, o_2, o_3, \cdots, o_t$, we define a quantity

$$\delta_t(i) = \max_{q_1, q_2, \cdots, q_{t-1}} P[q_1 q_2 \cdots q_t = i, O_1 O_2 \cdots O_t | \lambda]$$

i.e., $\delta_t(i)$ is the best score along a single path, at time t, which account for the first t observations and ends in state S_i, by induction

$$\delta_{t+1}(j) = \left[\max_i \delta_t(i) a_{ij}\right] b_j(O_{t+1})$$

in order to find the state sequence we need to keep track of state which **maximize** the above equation. we do this via array $\psi_t(j)$ for each t and stat j. Once the final state is reached corresponding state sequence can be found out using backtracking. Key point is, **Viterbi algorithm** is similar (except for the backtracking part) in implementation to the **Forward algorithm**. The major difference is maximization of the previous state in place of summing procedure in forward calculation.[1][2]

2.5.3 Baum-Welch Algorithm for Learning

Most challenging of all is to adjust the model parameter (A, B, λ) to maximize the probability of the observation sequence given the model. Given any finite observation sequence as training data, there is no optimal way of estimating the model parameters. But there are some heuristic procedure which try to find local optimization over global optimization.[2]

Re-estimation Procedure: In order to define a procedure for re-estimation of HMM parameter, we first define

$$\xi_t(i, j) = P(q_t = S_i, q_{t+1} = S_j | O, \lambda)$$

i.e.,the probability of being in state S_i at time t, and state S_j at time $t + 1$ given the model and observation sequence we can write

$$\xi_t(i,j) = \frac{\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)}{P(O|\lambda)}$$

where $\beta_t(j)$ = backward variable , i.e., probability of the partial observation sequence $t+1$ to end (T), given state S_i at time t and model. now we define $\gamma_t(i)$ as the probability of being in state S_i at time t, given the observation sequence and model; hence we can relate $\gamma_t(i)$ to $\xi_t(i,j)$ by summing over j, such that

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i,j)$$

if we sum $\gamma_t(i)$ over time index we get a quantity which can be interpreted as expected number of time state S_i is visited, or expected number of transition made from S_i. Similarly summation of $\xi_t(i,j)$ over time can be interpreted as the expected number of transition made from state S_i to state S_j. That is

$$\sum_{t=1}^{T-1} \gamma_t(i) = \text{expected number of transitions from } S_i$$

$$\sum_{t=1}^{T-1} \xi_t(i,j) = \text{expected number of transitions form } S_i \text{ to } S_j$$

Using above formula we can give a method for re-estimation of the parameter of an HMM.

$$\bar{\pi} = \text{Expected number of times in state } S_i \text{ at time (t=1)} = \gamma_1(i)$$

$$\bar{a}_{ij} = \frac{\text{expected number of transition from state } S_i \text{ to } S_j}{\text{expected number of transition form state } S_i}$$

$$= \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}$$

$$\bar{b}_j(k) = \frac{\text{expected number of times in state j and observing symbol } v_k}{\text{expected number of times in state } j}$$

$$= \frac{\sum_{\substack{t=1 \\ s.t\ O_t = V_k}}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}$$

Using this three algorithm we can solve all three problem associated with HMM. In next section we will see how this algorithm are used in speech recognition using HMM.

3 Speech Recognition and HMM

3.1 Block Diagram Of Speech Recognition

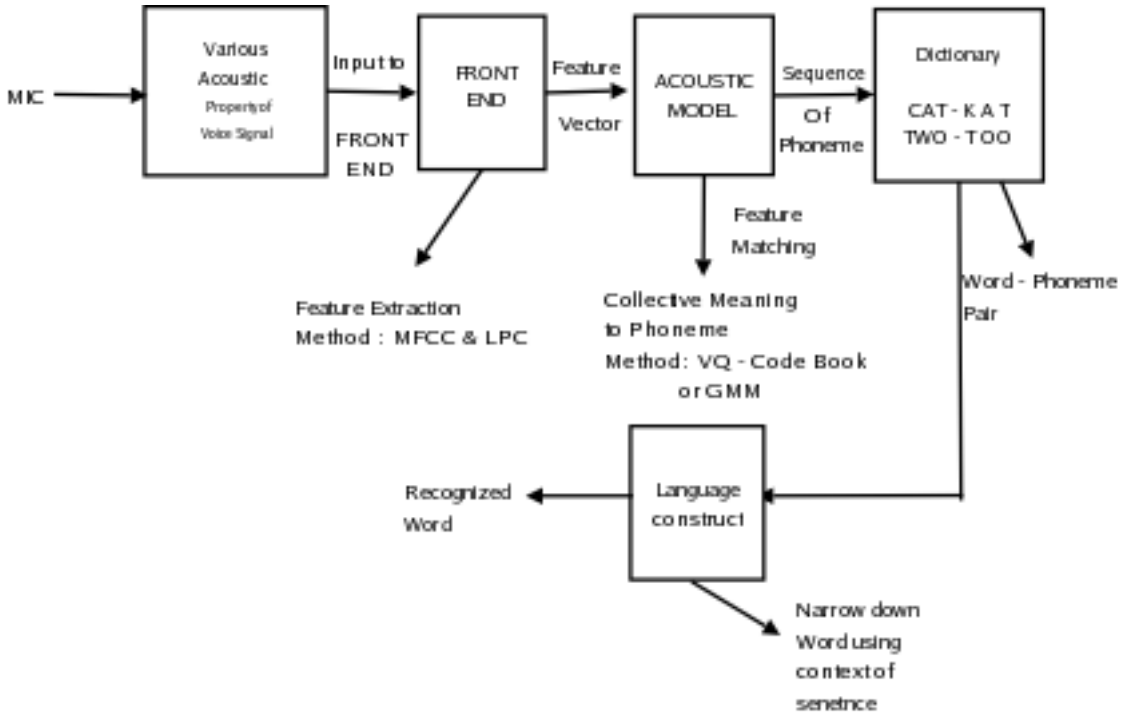

Figure 6: Block Diagram of Speech Recognition

Speech recognition consists of two main modules, feature extraction and feature matching. The purpose of feature extraction module is to convert speech waveform to some type of representation for further analysis and processing, this extracted information is known as feature vector. The process of converting voice signal to feature vector is done by signal-processing front end module. As shown in above block diagram input to front-end is noise free voice sample and output of it is feature vector.
In feature matching, the extracted feature vector from unknown voice sample is scored against acoustic model, the model with max score wins ,and it's output is considered as recognized word. Following are the few method for implementing front-end(for extracting feature factor)

- MFCC (Mel-Frequency Cepstrum Coefficient)
- LPC (Linear Predictive Coding)

Once the feature vector are obtained we build the acoustic model. The acoustic model is used to score the unknown voice sample. As shown in block diagram, Output of front-end is given as input to acoustic model. Different types of acoustic model are

- VQ-Code Book [1]
- GMM-Gaussian Mixture Model

Acoustic Model Representation : In speech recognition, basic unit of sound is phoneme. Phoneme is a minimal unit that serves to distinguish between meanings of words. For example

sequence of phoneme for "CAT" is K A and T. In English language there are nearly around 46 phoneme. We can construct any word from English dictionary using proper concatenation of this phoneme. In order to recognize a given word, we should extract phoneme from voice sample.

Due of slowly timed varying nature of speech signal short-term spectral analysis is the most common ways to characterize speech signal. When examined over a sufficiently short period of time(between 10 and 25 msec), its characteristics are fairly stationary. However, over long period of time the signal characteristic change to reflect the different speech sound being spoken. Using this observation, we find that feature vector extracted over 10 to 25 msec correspond to **single phoneme**. For speech recognition in HMM, we assign each basic unit(phoneme) a unique HMM. Study shows that each phoneme HMM can be represented by three state i.e. begin, middle and end state. as shown in the Fig. 7. Furthermore state correspond to feature

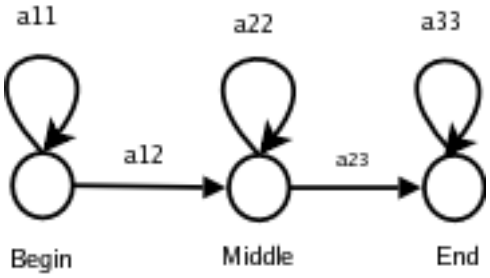

Figure 7: HMM for single Phoneme

vector. To understand this take example of isolated word recognizer, Each word in vocabulary has distinct HMM. When unknown word comes, it is scored against all HMM model and HMM with maximum score is considered as recognized word. As shown in block diagram output of acoustic model is sequence of phoneme. By looking dictionary in reverse way (phoneme - word) we can find corresponding word. But in general there are many word with same phoneme sequence, for example *car key* and *Khakee* has same phoneme sequence. In such case language structure comes in to picture. Language structure uses context information to narrow down the recognized word to resemble the given grammar construct.

This type of model is known as mono-phone or context-independent model, following are different types of HMM

1. **Context-Independent Phoneme HMM**
   - Number Of State : d-state HMM for each phoneme (d is normally equal to 3)
   - Accuracy : not accurate in continuous speech recognition
   - Compact : d-state HMM lead to less parameter to be calculated
   - General : Yes, we can build HMM for new word using existing phoneme HMM
2. **Context-Dependent Triphone HMM**
   - Number Of State : d-state HMM for each phoneme
   - Accuracy : Accurate, as it has left-right phoneme relation

- Compact : Each phoneme has immediate left-right relation, more parameter needs to be calculated
- General : Yes

3. **Whole-Word HMM**

   - Number Of state : No phoneme generation
   - assign number of state to model a word as whole
   - Accurate : Yes, require large training data and work for small vocabulary
   - Compact : No, need too many state as vocabulary increases
   - General : No, can't build new word using this representation

In practice triphone model is widely used in speech recognition using Hidden Markov Model. Now we are not going to discuss how feature vector are extracted from voice signal. we assume that using either MFCC or LPC we get required feature vector. In case of MFCC it is 39 dimension vector. The question is **how to map feature vector to HMM state ?**
for doing so we use technique called Vector Quantization.

3.2 Vector quantization

It is a technique for mapping MFCC type feature vector to symbol.

- create training set of feature vector
- cluster them in to small number of classes
- represent each class by symbol
- for each class V_k, compute the probability that it is generated by given HMM state.

In Vector Quantization, we define a codebook which contain entry for each symbol, which is also known as prototype vector or codeword. for example if we have 256 classes(i.e. 8 bit VQ), we make 256 entry in codebook. As shown in Fig.8 the incoming vector is compared with each prototype vector and one with minimum distance is chosen and it's index value is given to the input vector.[1]

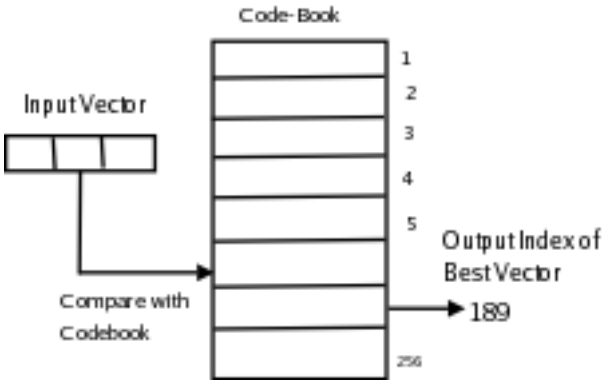

Figure 8: Vector Quantization

3.2.1 How to generate code book(K-means clustering algorithm)

- choose M vector from L training Vector - where $M = 2^B$ as initial code words (Choose M with MAX distance between them)
- for each training vector, find the closest code word (minimum distance). Assign this training vector to that cell
- for each cell, compute centroid of that cell. The new codeword is the centroid
- repeat last two step until average distance falls below threshold

As shown in Fig.9, all vectors are clustered around newly formed centroid.

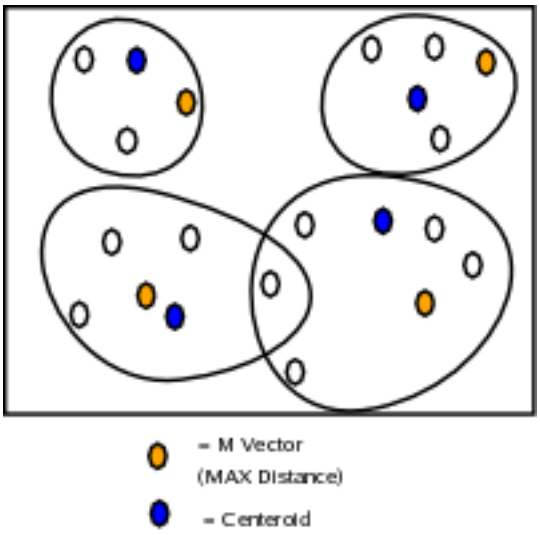

Figure 9: VQ Example
citecite

3.2.2 Searching algorithm

- To compute $p(o_t|q_j)$
- compute distance between feature vector o_t and each codeword entry (prototype vector)
- Choose the vector that is the closest to o_t and take its codeword v_k
- look up the likelihood of v_k given HMM state j in the matrix b
- i.e. $b_j(o_t) = b_j(v_k) s.t. v_k$ is codeword of closest vector to o_t

For example there are total 60 vectors mapped to state j out of that 20 are indexed to k in code word, then probability of $b_j(v_k)$ is

$$b_j(v_k) = 20/60$$

3.3 Evaluation Problem and Viterbi Algorithm

In Section 2 we saw evaluation problem i.e., given observation sequence $O = O_1, O_2, O_3, \cdots, O_T$ how to calculate $P(O|\lambda)$. There we apply chain rule for calculating $P(O|\lambda)$ as

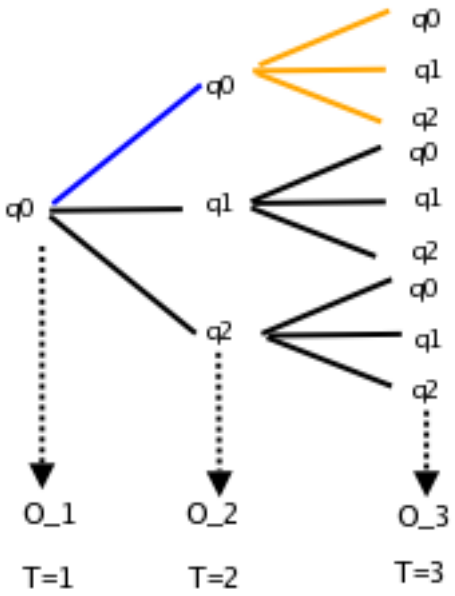

Figure 10: chain rule - intractable algorithm

$$P(O|\lambda) = \sum_{overallstate} \pi_{q_1} b_{q1}(O_1) a_{q_1 q_2} b_{q_2}(O_2) \cdots a_{q_{T-1} q_T} b_{q_T}(O_T)$$

which is an order of $O(N^T)$.

As shown in Fig.10 probability of $P(O_1, O_2, O_3)$ over all state sequence is

$$\begin{gathered} P(o_1, o_2, o_3|q_0, q_0, q_0) \\ + \\ P(o_1, o_2, o_3|q_0, q_0, q_1) \\ + \\ P(o_1, o_2, o_3|q_0, q_1, q_0) \\ + \\ P(o_1, o_2, o_3|q_0, q_1, q_1) \ .... \end{gathered}$$

For better efficiency we use forward variable which uses dynamic programming technique to reduce order to N^2T. In section 2.5.2 we defined forward variable such that

$$\alpha_t(i) = P(O_1, O_2, \cdots, O_i, q_t = S_i|\lambda)$$

and computation of $\alpha_t(j)$ by summing all previous values $\alpha_{t-1}(i)$ for all i. Fig.11 shows the calculation of word "ONE" whose phoneme sequence is W AH and N. In forward calculation we consider summing of all incoming node to find $P(O|\lambda)$. Since our phoneme model is **bi-gram model**, only it's predecessor states input is consider to calculate sum of probability. For better understanding refer Fig.11 at time t=3, input to state N is from state AH and N only and not from W. In this way we can find the $P(O|\lambda)$, given an observation sequence and model λ.

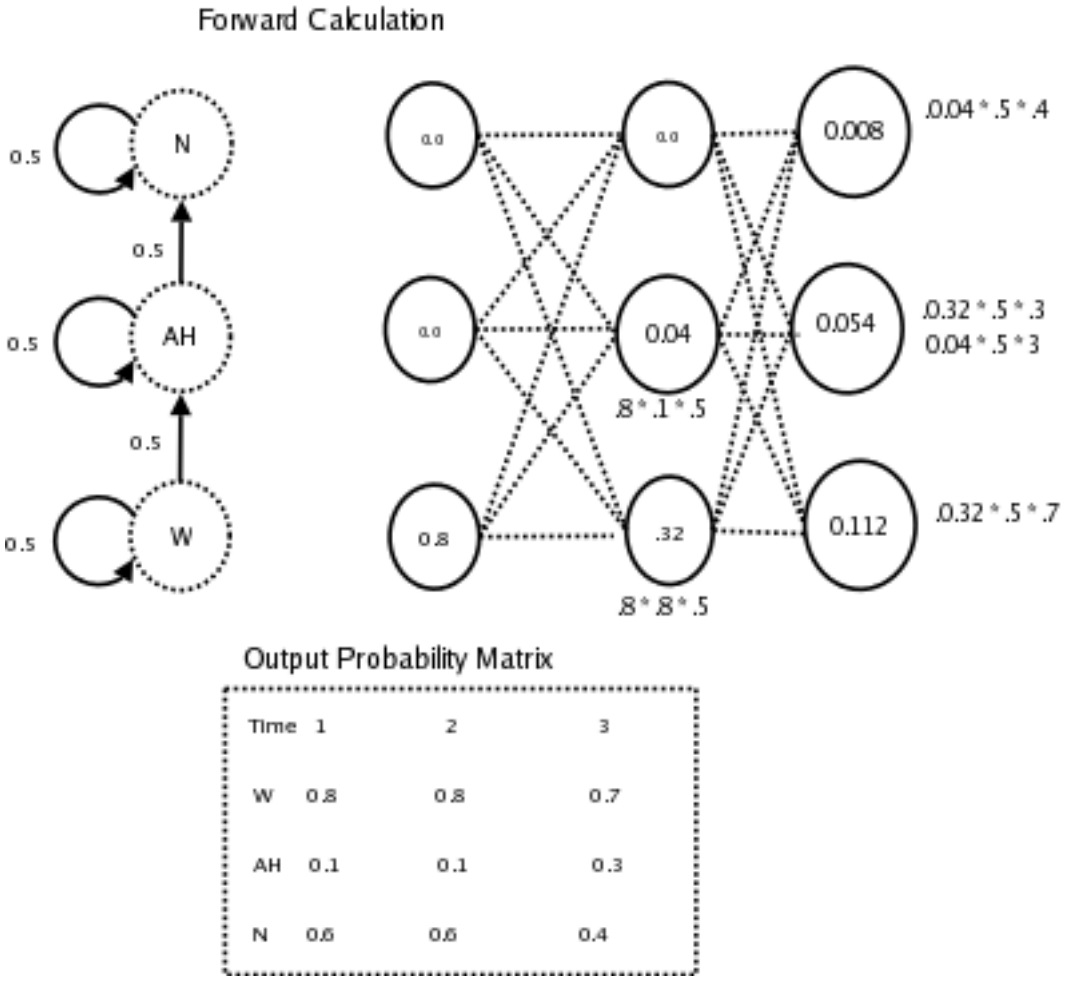

Figure 11: Forward calculation

Now we will look at Viterbi algorithm for finding best state sequence. As explained in section 2.5.2 Viterbi algorithm is same (except backtracking)in implementation as of forward algorithm. Major difference is, in Viterbi we take MAX of all incoming input for given state. As shown in Fig.12 line from state 1 at time t=1 to state 2 at t=3 is considered for probability calculation, while in Forward algorithm we sum up all probability.
So till now we have seen

- How feature vector are mapped on to HMM state, using Vector Quantization
- How to calculate probability of observation sequence given model i.e., $P(O|\lambda)$,using Forward algorithm
- How to find single best state sequence given observation sequence and model i.e., sequence of states which produces required output, using Viterbi algorithm

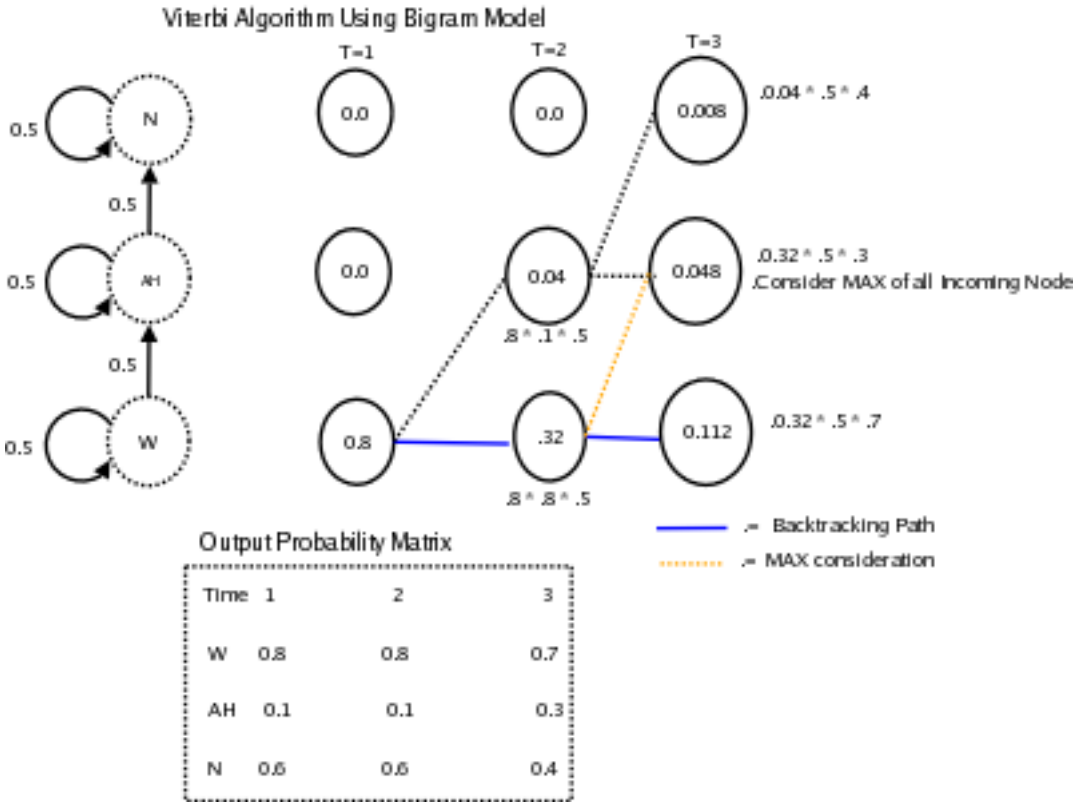

Figure 12: Viterbi Calculation

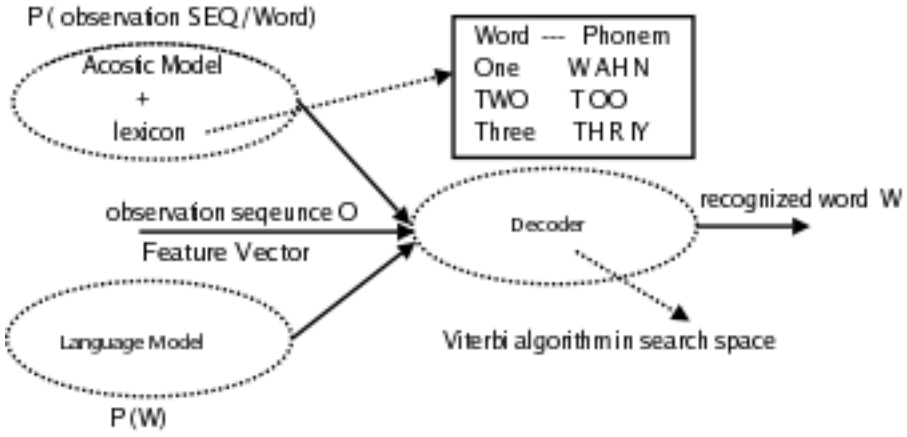

Figure 13: Language model, acoustic model and decoder

3.4 Language model

The model that compute $P(W)$ is known as Language model. Fig.13 gives how Acoustic model, Decoder and Language model work together to recognize given sentence/word W. Given a sentence how one can calculate the joint probability. One way is to use chain rule. In general $P(A, B, C, D) = P(A).P(B|A).P(C|A, B).P(D|A, B, C)$. For an example what is the probability of P(computer,got,crashed,while,installing,windows). Using above rule we can write

P(computer)* P(got/computer)* P(crashed/computer,got)* P(while/computer,got,crashed)* P(installing/computer,got,crashed,while)* P(windows/computer,got,crashed,while,installing)

In practice we will never get data to calculate the probability of long prefix such as P(windows/computer,got,crashed,while,installing)
In order to model practical application we can make assumption of calculating P(got/computer), P(crashed/got), P(windows/installing) and so on. This is known as bi-gram model. In general we write,

$$P(W_n|W_{n-1})$$

Language model help us in generating search graph. Search graph is collection of HMM of each phoneme in given vocabulary. Once the graph is created, incoming feature vector are quantized and search is performed to find best sequence of phoneme. Fig.14 will give better understanding of search graph.

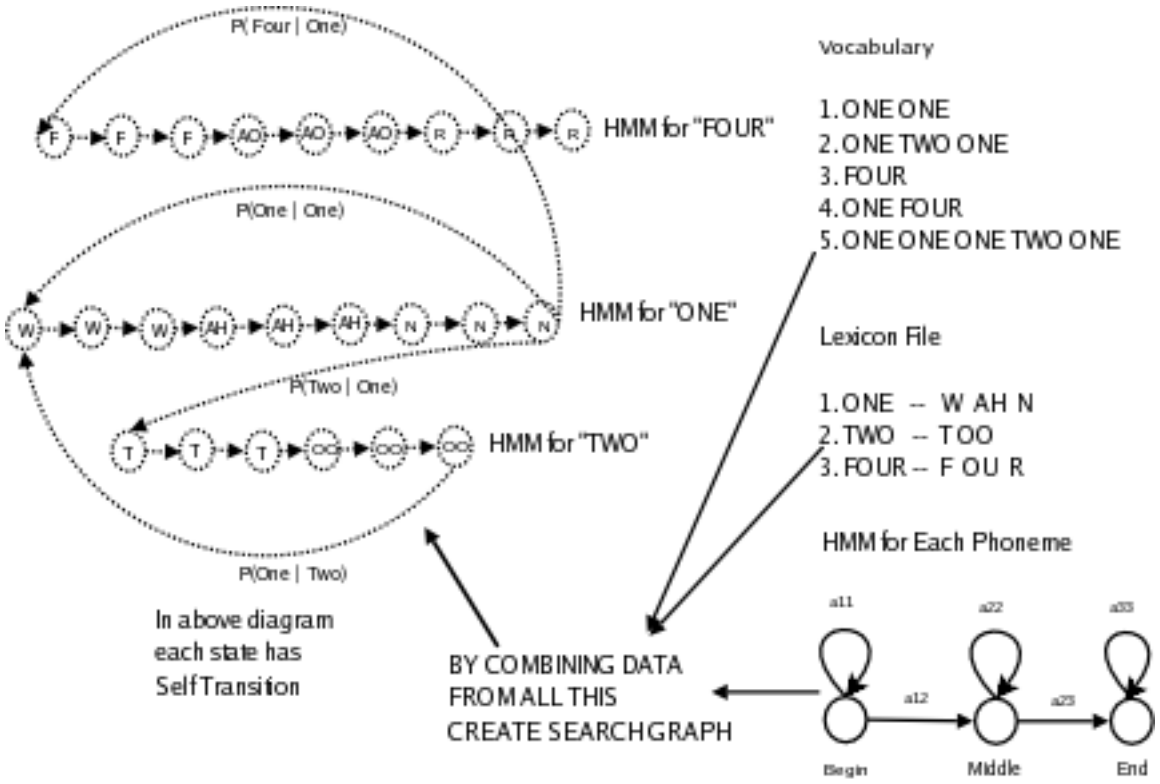

Figure 14: Search graph for given vocabulary

4 Isolated Word Recognizer

In this section we will quickly go through all the components used in speech recognition application.

4.1 Architecture Of speech recognition

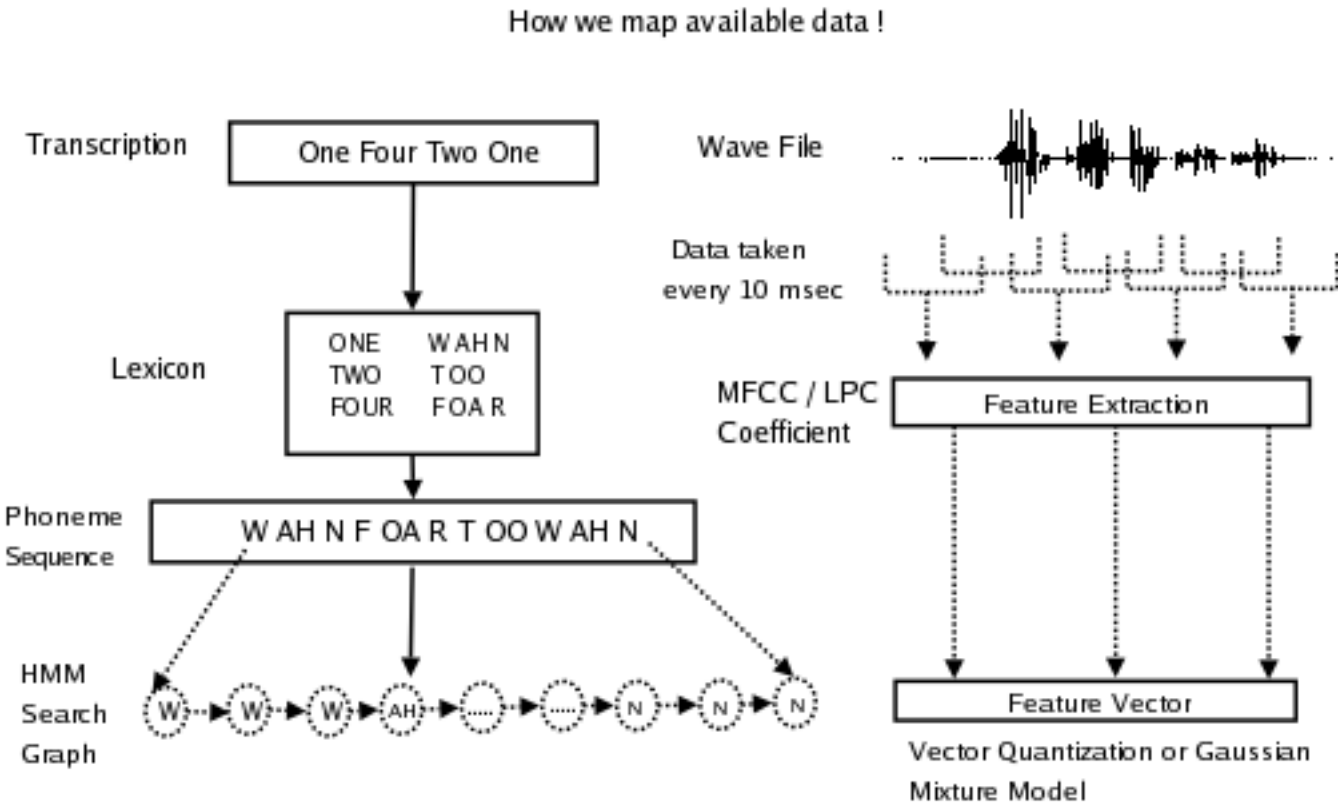

Figure 15: Architecture

Front-End: The purpose of the Front-End is to parametrize an Input signal (e.g., audio) into a sequence of output Features. Voice samples are taken every 10 - 25 msec. This sample data is feed to the Front-End module for further processing. Output of Front-End is list of feature vector (MFCC - 39D). This feature vector are then mapped to symbol using vector quantization.[3]

Vector Quantization: It maps Feature vector to symbol. This is also knows as acoustic modeling. This symbols represent HMM state. During recognition process this symbol are matched against unknown symbols. This gives us way to map complex vector in to manageable symbol set. This method is discussed in section 3.2.

HMM model creation: Depending on implementation, HMMs are created for every basic sound unit, in our case it is Phoneme. Further all HMM are linked together to represent the vocabulary under consideration. This linked representation is known as search space for given problem. During recognition phase this graph is searched for finding occurrence of given word.

To represent relation between the words in given sentence, bi-gram model is used. Language model, vocabulary and acoustic modeling forms the core of speech recognition. Lexicon (Fig.13) are used to map word from vocabulary to HMM. For example consider word "ONE", in lexicon listing we have it's corresponding phoneme sequence.

Training: The most difficult task is to adjust the model parameter to accurately represent the word under consideration. In training mode large amount of voice data (from different speaker) is given to HMM model. Using this, HMM adjust its probability distribution and transition matrix. There is no global optimal algorithm for learning. Every HMM must be trained to maximize it's (local optimum) recognition power. Initially HMM for phoneme (before learning) consists of 3-state and it's adjacency matrix and output probability distribution are initialized randomly. It gets automatically updated once the training starts.[4]

Recognition : unknown word is fed to HMM and it's output probability is calculated.

Five steps for automatic speech recognition(ASR)

- Feature Extraction: 39 MFCC features
- Acoustic Model: Vector Quantization - Output probability distribution for each state
- Lexicon: Word to Phoneme mapping - Used in creation of HMM
- Language Model: N-grams for computing $P(w_i|w_{i-1})$
- Decoder: Viterbi algorithm, using dynamic programming for combining all these to get word sequence from speech!

4.2 Example

The role of the recognizer is to do mapping between sequences of speech vectors and the wanted underlying symbol sequences. Two problems make this very difficult

- The mapping from symbols to speech is not one-to-one since different underlying symbols can give rise to similar speech sounds(example of car key and khakee having same phoneme sequence). Furthermore, there are large variations in the speech waveform due to speaker variability, noise, environment, etc.
- The boundaries between symbols cannot be identified explicitly from the speech waveform. Hence, it is not possible to treat the speech waveform as a sequence of concatenated patterns.

The second problem of not knowing the word boundary locations can be avoided by restricting the task to **Isolated word recognition** (Fig.16). In this the speech waveform corresponds to a single word chosen from a fixed vocabulary. Despite it is very artificial, it has wide range of practical applications. It serves as basic model to understand the concept of speech recognition.

We Assume that parameter a_{ij} and $b_j(o_t)$ are known for each model M_i. Given a set of example for a given word, HMM model parameter can be calculated automatically using learning algorithm discussed in section 2.5.3. Fig 17 summarize the use of HMM in isolated word

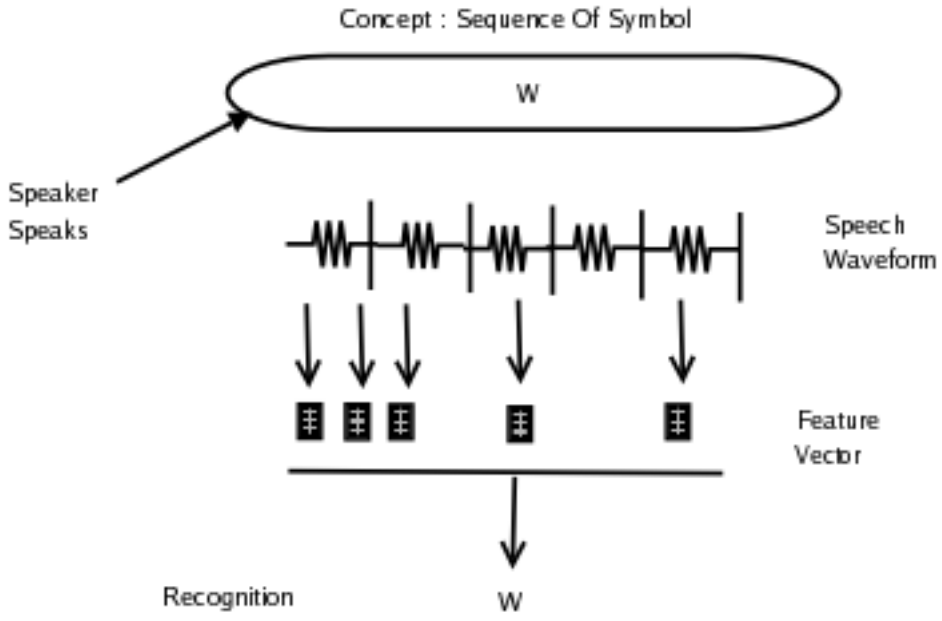

Figure 16: Isolated word recognizer

recognizer. Firstly, a HMM is trained for each vocabulary word using number of example of that word(this will adjust a_{ij} and $b_j(v_k)$). In this example the words are cat,dial and call respectively. In order ot recognize some unknown word, observation sequence is given to each HMM, output of HMM with maximum $P(O|\lambda)$ (it is calculated using Forward algorithm) is considered as recognized word. [4]

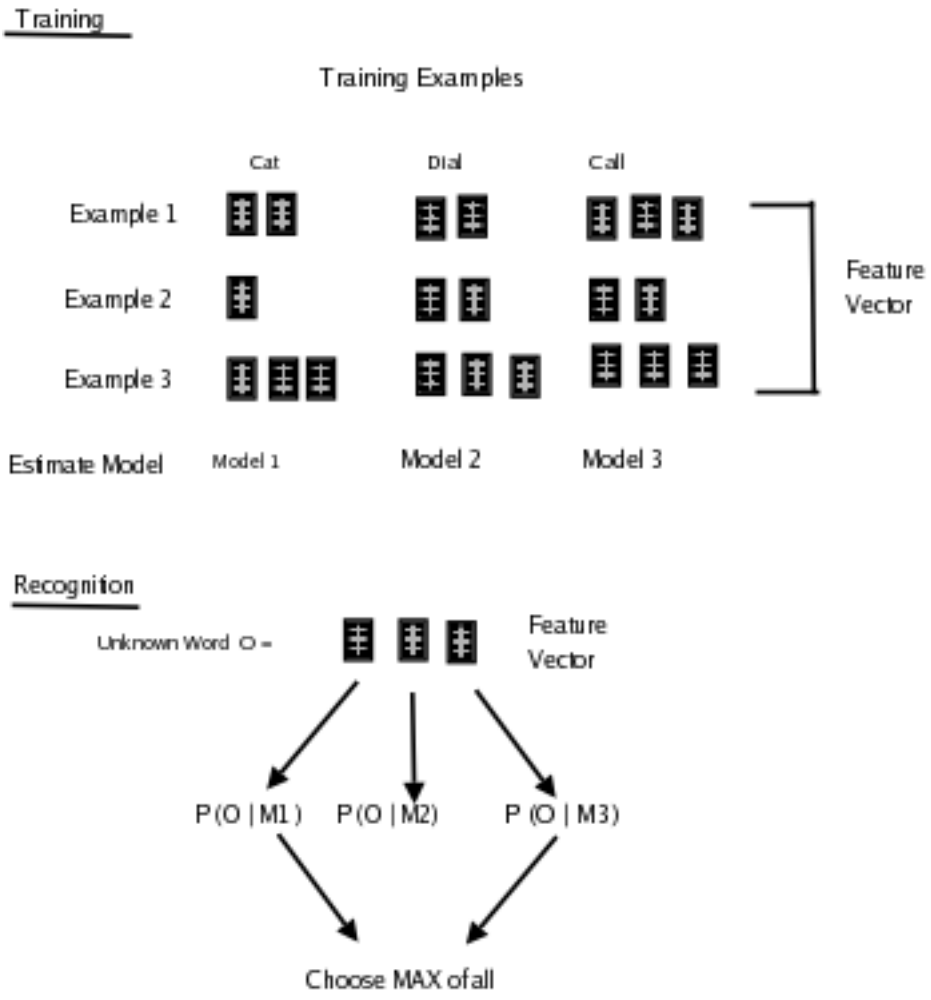

Figure 17: Markov Model for isolated word recognizer

5 Conclusion and Future Work

We saw how HMM can be used as signal model for speech recognition application. Forward, Viterbi and Baum-Welch algorithm provide solution to three problem associated with HMM. Speech recognition using HMM gives good result due to resemblance between architecture of HMM and varying speech data. Neural Network is another method, which uses gradient decent method with back propagation algorithm. Study shows that this method works well in presence of large amount of training data also the recognition accuracy is high if the word under consideration is from training data set. While in HMM, the recognition ability is good for unknown word. HMM is generic concept and is used in many area of research.

In whole architecture of speech recognition, HMM is just one block which help in creating search graph. It work in tandem with other block such as front-end, language model, lexicon to achieve desired goal. The purpose of HMM is to map feature vector to some representable state and emit symbol, concatenation of which gives desired phoneme sequence.

Problem with continuous speech recognition is, to determine word boundary. It requires knowledge of language construct and regional accent understanding. Other problem is presence of noise in sample data. Efficient solution to this two problem is required for accurate continuous speech recognition.

What we discussed in this seminar is recognition of English sentence, we can extend this model to recognize multi-lingual sentence.

References

[1] Dan Jurafsky. *CS 224S / LINGUIST 181 Speech Recognition and Synthesis*. World Wide Web, http://www.stanford.edu/class/cs224s/.

[2] Lawrence R. Rabiner. *A Tutorial on Hidden Markov Model and Selected Applicaiton in Speech Recognition*. IEEE, 1989.

[3] Willie Walker, Paul Lamere, and Philip Kwok. *Sphinx-4: A Flexible Open Source Framework for Speech Recognition*. SUN Microsystem, 2004.

[4] Steve Young and Gunnar Evermannl. *The HTK Book*. Microsoft Corporation, 2005.