SPEAKER DEPENDENT BENGALI KEYWORD SPOTTING IN UNCONSTRAINED ENGLISH SPEECH

By
Samarjit Das.

Bachelor of Technology, 3rd year
Indian Institute of Technology
Guwahati, India.

A project report submitted during summer internship under the supervision of Prof. P.C Ching, Dept of Electronic Engineering, The Chinese University of Hong Kong.

Department of Electronic Engineering
The Chinese University of Hong Kong.
Hong Kong, July 2005.

Abstract

Multi-lingual interfaces can be of great use in a number of applications. A very important issue for such systems is to first identify the segments of utterances corresponding to a specific language. Language boundary information is also very vital before any further processing can be done. Language specific keyword spotting can be used for this purpose. Thus such a word spotter can serve as an integral part of a typical multi-lingual system.

A speaker dependent 'Bengali' keyword spotter in unconstrained 'English' speech had been developed in this project. Two approaches were used. Both used whole word based HMMs for keywords. All the Bengali keywords were trained as isolated words. The first approach used whole word filler model. The second approach used trained English phoneme models with an all phone grammar network to model the filler part. For whole word based approach an optimal performance of 94.22% hit with 1.17 FA/KW/H was obtained while the maximum %hit for the same system was 97.92% but at the cost of 7.03 FA/KW/H. The second approach attained an optimal performance with hit rate of 95.83% with just 0.71 FA/KW/H. However, maximum %hit for this system was same as first approach but with lesser false alarm rate of 4.45 FA/KW/H. Performance improvements in terms of reduction of false alarms have also been proposed. Finally, further development of the existing system to a 'speaker independent Bengali keyword spotter' has been discussed.

Acknowledgement

I am greatly indebted to my supervisor, Prof P.C Ching, for providing me such a great opportunity of working in a wonderful environment. Also, I am utmost thankful to him for his support and thoughtful guidance throughout my internship. Without him, this work would have never been complete.

Thanks also to Prof. Tan Lee, who has given many valuable advice and suggestions.

Finally, I would like to express my sincere gratitude to my lab-mates/friends at the "DSP & Speech Technology Laboratory", who has made my stay here in Hong Kong very memorable.

I. Table of Contents

page no

II. List of Figures

page no

III. List of Tables

page no

Chapter 1

Introduction

1.1 Multilingual interfaces

Past decade has seen an exponential growth in the field of digital technology and information processing. With the rapid development of the very large scale integrated circuit (VLSI) technology, we have been able to achieve very high computation and data processing speed. This has made real-life applications of more and more complex algorithms a reality. Speech recognition algorithms are one of the most important among them. Nowadays, state of the art speech recognition technology is finding its place in many multimedia applications to spoken language command interfaces, information query systems and in many more like biometric authentications, surveillance [4] etc.

As speech driven interfaces are going to take over today's conventional interfaces over the next couple of years, we have to address some very important issues related to that. Most of the speech interfaces that exist today essentially expect the user to speak in a particular language. Or in other words, they are basically monolingual. But as the spread of application increases, quite often we have to come across the type of users whose utterance contains words from multiple (more likely to be 'two') languages. A very good example might be a user who has a 'weak' English background. This scene is very common in many countries where either English is not that widely spoken(example, France) or many people are less 'English-educated'(example, India) than the 'system might expect'. It is always going to happen that he might mix up his own native language when tries to use the interface in English. This is true for speech command interfaces and also for information query systems as well.

Due to the reasons stated above the need for systems/interfaces that can handle multilingual behavior of speech will increase with the course of time. One thing is pretty clear that a bi-lingual utterance will have a dominance of one specific language compared to the other. It means that, the words corresponding to the later will have occurrences at unpredicted intervals and at any part of the utterance. One important phase of a desired multilingual system will be to single out those words from a specific language as a preliminary step. Once the system has 'Spotted' the words from a 'different' language then it can take appropriate actions, which may include adaptive decoding for a command interface kind of an application or can be fed as an input to an advanced module to do the task of further language specific processing / understanding. This way we'll be able to come up with a better and more efficient multilingual system. A typical system block diagram is shown here in Figure 1-1.

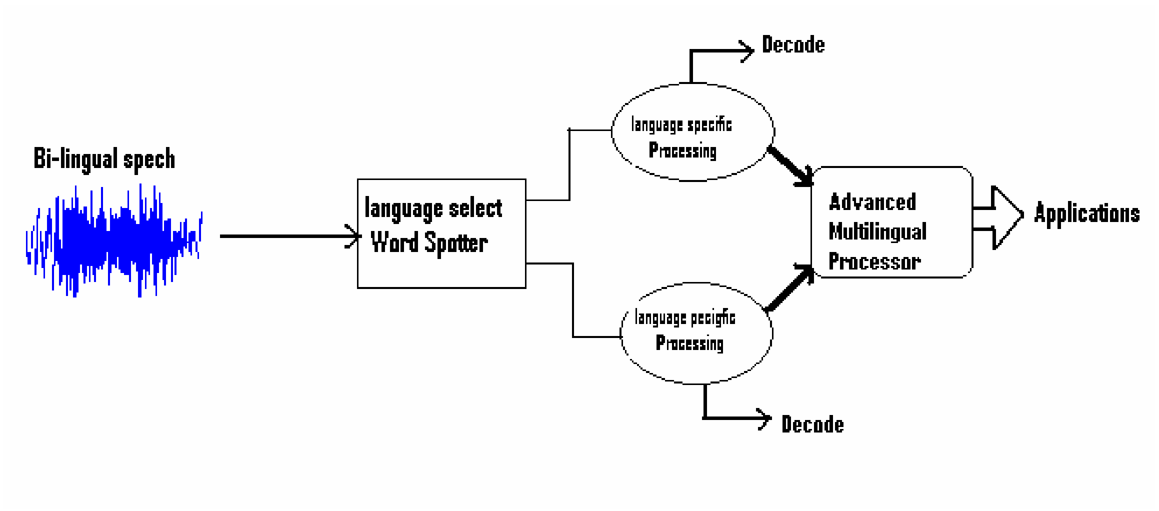
Bi-lingual spech
language select Word Spotter
language specific Processing
language pecigfic Processing
Decode
Decode
Advanced Multilingual Processor
Applications

Figure 1-1 A bi-lingual system using language specific word-spotting

Here the word spotter includes all the other essential speech recognizer modules like feature extractor, trained decoder etc. The system which is going to be discussed in this report in sections to follow, aims to implement the 'Language Specific' word-spotter of a multi-lingual system and as already told this part will play a major role in over-all performance of the whole system. We'll basically be concerned about a bi-lingual perspective which is more practical for several reasons.

1.2 A little bit about Bengali

Bengali is a widely spoken language in eastern part of India and Bangladesh. With nearly 200 million native speakers it is world's fourth most widely spoken language after Mandarin, Spanish& English. Bengali is of Indo-Aryan origin of south-Asia and comes as a successor of Sanskrit, Pali and Prakrit languages. Following nearly 200 years of British rule in India it is very obvious that there will be 'colonization effect' on the native spoken language. So, it is very likely in several parts of India that a speech spoken in native language might be 'contaminated' with English and vise versa. In case of Bengali (the language in its tong actually called 'Bangla') there's no exception. For all these reasons, spotting Bengali (Bangla) words (in fact, Key-words from a speech recognition perspective) in unconstrained English speech is of great interest to us. Such a system can further be developed to a 'Bengali-English' bi-lingual speech processor or understanding system which will have ample applications like information query systems in the Bengali native areas , say, eastern part of India.

1.3 Speech recognition and keyword spotting

A typical speech recognition system transcribes a spoken input into texts. Now, depending on the type of input they are classified in two major categories namely isolated word recognizer and continuous speech recognizer. For each of these cases a suitable acoustic model is chosen (either whole word based or sub word based) and they are trained using the features extracted from the training utterances. The decoder then takes an unknown utterance as input and finds out the best match or best match sequence form a set of trained models. The vocabulary of a recognizer decides how many words the recognizer can identify / transcribe. The task grammar of the recognizer will constrain the functional output of the recognizer to a limited set words / word sequences. It may also allow an out put to be of arbitrary length and unconstrained. Naturally these are more encountered in real world scenarios and will be much interest to us.

The problem of automatic speech recognition can actually be viewed from several perspectives such as acoustics, signal processing, pattern recognition, phonetics, linguistics, psychology, neuroscience, and computer science. ASR is a very difficult task. Although much advancement have been made in ASR research, the performance of today's best systems is more than an order of magnitude in error rate from human performance. The difficulties can best be described in terms of the tasks to be performed. For example, task might be to recognize isolated words to recognizing continuously spoken words or even more challenging spontaneously spoken language recognition. In all cases, the performance of the recognizer always depends on human 'cooperation' to large extent. The performance can also be severely affected by the environmental conditions. There are many other robustness issues happen to be there that still demands for more adequate methods to be solved.

Keyword spotting or simply word spotting is a special field of speech recognition that has got a significant focus over the past decade with the advent of speech recognition applications in real life. Although most of the Automatic Speech Recognition (ASR) systems are designed to recognize a limited set of utterances or some fixed set of word sequences it is quite unlikely that the users will always stick to those constraints. In a practical scenario it has been seen that in most cases user utters out of vocabulary words. So, we need to design systems that can recognize a limited set of vocabulary words or 'words of interest' or 'Keywords' embedded in extraneous speech (out of vocabulary words / background etc). Such a system or a keyword spotter is clearly different from a conventional ASR system. In keyword spotting we are not at all bothered about transcribing the whole of the input utterance. Only the keywords are of interest to us. They might occur any where in the utterance any number of times. Keyword spotter's job is to find out if a keyword was at all spoken or not and if yes then detect the correct keyword. Primary job here is to correctly figure out if the utterance if fully extraneous of there's keyword(s) embedded in it. Failing which the system might end up generating 'false alarms' or missing the keywords. Designers have to make a compromise between these two issues as we'll see later and these decide the performance characteristics of the keyword spotter.

In a typical keyword spotting system, the keywords are basically trained as isolated words (though exceptions are there) and the extraneous speech/background is represents by one or more models better knows as 'filler' or garbage models. The decoder finds out the best match of filler-keyword sequence during decoding. It is expected that if there happens to be a keyword then the optimum sequence generated by the decoder will contain the corresponding keyword model provided it has been 'properly' trained using a 'broad' class of spoken examples.

Modeling the filler is a very important issue. The keyword spotter's performance depends to a large extent on how the fillers/garbage is modeled and how 'well' they represent the extraneous speech without rejecting the correct keywords. There could be whole word based models for the filler [1] or it could be a phone based [6] (e.g. all phone network based keyword spotter). Decoding network also plays an important role in system performance. This will decide if the spotter allows the keyword to occur in the utterance without any constraints [1] or certain constraints to be imposed like one occurrence per utterance [11] or occurrence in the middle of the utterance etc. As expected the level of difficulty for spotting unconstrained occurrences are more than the constrained ones. A typical Keyword spotting system is show in Figure 1-2.

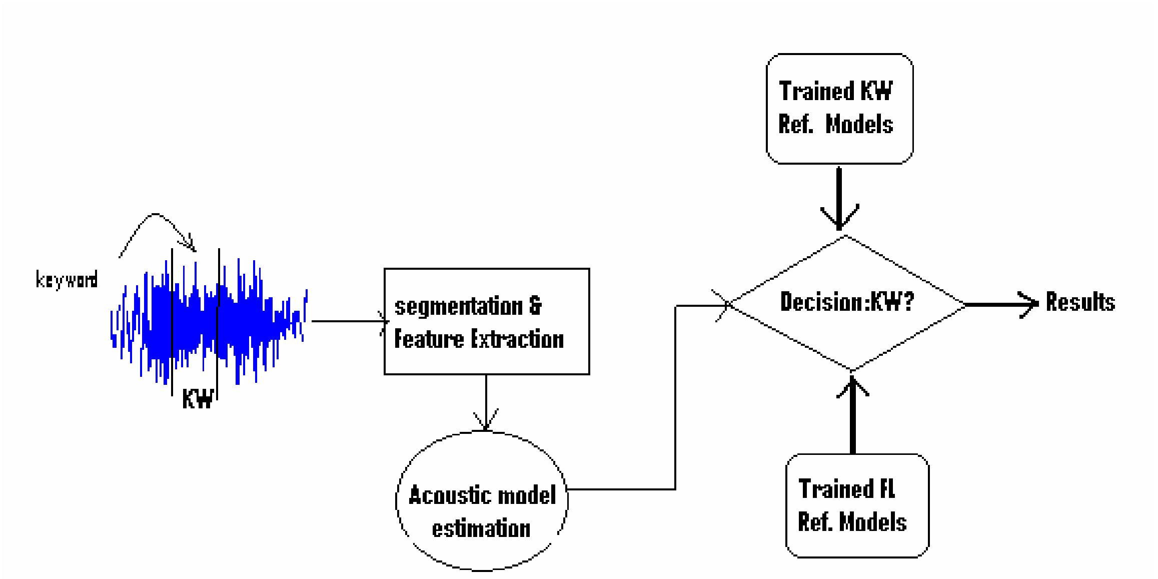

Figure 1-2 A typical Keyword spotting system

It must be noted that a conventional ASR system just transcribes the speech input and it has nothing to do with the contents or what they signify either. In other words it is NOT a speech understanding system. But a keyword spotting system, which has to make a 'decision' and pick out the so called 'words of interest' from an unconstrained speech can be considered as a 'step towards speech understanding' rather than just being a 'transcription machine'. Of course, it has the ability of information retrieval as well. In that way, it's a more intelligent version of a typical ASR.

1.4 Keyword spotting in recent years

The problem of Keyword spotting has been approached in several ways. Dynamic Time Warping (DTW) and Hidden Markov Model (HMM) [3] based algorithms have been mainly used. Some researchers have also tried ANN-HMM combined approach where neural networks work in the post-processing phase[9]. Bridle has approached the problem by introducing dynamic programming techniques for whole word templates [8]. Higgins and Wohlford introduced a continuous speech recognition approach to keyword spotting, and they also defined filler templates to represent the non-keyword portions [10]. Finally, Rose and Paul introduced HMM-based keyword spotting [1]. Similar approach was also followed by Wilpon & Rabiner [11]. In all cases the choice of garbage model considered to be a very important issue as it decides the systems performance in terms of false alarms and false rejections or deletions. Many cases, the system had to do a trade off between high detection rate and false alarms. Number of approaches has sought to achieve a high detection rate at the cost of high false alarms and then use further processing for 'removing the false alarms' or better known as 'rejection'. A number of post processing/ rejection strategies have been approached by the researchers. These include use of duration information; duration normalized acoustics score and a likelihood ratio scoring [1], use of classifiers like Gaussian or Neural networks and very recently discriminant methods like Support Vector Machines (SVM) [12].

Keyword spotting has been successfully used in many applications as of now. Such as, Cambridge University Video Mail Retrieval (VMR) project, topic identification systems e.g. topic identifier developed by K. Ohtsuki *et. al.* for Japanese broadcast news [5]. There are many others like say, Voice dialing [15] . Very recently, in 2004, Ai-Logix, US, has introduced a real time keyword spotting system named WordALERT™ .

1.5 Objective of this project

The goal of this project is to implement a 'speaker dependent' Bengali keyword spotting system in unconstrained English speech. Two major approaches will be considered. They basically differ in the type of filler models used and decoding network design. The trade off between false alarm rate and correct hits or percentage detected will be studied in both the cases. The project also aims to study the effect relative word insertion penalty, keyword-weights and language model scale factor on the system performance. An optimum operating point with high detection rate but with acceptable false alarms is to be sought. A possibility of combining the two approaches together for a better system performance will also be explored.

The speaker dependent keyword spotter will be HMM based. The keywords will be having whole word models trained as isolated words. On the other hand, fillers/garbage will be either whole word based or sub-word based depending on the approaches used. A study will be undertaken to find out whether there's any performance variation depending on the number of word based fillers used. The HMMs will be trained using both single Gaussian and then a GMM (Gaussian Mixture Density) based approach will be taken. The performance of mixture density based models will also be investigated.

The system will be built and evaluated on HTK v3.2.1 platform. An approach to make the system speaker independent will be sought. Finally, it will be discussed how the system can further be developed to a Bengali-English bi-lingual interface.

1.6 Report outline

The organization of the rest of the report is as follows:

In Chapter 2, some basics of HMM based keyword spotting will discussed with a little bit of mathematical fundamentals behind them. In Chapter 3, the system development including training, parameter selection will be explained.

In Chapter 4, the system is evaluated and analyzed followed by conclusions and future works together with further development of a bi-lingual interface will be discussed in Chapter 5.

Chapter 2

Keyword spotting using HMM

Spotting Keywords in unconstrained extraneous speech can be approached in several ways. Use of dynamic programming technique could be a promising one. Recently, iterative dynamic programming [13] has also been successfully used in keyword spotting. But still, NO method models the stochastic nature of speech signal as good as HMMs does. So far, HMM has been the most successful technique in several speech recognition areas and its application in keyword spotting is also very promising. To generate HMM based 'acoustics-stochastic' models from speech signal it is required to look into a statistical speech modeling perspective and then some fundamentals behind HMM modeling.

2.1 Stochastic model for speech

Speech is a very 'special' kind of signal. The vibrations in the vocal cord are transformed into a sequence of 'pressure waves' in the air media. The characteristics of these waves are basically controlled by the articulary system generating a particular phone /sound. Then electrical transducers (Microphone) map these vibration sequences to a varying amplitude electrical signal. Our rough perception about so called 'Speech Signal' is actually this mapped electrical signal which can be further digitalized to sequence of numbers for numerical/algorithmic processing like say, speech recognition. A typical speech signal is shown in Figure 2-1.

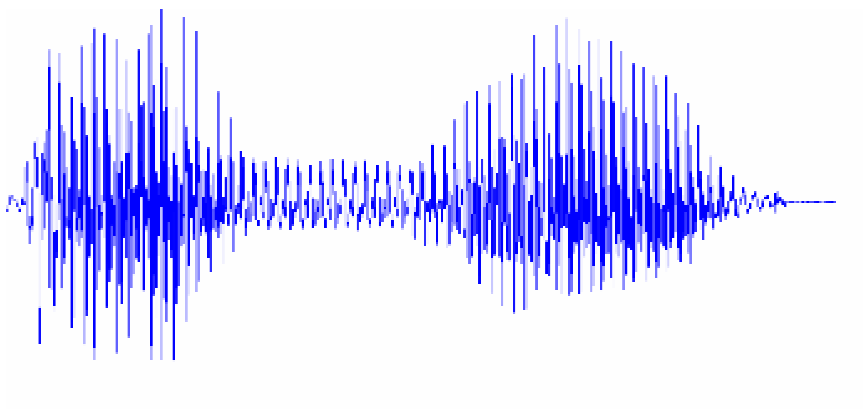

Figure 2-1 Waveform of the word 'Bangla'

2.1.1 Acoustic modeling

Perception of a segment of uttered speech like a phone or a whole word depends on several characteristics of speech. Simply time domain analysis can never reveal those. Actually, same word or phone uttered at different instances could have a very different time domain representation. There are several reasons for these including inter/intra speaker variations, articulary systems and environmental conditions. So, the identity of a word or a phone or a 'specific sound' is not directly embedded in time domain. Hence, we go to frequency domain analysis of speech which will give us a better and robust parameterization to discriminate different phones/words from each other. It's pretty obvious that for keyword spotting that's a very important issue.

The speech waveform is first digitalized using techniques like Pulse Code Modulation (PCM) with a sampling frequency of 8 kHz or more. After the speech signal has been mapped to a sequence of numbers, different Digital Signal Processing (DSP) techniques like FFT etc are used for its parametric representation. This is also called feature extraction. The output is speech signal represented in feature space (Frequency domain) and such 'coded' signals are further utilized for speech recognition applications.

The parameters are dynamic. They change with time. But each set of parameters are used to describe a limited period of time following the assumption that speech signal is stationary for at least 10 ms time interval. So, normally, speech is windowed (typically of 250 ms duration) and corresponding to each window the parameters are calculated. Keeping in mind about the 'short time stationarity' of speech the time difference between successive windows is typically set at 10 ms. This is better known as 'short time speech analysis' and results in a Spectrogram when DFT is used as feature. A speech spectrogram is shown in Figure 2-2.

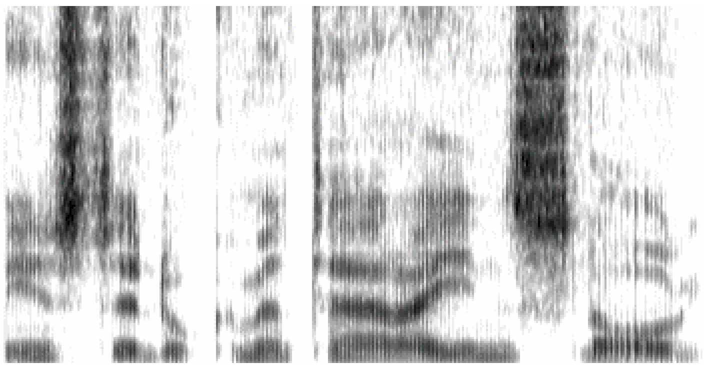
Figure 2-2 Speech Spectrogram

Spectrograms give a better representation of speech than time domain. The dark bands that can be seen are known as formants. Their variation with time and spacing has the information about a specific phone embedded in the speech. But still, it is not enough for advanced automatic speech recognition techniques.

State of the art Automatic Speech Recognition (ASR) methods use parametric representations for speech such as smoothed spectra or Linear Prediction Coefficients (LPC) plus various representations derived from these, like cepstral and log cepstral coefficients . Still it is a challenge for the researchers to find out an even better set of parameters that captures more of the speech characteristics.

From LPC coefficients we can derive cepstral coefficients, the coefficients of Fourier Transform representation of the log magnitude spectrum. They are found to give a better representation of the local spectral properties of the signal for a given analysis frame and hence they are more robust. Besides, their temporal derivatives like Delta, Delta-delta (acceleration) coefficients are shown to improve recognition accuracy. They also have a closer 'mapping' to the articulary features like say, variation of vocal track / vocal tract transfer function during an utterance.

To mimic human auditory system we generally do a Filter-bank analysis of speech before parameterization. This take care of critical band response of human ear (frequency scale corresponding to that is call Mel scale). A typical filter bank may look like this.

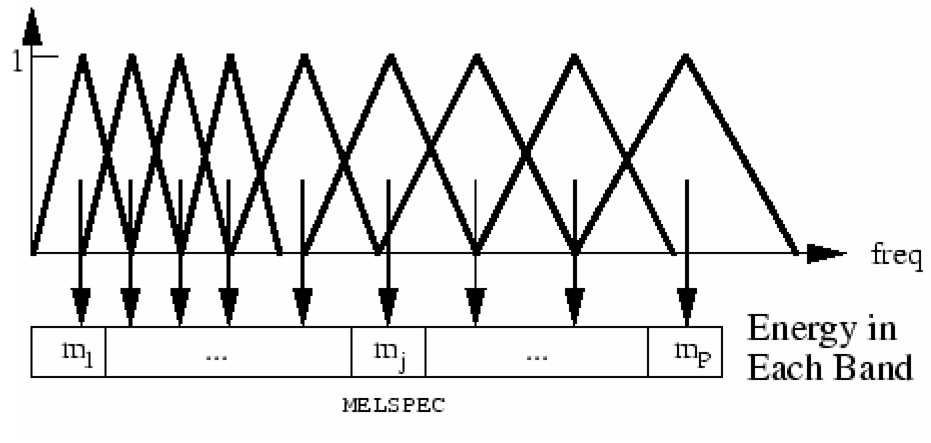

Figure 2-3 Mel scale Filter-bank

The cepstral coefficients thus obtained are known as Mel Frequency Cepstral Coefficient (MFCC). These along with their temporal derivatives (delta and delta-delta / acceleration) or MFCC_0_D_A will be used in the HMM based KWS system design to follow. The block diagram view of a typical feature extraction module of a speech recognizer has been show in figure 2-4.

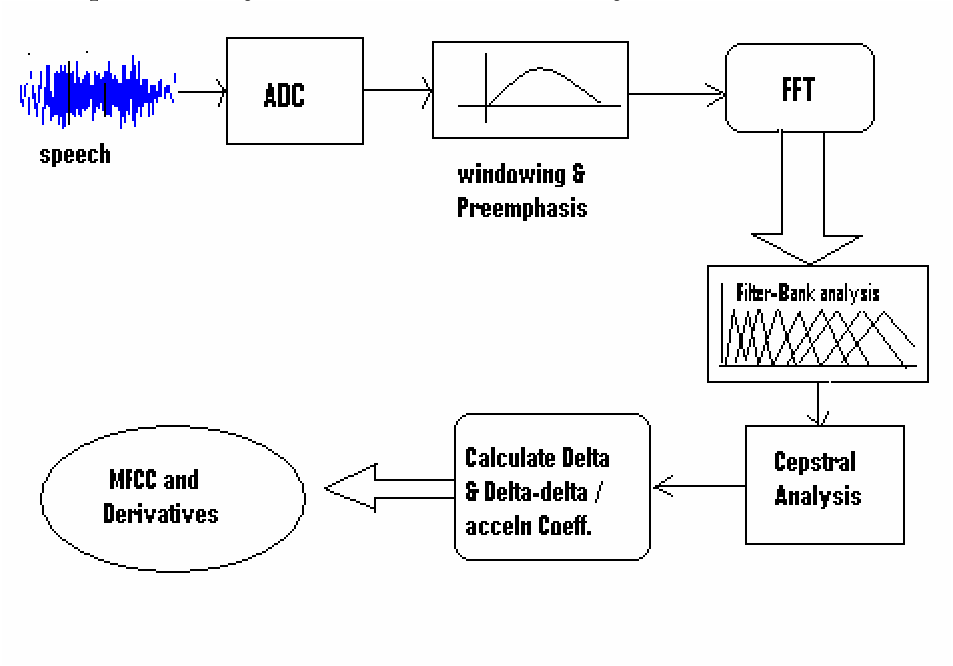

Figure 2-4 Feature extraction modules

2.1.2 Hidden Markov Models

With acoustic modeling we are able to represent speech as a sequence of feature vectors. Such a sequence optimized for a given segment of speech (word or sub-word) from a number of training utterances can definitely be used as a reference model for recognition of that particular segment. We can use Euclidian distance in N-dimensional feature space for matching or better known as template matching. For duration normalization a Dynamic Time Warping (DTW) algorithm can be used. But this kind deterministic template based method fails to capture the stochastic behavior of speech. So, we look for models that incorporate the stochastic nature of speech. We can say that they are sort of a 'Statistical Templates' which are optimized for N-dimensional 'random' feature vector in the corresponding vector space. Hidden Markov Models or HMM exactly does that and attempts to model stochastic behavior of speech [14].

Suppose, we have a feature vector sequence represented by '***O***',

$$\boldsymbol{O} = \boldsymbol{o}_1, \boldsymbol{o}_2, \ldots, \boldsymbol{o}_T \tag{1}$$

Where $\boldsymbol{o}_t$ is the speech vector observed at time t. We need to find a word w_i (model optimized corresponding to the ith vocabulary word) in such a way that the probability of **O** being 'observed' is maximized.

$$\arg\max_i \{P(w_i|\boldsymbol{O})\} \tag{2}$$

Following Bayes' rule, we have

$$P(w_i|\boldsymbol{O}) = \frac{P(\boldsymbol{O}|w_i)P(w_i)}{P(\boldsymbol{O})} \tag{3}$$

From the above expression it is clear that for a given set of prior probabilities $P(w_i)$, the most probable spoken word depends only on the likelihood $P(\boldsymbol{O}|w_i)$.

Usually, '***O***' is of many dimensions. So, direct estimation of the conditional probability $P(\boldsymbol{o}_1, \boldsymbol{o}_2, \ldots |w_i)$ is not feasible. Instead a parametric model for speech (say, word) production is assumed. And there comes the Markov Model. Due to introduction of such a model estimating the class conditional observation densities $P(\boldsymbol{O}|w_i)$ is replaced by the much simpler problem of estimating the Markov model parameters. In HMM based speech recognition, it is assumed that the set of observation vectors (temporal feature vectors) are generated by a Markov model as shown in Figure 2-5.

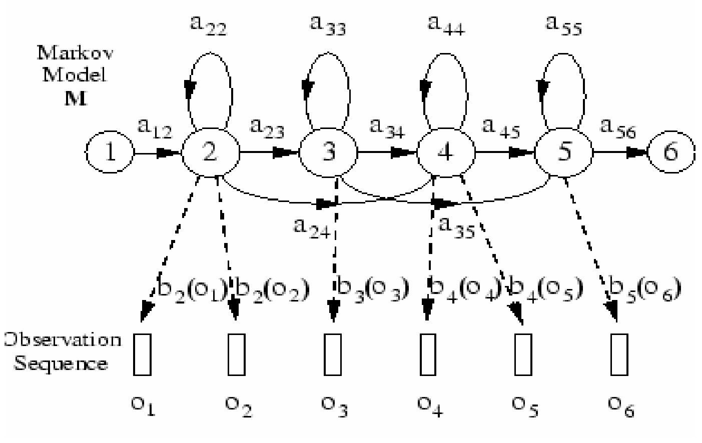

Figure 2-5 Markov generation model

A Markov model is a finite state machine which changes state once every time unit and each time t that a state j is entered; a speech vector $\boldsymbol{o}_t$ is generated from the probability density $b_j(\boldsymbol{o}_t)$ which happens to be a single or mixture Gaussian density. Furthermore, the transition from state i to state j is also probabilistic and is governed by the discrete probability a_{ij}. The joint probability that ***O*** is generated by the model ***M*** moving through the state sequence ***X*** is calculated simply as the product of the transition probabilities and the output probabilities. So for a state sequence ***X,***

$$P(\boldsymbol{O}, X|M) = a_{12}b_2(\boldsymbol{o}_1)a_{22}b_2(\boldsymbol{o}_2)a_{23}b_3(\boldsymbol{o}_3)\ldots \tag{4}$$

However, in practice, only the observation sequence ***O*** is known and the underlying state sequence ***X*** is hidden. This is why it is called a *Hidden Markov Model* (HMM).

Given that X is unknown, the required likelihood is computed by summing over all possible state sequences, $X = x(1), x(2), x(3), \ldots, x(T)$, this follows that,

$$P(\boldsymbol{O}|M) = \sum_X a_{x(0)x(1)} \prod_{t=1}^{T} b_{x(t)}(\boldsymbol{o}_t)a_{x(t)x(t+1)} \tag{5}$$

Where $x(0)$ is constrained to be the model entry state and $x(T+1)$ is constrained to be the model exit state. Equation (5) is very computation intensive. No of computations required is of the order of ~ 2T. N^T, where N is number of states in the model. Considering that for a word T=100 => 2.100.5^100 or 10^72 computations are required and so, it'll be a definite hazard for real-time implementations. Hence an alternative approach is taken considering the fact that the contribution of the 'Most Likely' state sequence to the summation is maximum. Or, instead of calculating the sum for all possible state sequence we can just compute it for the most likely state sequence and that will given us a very good approximation of the likelihood probability. Or,

$$\hat{P}(\boldsymbol{O}|M) = \max_X \left\{ a_{x(0)x(1)} \prod_{t=1}^{T} b_{x(t)}(\boldsymbol{o}_t)a_{x(t)x(t+1)} \right\} \tag{6}$$

Given a set of models M_i corresponding to words w_i , we can solve HMM recognition problem by equation (3) and assuming that

$$P(\boldsymbol{O}|w_i) = P(\boldsymbol{O}|M_i). \tag{7}$$

All these analysis assumes that the parameters $\{a_{ij}\}$ and $\{b_j(o_t)\}$ are known. They can indeed be estimated from a sufficient number of representatives (or training utterances) of each word(corresponding model M_i) using Baum-Welch forward-backward re-estimation process which works on statistical Expectation Maximization (EM) algorithm. During recognition process or to be precise , while calculating the likelihood probability (called likelihood score) of an unknown utterance , the most likely state sequence is optimized by Viterbi Algorithm which traces out the most optimal through the decoding lattice comprising of observation vector sequence and HMM states.

HMM has been so far the most successful method in several areas of speech recognition with considerably high accuracy. Pertaining to the fact that speech recognition happens to be a 'pattern recognition' problem , HMM can be efficiently used in many other patter recognition areas. They have actually been successfully used in written script identification. Next section will discuss keyword spotting using HMM.

2.2.1 Keyword spotting

In keyword spotting it is required to recognize a given set of keywords in extraneous unconstrained speech. This problem was initially addressed by the researchers [8] using dynamic programming techniques. As told earlier still today some researchers find them useful. In a method, called sliding window technique, the word template is sort of slided along the whole utterance to find a possible match at any instance of the utterance. Several pruning methods have also been implemented for complexity reduction. In any case, the most basic template matching based algorithms in the way show below.

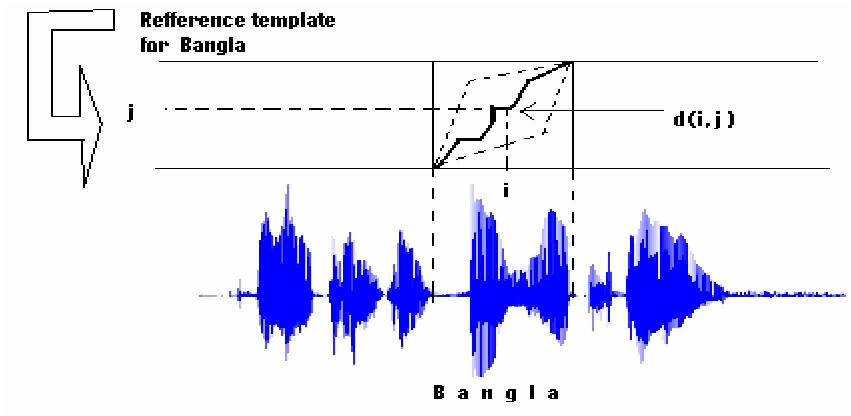

Figure 2-6 Template matching based KWS

As shown in Figure 2-6 the path P(i , j) gives the optimal distance between the reference template and the actually uttered segment. Here distance means cumulative sum of Euclidian distances between the corresponding vectors in successive frames. A threshold can be put to decide between a putative keyword hit and extraneous speech. In spite of all these, these methods can't handle the stochastic properties and randomness of different instances of the same utterance. This is a major disadvantage.

HMM based keyword spotting system of course uses reference templates but they are not fixed templates like those of template matching method. They are basically Gaussian HMMs optimized for each keyword and a class of fillers. So, they are more like a 'statistical template' rather than being a fixed one. In that way they are dynamic. Hence, it's better to call them 'Reference models' and NOT 'reference templates'.

Typical HMM based keyword spotter will have a set of fillers trained by extraneous speech for its representation. They could be whole word models [1] or sub-word based like phonemes [6]. The keywords are generally trained as isolated words as context dependency has shown to have NO significant performance improvement. Different grammar networks can be used for the decoding purpose. This depends on the task of the word spotter. Some cases it is assumed that there will be only one keyword per utterance [11] and grammar network compels the optimal path to traverse through keyword networks. Such a network is shown below.

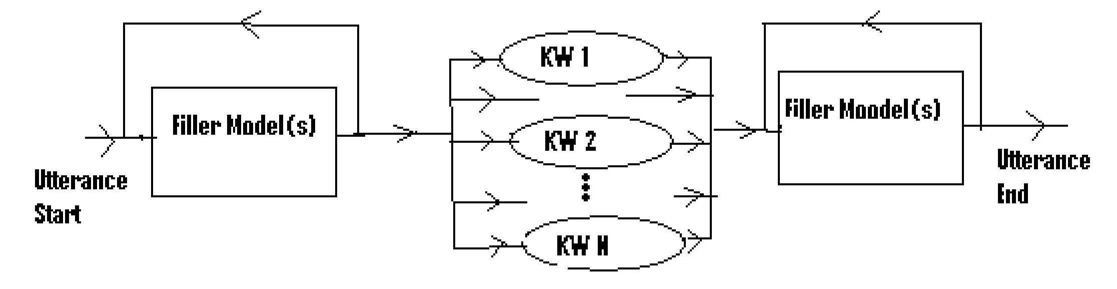

Figure 2-7 Networks for one KW per utterance

Decoding generates a sequence of filler(s) followed by a keyword and then another sequence of filler(s). A specific keyword is chosen depending on best likelihood score of the whole sequence.

But in a practical scenario it can't be said that there will be exactly one occurrence of keyword per utterance. It can be any keyword, any number of times and any where in the utterance. Hence, an unconstrained grammar network is required. Overviews of two different kinds of unconstrained grammar networks are discussed. Those are the ones used in the system design. The first one [1] looks like this,

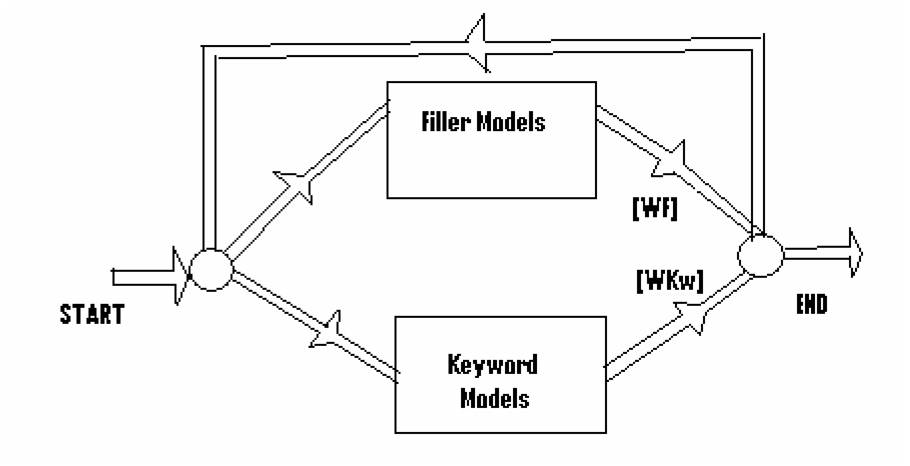

Figure 2-8 Unconstrained grammar networks for KWS

The grammar network above will allow any sequence of keywords and fillers and finds out the best matched sequence giving the highest score given an unknown/unlabelled utterance. So, this is more practical. But it can also happen that an utterance having keywords gets matched to a sequence of fillers only as the optimal path is not forced to the sun-network of keywords. However, weights **[WKw]** can be used to boost the likelihood of key words relative to the fillers who is relatively less weighted with **[WF].** But then the chance of false alarms also increases. Several post processing rejection (reject a segment as extraneous part instead of declaring it as keyword) techniques can be used to remove false alarms. Nevertheless, there has to be a trade-off as per system requirements.

The 'filler block' can be having one or more whole word models. A modification of this filler sub-network leads to another unconstrained network as shown Figure 2-9. Any utterance of a particular language can be represented as a sequence of phonemes of that language. So, an all phone network, which allows any arbitrary sequence of phonemes, can be used for recognizing ANY utterance. Hence it'll serve the purpose of filler pretty well. But it will also tend to represent the keywords as well in case they are from the same language or the keywords can have a representation in terms of phonemes in the all phone network. Whole word based keyword models can be used to compete with the all phone network, provided they are properly trained from a wide variety of training utterances. In case, the keywords are of different and contains some language specific 'special phonemes' then it's even better in terms of system performance for obvious reasons. Still there will always be a clash between keyword models and all phone networks. Hence, trade-off between hits & false alarms has to be done by adjusting the words transition penalty which will be discussed in detail while explaining the system design.

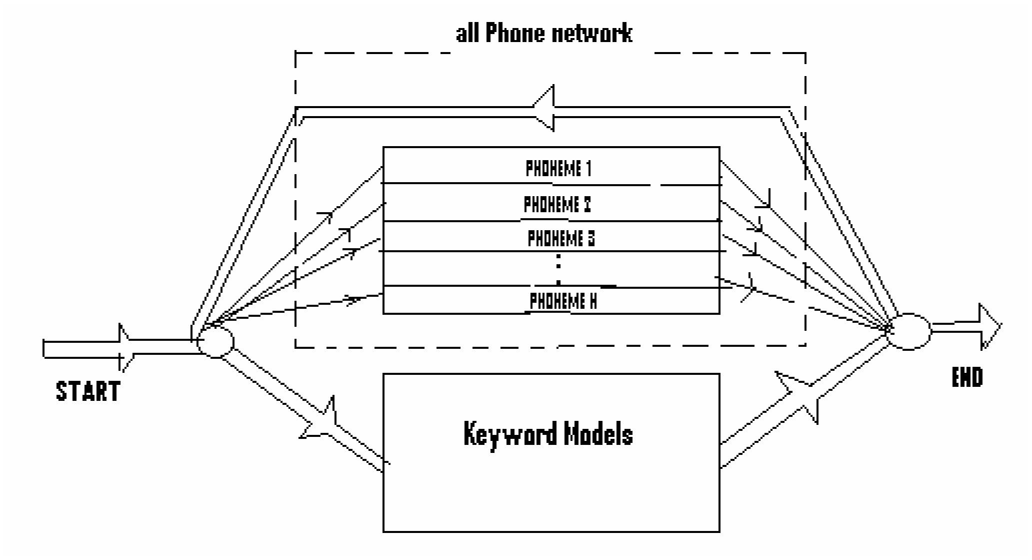

Figure 2-9 All-phone network based filler

Apart from all the approaches discussed so far there are several others. Using a large vocabulary recognizer as a keyword spotter has been very popular. Instead of having limited number of filler models to represent the whole extraneous speech very precise models for non-keywords are trained. It's like every non-keyword is having its own model like every keyword. So, it's more like a large vocabulary (similar to a dictation machine) recognizer which generates text transcription of spoken utterance and then text reference of keywords are used to check for a putative hit. This way, the requirement for a word spotting specific network is avoided. But building a large vocabulary recognizer is always a difficult job and it's a major disadvantage.

2.2.2 Language model in keyword-spotting

For a constrained grammar network the recognizer out put will be confined to a limited set of possible utterances. But NOT all of them are supposed to be meaningful obeying the grammatical rules of a particular language. So, to boost the likelihood of a possible meaningful utterance (assuming that the actual utterance was 'meaningful' i.e. it was grammatically correct) we use language model. It's sort of prior information that backs up the decoding process so that the recognizer does not end up with an unlikely meaningless utterance.

$$P(w_i|\boldsymbol{O}) = \frac{P(\boldsymbol{O}|w_i)P(w_i)}{P(\boldsymbol{O})} \tag{1}$$

In equation (1) the term $P(w_i)$ is the likelihood of occurrence of a particular word. As far as grammatical rules are concerned, this probability should depend on N-words that occurred preceding the word. So, it's essentially a conditional probability. When N=0 then it is called unigram language model. The probability depends solely on the word itself. There could be bi-gram or trigram or n-gram models depending on N=1 or N=2 or N= n-1. They are estimated from their occurrences in a large corpus / daily newspapers etc.

But, since keyword spotting uses an unconstrained grammar network or better to say, a grammar-free network so, use of language model does not make sense. Also, keyword spotting system is not at all bothered by grammatical correctness of the utterance. Its job is just to detect the keywords, be it embedded in a grammatically incorrect meaningless utterance. So, language model is NOT directly used in keyword spotting. However, unigram language models can be assigned manually to implement keyword weighting in grammar/decoding network and will be discussed in detail in next chapter.

2.2.3 Performance measure for word-spotting

Performance analysis of a keyword spotter is different from a typical ASR. It is expressed in terms of hits, substitutions, misses/deletions, false-alarms and trade-off between false alarms (FAs) and hits or percentage detected.

Hit means a specific keyword being spotted correctly. It is expressed in terms of percentage of keywords correctly detected. Substitution means one keyword is mistaken as the other. Misses correspond to instances when the spotter fails to detect keyword(s) embedded in the utterance. Reverse case is FAs(false alarms) which implies that in spite of no occurrence of keyword in actual utterance the spotter declares occurrence of keyword(s). FAs are usually expressed as number of false alarms per keyword per hour of speech input (FA/KW/H) or simply FA/KWH.

By adjusting the relative word insertion penalty and keyword weights (to be discussed soon) a tradeoff can be played between FAs and hits. A performance curve, called **ROC** (Receiver Operating Characteristic) can be drawn to reflect this tradeoff. This curve is a plot of percentage hit vs. FA/KWH and depicts the performance characteristics of the system very well.

Chapter 3

System design for Bengali keyword spotter

This chapter will discuss the development of speaker dependent Bengali keyword spotter in unconstrained English speech. Vocabulary size of the spotter is 12. Two different approaches were used and both will be explained in detail. Discussion will include training/testing database(s), type of filler/keyword models used, variation of grammar networks and selecting parameters like relative word insertion penalty, keyword weights and language model scale factor for an optimal system performance.

3.1 Speech Data

Training/test data is very important for designing any speech recognition system. The system to be developed here is a speaker dependent one. This essentially means that the keywords (Bengali) are assumed to be uttered by a single speaker. Also, isolated keywords uttered by him can be artificially put in the non-keyword utterances of other speakers and those 'hybrid' utterances can also be used for testing the word spotter. Otherwise, the whole utterance could be uttered by the specific speaker. So, extraneous speech part can be 'speaker independent'. This will be kept in mind while choosing training/testing data. In all cases speech data was in windows '.wav' format and TIMIT utterances were used after converting them to windows '.wav' format from NIST format by using a software call 'ch_wave'. Sampling frequency was 16 kHz and there were 16 bits per sample.

3.1.1 Training data for keywords

Keywords are Bengali words. They are to be trained for a specific speaker. Hence training data used for them were isolated utterances of the keywords. All utterances were essentially from the particular speaker. For each of the twelve (12) Bengali keywords, 25 isolated utterances were used for training. So, total size of the keyword training data base was 300. The 12 Bengali keywords were as follows: **Ashram, Assam, Bangla, Bazzar, Bramhaputra, Guwahati, Himalaya, Jungle, Kolkata, Paajama, and Sristi & Yoga.**

3.1.2 Training data for fillers

Extraneous speech for the system was English. Fillers, representing English, were whole word based models for one approach and phoneme based for the other. For the first case the fillers were trained with more than 350 random utterances in English which include chosen from TIMIT database and utterances from the specific speaker as well. On the other hand, the phoneme based filler model had the whole of TIMIT database as its training data.

3.1.3 Test data

Total number of test utterances was 240. These included utterances from the specific speaker and also from TIMIT test database. They had one, multiple or no occurrence of Bengali keywords. 120 utterances had one or more Bengali keywords while rest of them had no such keywords. That means, chances that a given utterance will contain a Bengali word from the vocabulary is 50%. Occurrence of keywords instance was unconstrained for a given utterance. For TIMIT utterances, Bengali keyword was artificially 'injected' at a random instance. This was done in two steps. First, the isolated utterances of keywords were processed with Wavesurfer and then a MATLAB code was used to put the keyword at a random instance of the utterance. Thus sort of hybrid test utterances were generated. Reason for doing this was to make sure although the keywords were speaker dependent, the extraneous speech could be from any arbitrary speaker.

3.2 Choosing Keyword & Filler models

This is one of the most crucial phases of system design. As already told, there were two different design approaches. In one approach, whole word based fillers were used while phoneme based in the other. However, keywords had whole word models in both cases. Next section will discuss how prototypes HMMs were chosen for keywords and fillers.

3.2.1 Keyword models

For all 12 Bengali keywords identical prototype HMMs were chosen. It might be more reasonable that each word should have its own unique model. But, there were NO sub word level information available for Bengali words and hence prototype HMMs, each of 15 states were chosen. Also, when vocabulary size increases it won't be feasible any way to choose unique prototypes for each individual word. In any case, some 'common' prototype has to be chosen. But, here, pertaining to the fact that vocabulary size was only 12, so, for better system performance word-specific HMMs could be chosen. At least, for the 'classical Bengali words' like 'Bramhaputra' which is of longest duration and supposed to be comprising of many sub-words/phones. It is very important that an optimal model is chosen for such words. Number of states should be chosen after carefully studying the sub-word level structure. Otherwise, as will be seen in evaluation section, words like Bramhaputra gets matched 'too often' to several segments of utterances as the relative word insertion penalty is increased to higher values. In those cases, increment in relative word insertion penalty might NOT lead to increased false alarms and NO increment in hits as well. Most the utterances will tend to get matched to the word 'Bramhaputra' as it's longest in duration and hence sequences containing this word will have lesser penalty and a better score. Significance of all these things is that if possible, it's a very good approach to choose a 'proper' model for each word and that will certainly improve system performance.

Number of streams for each model was one. Density corresponding to each state was chosen to be single Gaussian initially and then gradually numbers of mixtures were increased to 2, 4, 8, and 16. Mixture densities give a better approximation of the original distribution. It's sort of similar to the fact that neural networks give a universal approximation to any function. So, GMMs (Gaussian Mixture Models) are supposed to given a better hit rate and a better system performance.

Mel Frequency Cepstral Coefficients (MFCC) and their temporal derivatives were used as acoustics parameters/features. In HTK configuration file it was set as MFCC_0_D_A. Number of cepstral coefficient s were set to be 12. Hence each observation vector was of dimension 39, with basic cepstral coefficients, the zero-th cepstral coefficient followed by delta and delta-delta coefficients. The observation/feature vector structure shown in Figure 3-1.

While feature extraction, window size was set at 250 ms and frame rate was chosen to be 10 ms making sure that speech is stationary with in that interval as we'll be assigning fixed distribution parameters (like mean & variance) corresponding to each frame and rule of probability distribution requires that the distribution 'has to be' stationary for doing that.

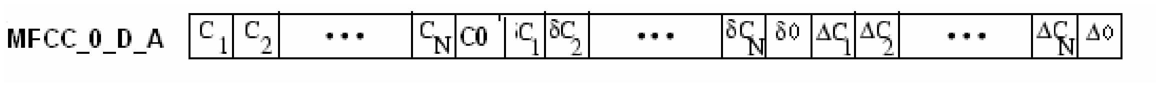

Figure 3-1 Feature vector structure

3.2.2 Filler models

Word based filler models were exactly similar to the standard keyword models used. However it is important to decide how many filler models should be used for training and what is the optimal number of fillers is that gives the best representation of the extraneous speech i.e. English. It is very important to note that too much precise filler models i.e. more number of fillers might degrade the system performance. Because as the number of filler increases the distances among keywords and fillers decreases in feature vector space. Hence, chances of confusing fillers with keywords or vise versa become very high.

The other approach uses HMMs with three states to represent an English phoneme. Though the number of context independent basic phonemes in English is 41 but a total of 61 phoneme-based models were used for representing the extraneous speech using an all-phone network to be discussed shortly. Extra 20 phonemes were used to incorporate a little bit of context dependency (but not like that of tied state triphones) although it has been found that context dependency hardly improves keyword spotter's performance. It's obvious. Because, word spotter has nothing to do with what is the exact transcription of the filler, rather it just has to differentiate between filler and keywords given an utterance. Here, of course, keywords have to be distinctly identified or let's say transcribed. Hence, in case, there's a phoneme based model for keywords rather than a word model then it's going to be a good idea to incorporate context dependent phonemes (context is 'in which word they occur') in keywords as was done in Video Mail Retrieval Project (VMR) of Cambridge University [7]. But for fillers which are never going to be precise any way, context dependency does not seem to make sense, at least for a simple case.

However, in a more complicated scenario it might so happen that there could be close decision between a sequence of phonemes (fillers) and a whole word. In those cases even 'a little bit' of context dependency might well help to discriminate between them as context dependent models are supposed to be more precise. The extra phoneme-based models included a silence model (labeled as h#) as well. Others were axr, epi, pcl, gcl … etc. Acoustic features or HMM parameters like mixtures/steams etc used were exactly same as keyword models. Here one point is to be noted that it is not very evident, how more number of mixtures are going to help the filler models as they are not precise in themselves. Means, the distribution itself is not going to be precise unlike say, the case of keywords which has some sort of precise representation. So, use of mixtures might better optimize the model for the keywords but we can say 'anything' like that about filler models. In any case, introduction of more and more mixture components without insufficient data might lead to 'over fitting' and might degrade the performance instead of improving!

3.3 Training

This is the most important step in the word-spotter design. All the system performance characteristics will depend on how well this phase is carried out. All the trainings were performed under HTK v 3.2.1 platform. Following section will have detailed discuss on how the keywords and filler HMMs were trained.

3.3.1 Training of word based keywords and fillers

Before training could be done all the speech '.wav' data files were converted to sequence feature vectors in MFCC files '.mfc' with appropriate configuration parameters. It was done using HTK tool 'HCopy'. Then having set appropriate prototype, HMM tool 'HCompV' was used for parameter initialization of the model. In this concern it is very important to state that the filler and keywords here were trained together. That means the global mean and variance calculated by HCompV included both keywords and fillers. Those parameters were used for initial estimate of the keyword models as well and it is known that the initial parameter values influences the ultimate models that come after several iteration of Baum-Welch algorithm. So, instead of training the keywords and filler together, they could be trained separately and that is expected to have given a better initial and hence final estimate of the keyword model parameters. In that way the keyword models could be more 'orthogonal' to the filler models which are good for a better system performance. During system evaluation it will be investigated what is the affect of separate is training for keywords and fillers or if there is any affect at all on system performance.

Keyword trainings were similar to isolated words. The training files corresponding to keywords had their silence part removed. While the non-keyword training files were used to train 'one' or more (Four) 'concatenated' filler models. NO silence model was trained explicitly as it was assumed to have been included in filler part. There were two types of fillers generated. In one case each training file was used to train a concatenation four fillers. While, all the non-keyword utterances were used to train 'single filler' in the other case. A diagrammatic view is given below.

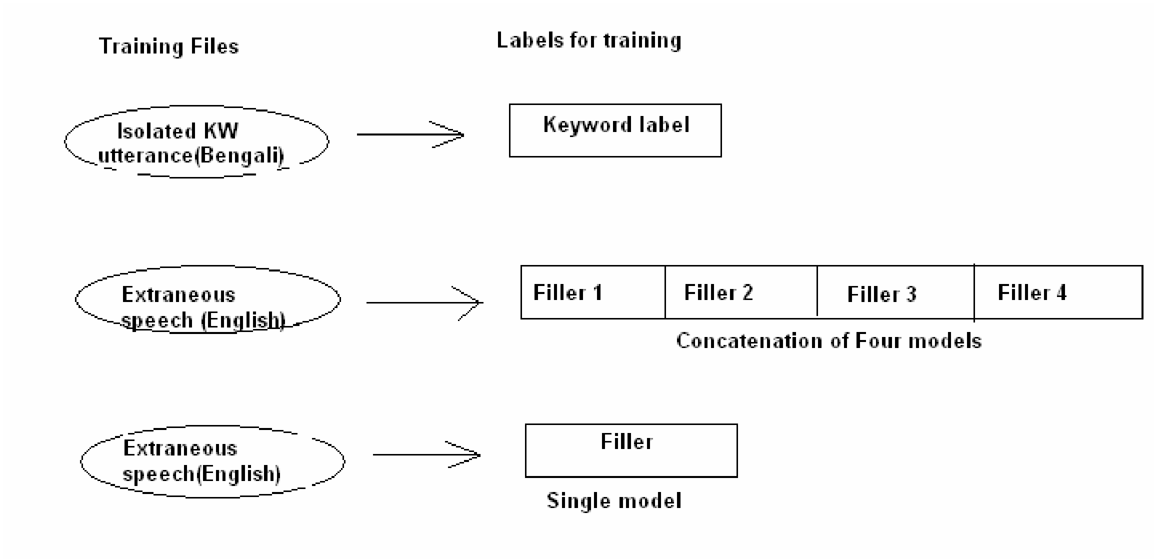
Training Files
Labels for training
Isolated KW utterance(Bengali)
Keyword label
Extraneous speech (English)
Filler 1
Filler 2
Filler 3
Filler 4
Concatenation of Four models
Extraneous speech(English)
Filler
Single model

Figure 3-2 Training file labeling

HTK tool 'HERest' was use to do embedded training an implement Baum-Welch algorithm. First seven iterations had single Gaussian densities. Then script editor HHEd was used to increase the mixture components to two. Then after 12th iteration the mixtures components were increased to 4 and the to 8 after 16th iteration. Finally, from 20th iteration onwards, 16 mixture components were introduced and the GMMs were further trained up to 23rd iteration. During all these process the pruning threshold was set as 250.0 150.0 1000.0.

3.3.2 Training the English phonemes

All-phone network based approach uses English phonemes to represent the extraneous English speech. All these phonemes were trained using the whole TIMIT database and its phoneme label transcriptions. The HMMs were trained in a similar way like that of the whole word models. A MATLAB program automatically trained all the 61 phonemes (including the extra ones) by accessing the whole TIMIT database and its phoneme level transcription files. A silence model was also created in the process although it's never used explicitly in grammar network and was included in the all-phone network like other phonemes. So, here also, it's sort part of the filler model(s). It is important to remember that what has been done is similar to using 61 filler models and it was made sure that they were able to represent the extraneous speech, which is a whole language (English). As phonemes are the basic building blocks of a particular language, so, using phonemes as fillers (where, filler is the language itself) with the help of an all-phone network (allowing any sequence of phonemes) does make sense! The phoneme training phase has been diagrammatically shown in Figure 3-3. Keywords trained in the previous case could well be used with these phonemes/all-phone networks.

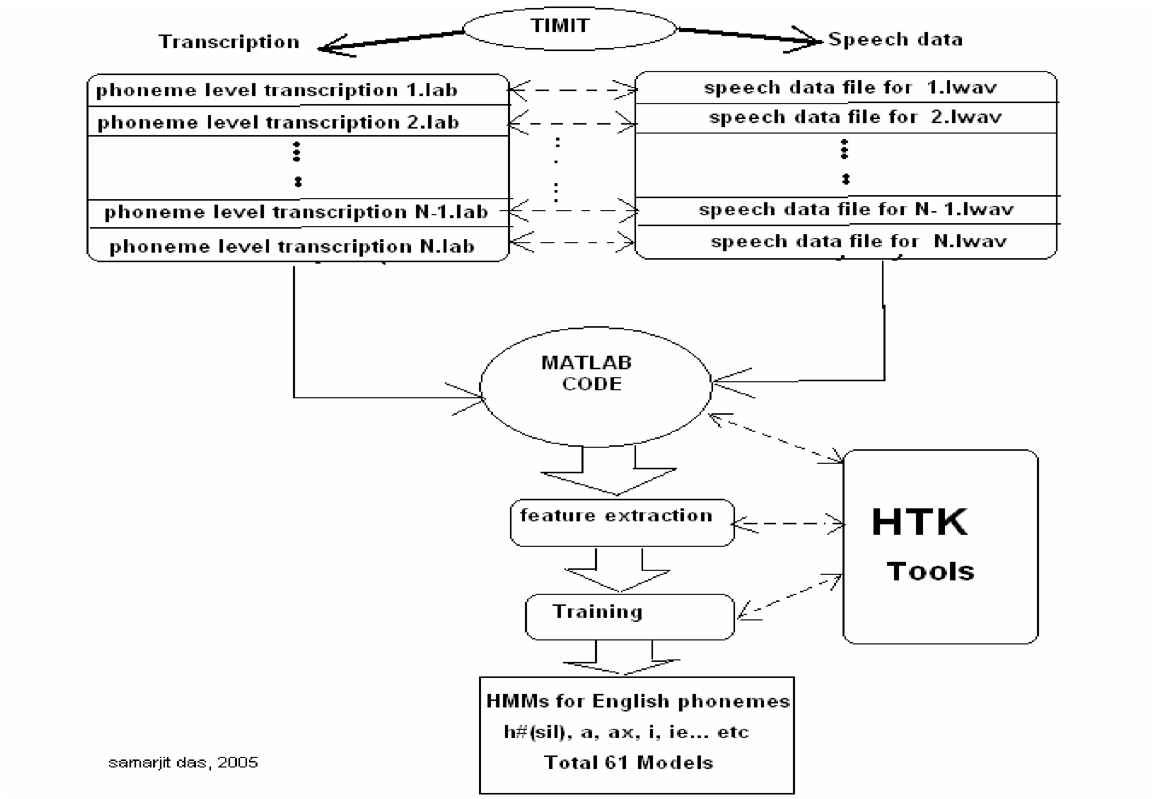
TIMIT
Transcription
Speech data
phoneme level transcription 1.lab
phoneme level transcription 2.lab
phoneme level transcription N-1.lab
phoneme level transcription N.lab
speech data file for 1.lwav
speech data file for 2.lwav
speech data file for N- 1.lwav
speech data file for N.lwav
MATLAB CODE
feature extraction
Training
HTK Tools
HMMs for English phonemes
h#(sil), a, ax, i, ie... etc
Total 61 Models
samarjit das, 2005

Figure 3-3 Training the English phonemes with TIMIT

3.4 Grammar networks

Grammar network is the key factor in differentiating the two approaches used. Both types of networks will be discussed in detail with appropriate trained models used. They are unconstrained grammar networks or in other grammar free networks. This is pertinent to the fact that system should be able to spot Bengali keywords in 'unconstrained' English speech. Also, there's no constraint on the occurrence of Bengali words as well.

3.4.1 Approach 1

This utilizes whole word based keywords and fillers. Both one and four filler models were used separately (M=1 or 4).The keywords were used parallel to the fillers and they were allowed to generate any possible sequence among themselves as per the best match with the test utterance. Structure of this network is shown in Figure 3-4.

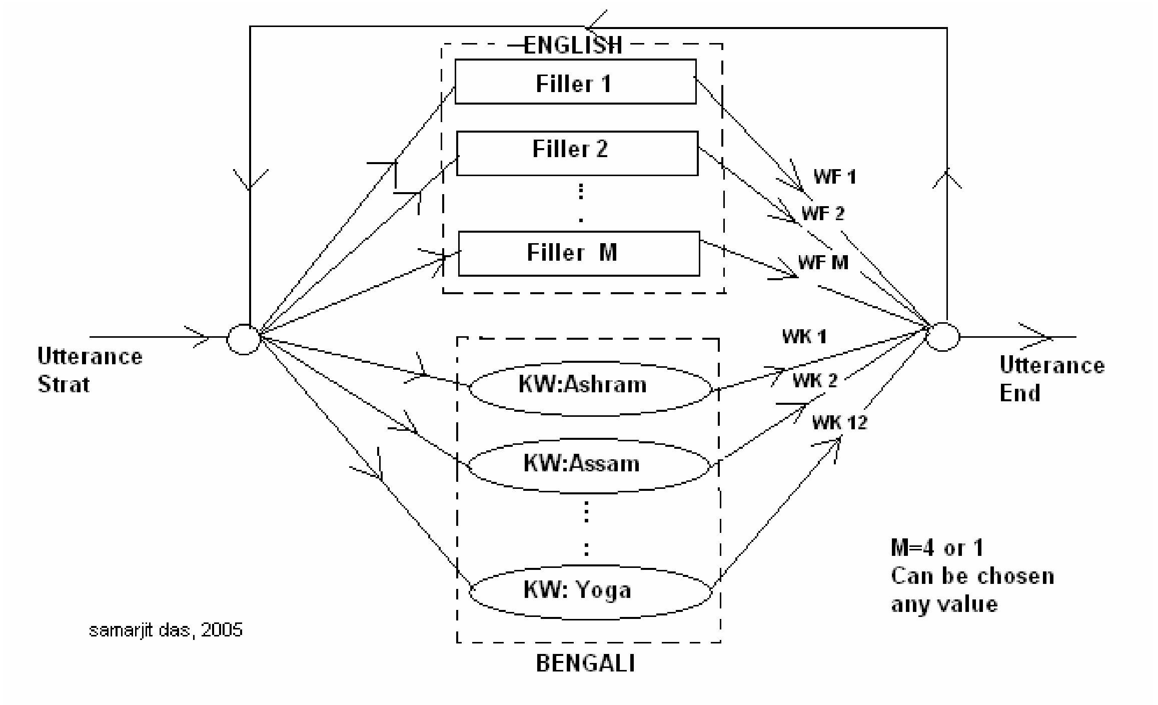

Figure 3-4 Grammar network using whole word KW/filler model

The feedback kind of a path on top provides the optimal path to traverse through keywords/fillers any number of times generating free length sequences of keywords and fillers.

The weights corresponding to the keywords are **WK i** that corresponding to the fillers are **WF i.** They are the key factors while playing a tradeoff between hits and false alarms. As already discussed, relative weights assigned to the keywords are used to boost to likelihood of keyword being detected. It's like we are emphasizing there occurrence beyond just the acoustics matching. They could be incorporated into the system just like a language model score. Actually, they are sort of similar to language models from the perspective that both acts as an extra information about the likelihood of occurrence of a particular word. When the value of **WKi** s are more relative to **WFi** s then we are sort of forcing the decoder / recognizer to make a decision 'biased' towards the keywords. Thus increments of the keyword weights will definitely give a hit detection rate by boosting acoustics scores for keywords but at the same time it will result in more and more false alarms. In HTK the weights are assigned as an effective unigram language model score. The word network file is manually modified for assigning the weights. Say, all keywords have same weight and so has the fillers. Their weight ratio is K: 1, i.e. WKi = k .WFi. This is implemented by assigning language model (LM) score for KWs as 'K' (though not a real LM scores) and that for fillers as 1 in the word network. So, LM scores are effectively serving the purpose of weights here.

3.4.2 Approach 2

This approach is based on all-phone network. As already discussed, it allows any sequence of phonemes and hence using all English phonemes in the network, the purposed of fillers can be served pretty well as the filler it self here is English language. In this network, setting the value of relative word insertion penalty plays a major role as it decides the tradeoff between hits and false alarms. It will be discussed more in detail in the next section. Same word based models for Bengali keywords were used in parallel to the all-phone network. The network structure is shown in Figure 3-5.

The weights that could be seen corresponding to the keywords are meant for incorporating some features from the Approach 1 into this and try to better the performance of the system with an acceptable false alarm rate. This will be discussed in detail later.

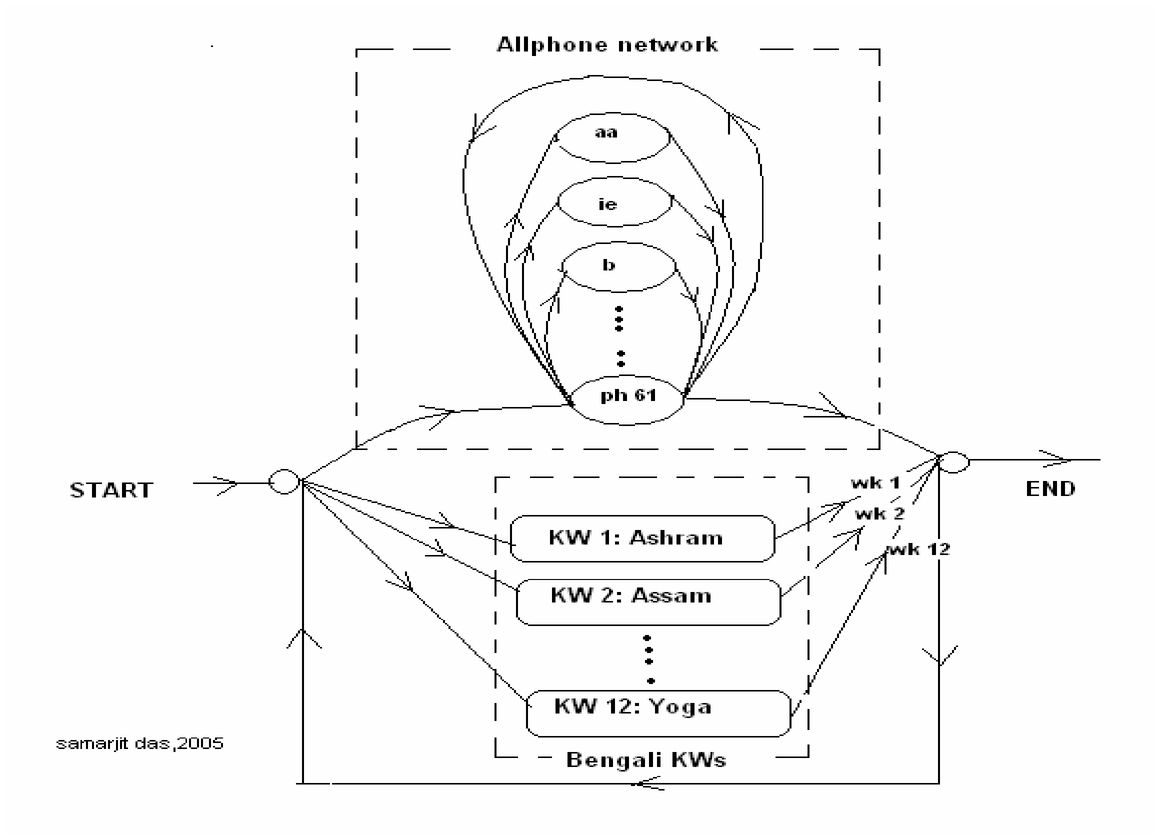

Figure 3-5 All phone network based grammar network

The master-macro file of HTK used for decoding /spotting had the entire trained phoneme set along with the word based keywords models together.

3.5 Decoder and its parameter selection

This was the actual working module of the system or so called 'Bengali word spotter'. HTK tool 'HVite' with appropriate set of parameters was used for this purpose. It used Viterbi search algorithm trace out the most optimal path (one with the best acoustics score) through the trained phonetic/word lattice. It's basically a time-synchronous search and sort of a 'breath first search' with pruning. The dictionary of the decoder was modified in such a way that it displayed the Bengali keywords only, with duration information, if any, in the output transcription file. Also, the decoder could be used in real-time or 'online' by using HVite with proper HTK configuration file. Decoder could also provide with N-best out put with corresponding scores, for a particular utterance and this is essential for post processing module to reduce FAs and increasing hits as well. Values of two decoding parameters were to be chosen optimally. The first one is 'language model scale factor' and the second one is relative word insertion penalty. They are discussed is the section to follow.

3.5.1 Language model scale-factor

This is the factor by which the language model score is scaled while decoding. It influences the relative levels of insertion and deletion errors of the system. The effect of varying this parameter will be studied during system evaluation. An optimal value was chosen after detailed performance analysis. In HTK it is set by the –s option of the tool 'HVite'.

3.5.2 Relative Word insertion penalty

This would govern the performance characteristics of the all-phone network based approach. This is a fixed 'value/offset' added to the scaled LM score during decoding. It was set by the –p option of 'HVite'. That means the more negative is the value of p the more is the score deduction up on a symbol generation (passing through a token) by the decoder. Its effects on the recognizer's/word-spotter's performance and the relevant explanations will be dealt during system analysis and discussion in the chapter to follow.

3.6 Summary

This chapter has discussed the detailed system design for a speaker dependent Bengali keyword spotter in unconstrained English speech. Two major approaches, depending on the type of filler models and grammar networks have been explained in detail. In the next chapter, the system will be evaluated and analyzed. Results from various approaches in terms of filler modeling, grammar networks, and number of filler models used etc will be compared and an optimal design will be sought.

Chapter 4

System evaluations and analysis

This chapter deals with results obtained from evaluations of the word spotter whose design has been explained in the previous chapter. Relevant discussions have accompanied each analysis. Different approaches will be compared in terms of system performance and possibility of utilizing advantages of different methods into a single system will be explored. All evaluations have been done under HTK v3.2.1 platform and same acoustic parameters as training phase were used. Total 240 utterances were used. Half of them had one or more Bengali keyword embedded in them while rest had none. Evaluation is NOT done on number of utterances as each could have numbers of keywords and detecting each of them is important. So, performance is reported relative to number of total keywords. Total number of keywords embedded in 120 utterances was 144. Total duration of test data was 0.3010 hrs or nearly 18 minutes. This will be used for evaluating performance measure in terms of FA/KW/H. In all cases LM scale factor will be addressed as 's' and word insertion penalty as 'p'.

4.1 Evaluation of Approach 1

This approach used whole word based models for both keywords and fillers. Details about this approach can be found in Chapter 4. Basically, the system was evaluated using single Gaussian HMM based four filler models. Results obtained from 'one filler model' approach were compared and the effects of introducing more number of mixtures in HMMs were also studied. It is important to note that substitutions won't be considered separately. 'Substitutions' will be considered as 'Deletions'. This is an evaluation criterion that assumes if a particular keyword is not 'correctly spotted' i.e. confused with other keyword, it is NO better than deletion. Although it's just an assumption and can be modified if required. Following sections discuss the performance depending on different parameters and system configurations.

4.1.1 Effect of relative word insertion penalty

The value of ‘s’ was fixed at 5.0 and the four different values of ‘p’ were chosen. The following table shows the performance. Total number of Bengali keywords was 144.

Table 4 -1 Performance variation of Approach 1 depending on ‘p’

p	**Hits**	**FAs**	**Missed**	**FA/KW/H**	**% Hit**
500	52	27	92	0.62	36
50	131	3	13	0.069	90.97
0	130	4	14	0.092	90.27
-50	130	4	14	0.092	90.27
-500	61	0	83	0.00	42.11

In case, the value of p was not too high (like several hundreds) in magnitude, the results did not seem to depend on p. It became evident after studying results for p=50, 0 and -50. It was expected, as both fillers and keywords were whole word based. But results could be unexpected as value of p chosen to arbitrarily on either side of ‘zero’. Both for p=500 and p=-500, results were unacceptable. False alarms is more in case of p=+500; this can be explained by the fact that the keywords are of shorter duration than prototype fillers because a whole utterance that might be containing many words was represented by only four filler models. For the same reason, FA is dropped to zero when score deduction due to word transition penalty (p=-500) became too high. Still results for p=500 and -500 not in accordance with theoretical expectations. Also, hit rate is too low. Hence such values should be carefully avoided. A reasonable value ‘p=0’ was chosen for evaluating all other cases of this approach.

4.1.2 Effect of LM scale factor

The keywords were weighted five times than the fillers and the value of 'p' was set at '0'. Performance variations on varying 's' is shown in Table 4-2

Table 4 -2 Results for varying LM scale factor

s	Hits	FAs	Missed	FA/KW/H	% Hit
1	130	4	14	0.11	90.27
5	131	4	13	0.09	90.97
10	132	5	12	0.11	91.66
20	133	15	11	0.34	92.36
25	133	22	11	0.50	92.36
30	133	36	11	0.83	92.36
35	135	51	9	1.17	93.75
40	138	72	6	1.66	95.83
50	138	133	6	3.06	95.38
100	138	1422	6	32.80	95.38

The table above clearly shows that increment of 's' leads to increment in the number of false alarms. But %hit is also improved. Reason is, as the scale factor increases relative score difference with fillers increases, give a specific weight ratio. Or, mathematically, let's say given a KW: FL weight ratio of W: 1 and s=1 the LM score difference between a specific KW (keyword) and FL (filler) is D; i.e.

$$\mathbf{D= (LM\_score\_KW) – (LM\_score\_FL)} \tag{1}$$

But, when s=k,

$$\mathbf{D'=k. [(LM\_score\_KW) – (LM\_score\_FL)]} \tag{2}$$

Or,
$$\mathbf{D'=k.D} \tag{3}$$

Or, relative score difference increased s times. So, this is to say for given weight ratio, increase in s will lead to a situation where decoder will be more biased toward keywords. And, hence increment in the %hit at the cost of high false alarms. It is also to be noted that as increasing s increases the likelihood of KW detection (explained above) so, tradeoff between FAs and %hit is reflected in the table above.

Actually, every when there's an increase in the %hit there will be an increase in the FA rate. That is inevitable. It doesn't matter which parameter variation is leading to that. We always have to make a compromise!

After all the analysis, the optimal value of s was chosen to be 20, at which 95.83% hit was achieved with FA (false alarm) rate of 0.34. This would mean that out of every three keywords appearing at the output one might be a false alarm. Value of s will be kept fixed at this value for further analysis.

4.1.3 Effect of relative keyword weights

Keyword weights were also used to forcefully boost the likelihood of keywords while decoding. All the keywords were set to have same weights. Similar thing was applicable for the fillers. The ratio between KW weights and FL weights (WK:WF) was varied and the tradeoff between hit and false alarm was studied. The ratio was represented as K: 1 (i.e. Wt_KW: Wt_FL). Value of s was set at 20. Following table shows the system performance variations upon varying K.

Table 4 -3 Results for variation of relative KW weights

K:1	Hits	FAs	Missed	FA/KW/H	% Hit
1:1	130	4	14	0.092	90.27
2:1	131	8	13	0.184	90.97
5:1	133	15	11	0.34	92.36
6:1	134	38	10	0.876	93.05
7:1	136	51	8	1.17	94.44
8:1	137	67	7	1.54	95.25
10:1	139	130	5	2.99	96.52
11:1	139	180	5	4.15	97.22
12:1	140	234	4	5.40	97.22
13:1	141	305	3	7.03	97.92
14:1	141	413	3	9.52	97.92
15:1	141	535	3	12.34	97.92

Diagrammatic representations of Hits, FAs and Misses/Deletions with respect to varying K are shown in Figure 4 -1 to Figure 4-3.

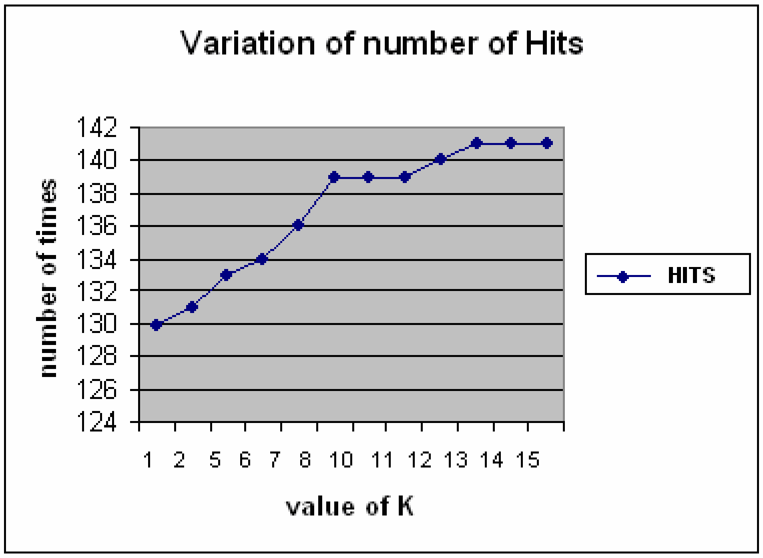

Figure 4 -1 Variation of number of hits with varying K

This finding is in accordance with the theoretical expectations. As value of K increases the likelihood of keywords relatively increases giving a better hit rate.

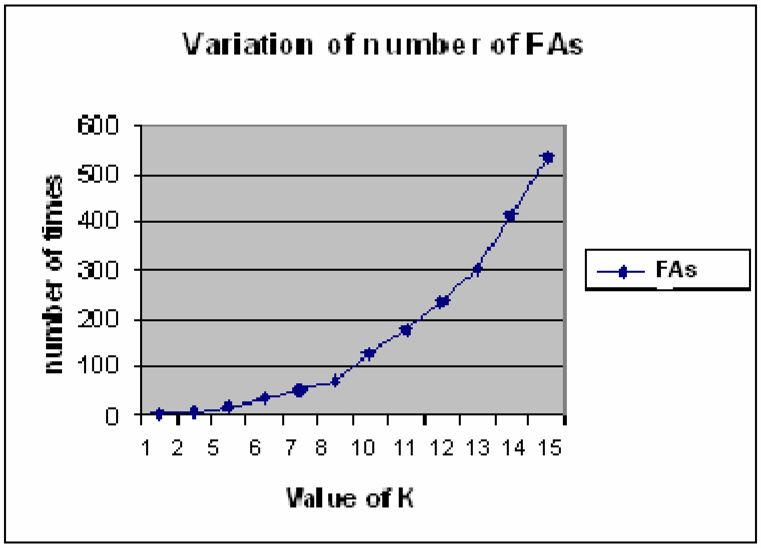

Figure 4 -2 Variation of number of FAs with varying K

This result is also in as expected. As can be seen in the Figure 4 – 2, there was an exponential increase in the number of FAs for higher values of K.

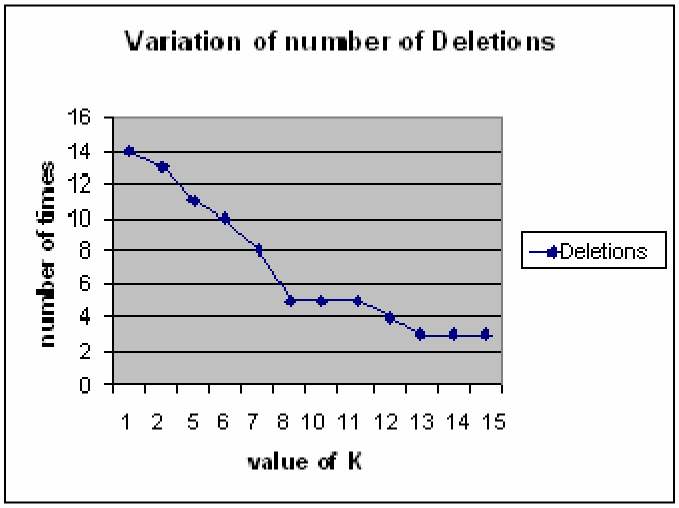

Figure 4 – 3 Variation of number of Deletions with varying K

This is just a plot of 'number of hits' deducted from total number of keywords i.e. 144.

The tradeoff between FAs and Hits were studied by analyzing FA/KW/H and percentage Hit. To reflect this, the ROC for the keyword spotter was drawn which is shown in Figure 4 – 4.

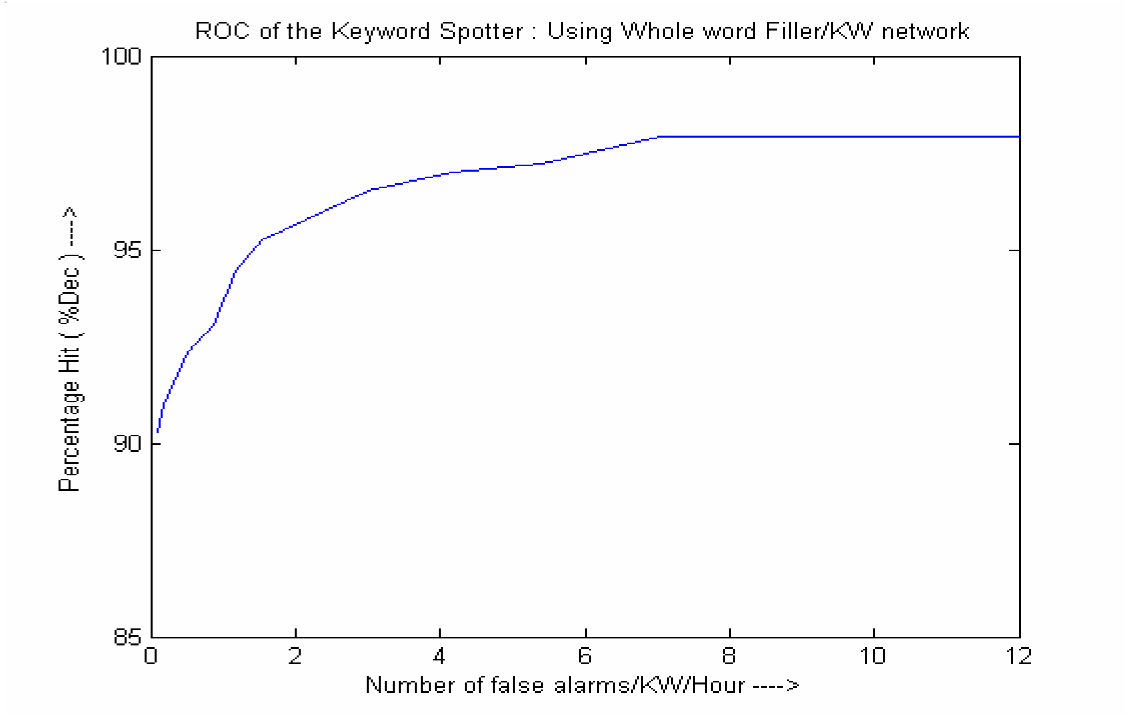

Figure 4 – 4 ROC of the Bengali keyword spotter

It is to be noted that some data points of the ROC was a little bit manipulated for a smother fit. Having analyzed the whole system and studying the ROC, finally optimal keyword weight could be chosen to be 7. The optimal performance being 94.44% hits with a false alarm rate of 1.17 FA/KW/H which is close to 1. Physically that means that each keyword at the out put is equally likely to be a false alarm. Never the less, depending on applications any other design could be chosen that gives a lower FA rate at the cost of lower %hit.

4.1.4 Overall performance of approach 1

Optimal design chosen for approach 1 is as follows. Single Gaussian HMMs, Four filler models, LM scale factor s=20, word insertion penalty p=0, relative KW weight ratio, K: 1=7:1. Overall performance with the optimal design is %Hits=94.44 and corresponding FA/KW/H=1.17. The maximum %hit achieved by the system was 97.92 %(~ 98%) with 7.03 FA/KW/H and the corresponding relative keyword weight ratio was 13: 1.

4.1.5 Some other evaluation issues

Apart from all those discussed above some other evaluation issues were also taken into consideration. The first one being number of filler models used. The system was tested with one filler model trained with all non-keyword utterances. But this did not made a large difference any way as it is sort of similar to using four concatenated fillers. Ultimately, there would not be many differences in the single filler model with each of the four concatenated filler as they don’t have any specific speech reference. So, at the time of decoding sequence generated by the concatenation of four fillers will be represented by repetitions of the single filler model. But this does not hamper the keyword part any way. FA rate also remained same as the two kinds of fillers were sort of identical as already told.

Separate training for keywords and fillers did not show any significant performance improvement. This might be due to insufficient training/testing data. It could be left for further study using large databases.

Already the system had shown close to 100% performance (hits) using single Gaussian HMMs. After using increased number of mixtures, no significant improvement in the performance was observed. Rather using large number of mixture Components (say, 16) with insufficient data (as the case here) had shown to have degraded the system performance due to ‘over fitting’. Of course, it is expected that with large training/testing data base, introduction of mixtures will improve the system performance compared to single Gaussian. It that case problem of over fitting also won’t appear provided large enough database is used for training. During evaluation, it has to be made sure that large enough test database is used to ‘reflect’ the improvements in the performance due to more ‘detailed acoustic models’ due to introduction of mixture components.

4.2 Evaluation of Approach 2

This approach used all-phone network. Details about this could be found in chapter 3. Similar evaluation was also done for this approach like the previous one. Similar assumptions were considered. Also, single Gaussians were used. As this approach involves NO keyword weights as such so there should not be any significant effect on the system performance for varying ‘s’. The value of ‘s’ was kept fixed at 5.0. The value of word insertion penalty ‘p’ was varied to investigate the tradeoff between FA and correct hit.

4.2.1 Varying the value of LM scale factor

By doing this, as per assumption, NO effect on result was found. The Following table shows that.

Table 4-4 Results obtained by varying the value of ‘s’

s	Hits	Missed	FAs	FA/KW/H	% Hit
25	138	6	47	1.08	95.83
5	138	6	46	1.06	95.83
0	138	6	46	1.06	95.83

4.2.2 Effect of word insertion penalty

The following table shows the results obtained from varying the word insertion penalty.

Table 4 -5 Results obtained by varying ‘p’

p	Hits	Missed	FA	FA/KW/H	% Hit
30	127	17	4	0.09	88.19
20	134	10	19	0.43	93.04
10	138	6	31	0.71	95.83
5	138	6	39	0.90	95.83
0	138	6	46	1.06	95.83
-5	138	6	56	1.29	95.83
-10	138	6	64	1.47	95.83
-15	138	6	77	1.77	95.83
-20	139	5	90	2.07	96.52
-22	139	5	95	2.19	96.52
-24	139	5	102	2.35	96.52
-25	140	4	105	2.42	97.22
-26	140	4	110	2.53	97.22
-27	140	4	110	2.53	97.22
-28	140	4	112	2.58	97.22
-29	139	5	113	2.60	96.52
-30	139	5	113	2.60	96.52
-40	139	5	130	3.00	96.52
-60	140	4	170	3.92	97.22
-75	140	4	192	4.42	97.22
-90	141	3	193	4.45	97.92

Diagrammatic representations of Hits, FAs and Misses/Deletions with respect to varying p are shown in Figure 4 -5 to Figure 4-7.

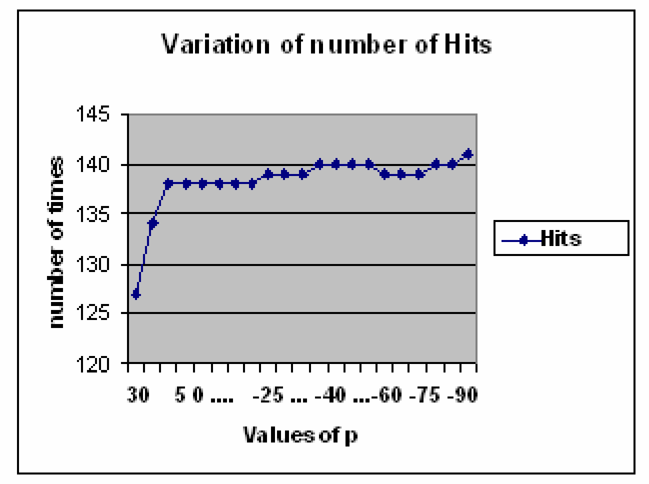

Figure 4 – 5 Variation of number of Hits with varying p

As can be seen in the Figure 4 -5, increasing p has led to increase in number of hits but the variation characteristics is different from that of approach. The analysis will follow in the next section.

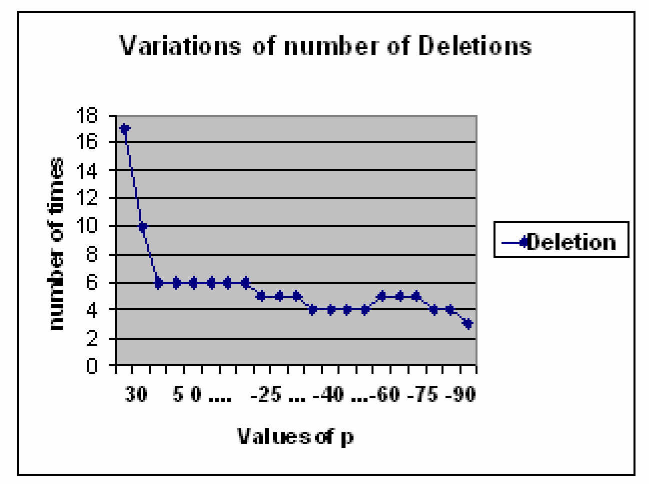

Figure 4 – 6 Variation of number of Deletions with varying p

Like approach 1, in this case as well, it's just a characteristics obtained by deducting the number of hits from total number of keywords.

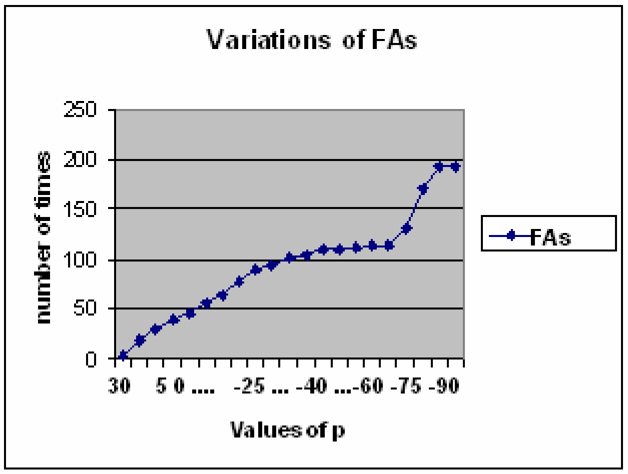

Figure 4 – 7 Variation of number of FAs with varying p

Number of false alarms variations, as can be seen, showed quite different characteristics compared to previous approach. Here, a lesser 'abrupt' variations was found. Also, variations range relative to %hit was less as well.

The system's tradeoff between false alarms and correct hits was captured by the ROC. But, this time the characteristic was sort of deviated from expected near 2.5 FA/KW/H and hence an approximated version is also presented in the Figure 4-8. The reasons behind these kinds of deviations have been sought in the section to follow.

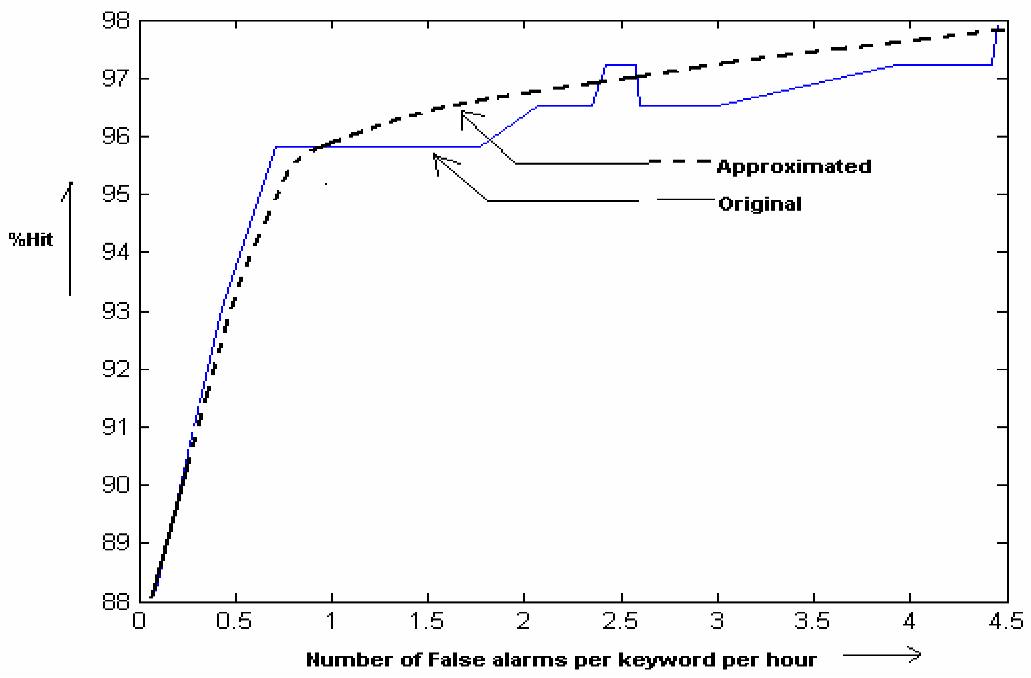

Figure 4-8 ROC of the system using Approach 2

4.2.3 Performance analysis of approach 2

As the value of p becomes more and more negative, more is the ‘penalty’ for each symbol insertion at the output. In other words, score deducted for traversing through each token (phonemes/whole keywords) is more. But, it is to be remembered that equal scores are deducted, be it a phoneme token or a whole word (keyword) token. So, a segment matched with keywords will have lesser score deduction compared to that with phonemes. Because, phonemes are of relatively lesser durations. So, representation of a segment by phonemes will involve more symbol generation and hence more pruning. This is the reason behind the fact that as pruning increases (more negative value for p) the out put gets biased to output keywords (high score due to lesser pruning)instead of sequence of phonemes (filler) . Thus False alarms also increase.

So, here word insertion penalty is effectively playing the role of keyword weights in the previous case. But, it is to be noted that the variations are not similar. FAs don’t vary in the same as they did for increasing keyword weights. Here some times FAs might remain constant with increasing insertion penalty or some times might even decrease when value of p is in a higher range. Same can be said about hits. The reason is ‘Substitutions’ were considered to be ‘deletions’. Now, while a high value of pruning (more negative p) is used then out put even gets biased to ‘longer’ duration keywords. This leads to substitution which is considered here as an effective deletion. For the case of false alarms as well, instead of more and more insertions the output gets mapped to fewer numbers of ling duration words like ‘Bramhaputra’. This was exactly found to happen practically. This explains all the performance characteristics found above and their deviation from what was ideally expected.

4.2.4 Overall performance of approach 2

Approach 2 had been found to out perform approach 1 in terms of trade off between FAs and %hit. The system showed a maximum hit rate of 97.92 %(~98%) with 4.45 FA/KW/H with value of p=-90. This is way better than sane hit rate with 7.03 FA/KW/H obtained in approach 1. Also, optimal performance of 95.83% (~96%) of hit with 0.71 FA/KW/H is better than 94.44% hit with 1.17 FA/KW/H of approach 1.

4.3 Improving the system performance

Although the performance of approach 2 was found to be better than that of approach1 but there was a problem with longer duration words in case of approach 2. To get rid of this, approach 1 was merged with approach 2 to take advantage of KW weighting to solve that problem. As it was found that longer duration words like 'Bramhaputra' tends to substitute / Generate FA so, the word was weighted relatively less compared to all others in a similar way as in approach 1. This weighted keyword sub network was used with all-phone network of approach 2. The combined approach, with word Bramhaputra weighted less than other keywords, but all keywords being more weighted (relatively) then phonemes in all phone network gave a very high %hit of 98.61(~99%) hit rate but at the cost of 11.30 FA/KW/H !

Several other ways could also be tried for a better system performance. Since the word spotter is speaker dependent, the %hit is already close to 100 %. But for a more complicated case detailed model (mixture Gaussian) for keywords could be used to reduce number of misses. Amount of false alarms should also be reduced. A better filler model could serve the purpose. Multiple mixture Gaussians would be a very good choice. But as here, the test database is not too large so performance improvement could not be reflected. Also, several other confidence measures could be used to improve the performance in terms of false alarms. Those were left for future works and will be dealt in next chapter.

4.4 Summary

This chapter has analyzed two major approaches used to develop a Speaker dependent Bengali keyword spotter in unconstrained English speech. The methods were compared in terms of several respects. Optimal choice of parameters with an acceptable FA rate was chosen. Percentage hit was close to 100% even though substitutions were considered as deletions. Depending on applications substitutions can also be considered as a putative hit. Thus system performance might even better. Results are 'good' for speaker dependent case as far as %hit is concerned. Several methods to reduce FAs for even a better system design will be discussed in future works section of next chapter.

Chapter 5

Conclusions & future works

With the advent of high speed data processing devices, spoken language interfaces with advanced speech recognition algorithms becoming more practicable. The speaker dependent Bengali keyword spotter could be used as a preliminary module for an English-Bengali bi-lingual interface.

5.1 Conclusions

The system was carefully designed using two different approaches. Approach one gave an optimal performance 94.44% hit at an FA rate of 1.17 FA/KW/H. Similar performances were obtained while using one and four concatenated filler models. Single Gaussians worked well. Multiple Gaussians were also used. Due to unavailability of a large Bengali database, keyword models with many mixture components showed a problem of over fitting. Relative keyword weights were used. They were varied to study their effect on system performance. The %hit increases as keyword weights increases but numbers of false alarms were also found to increase. Similar results were found when the language model scale factor was varied. It was found to be a good idea to set the word insertion penalty at '0' for this approach. Overall system performance was satisfactory (above 90% hit) with very low FA rate (0.092 FA/KW/H).

Approach two had all phoneme models for English extraneous speech. It used an all phone network to implement the filler part. Same whole word based keyword models were used in parallel. Performance was found to be better than approach one. It achieved a maximum hit rate of 97.92% with 4.45 FA/KW/H. An optimal performance was chosen. It had 95.83% hit with just 0.71 FA/KW/H. It was found that relative word transition penalty should be kept at a reasonable value. Otherwise, some unpredictable results might come up. Long duration words seemed to create problems while decoding using high word insertion penalty. A hybrid approach was attempted to assign lesser priority to the long duration words by relative word weighting and that, used with all phone network gave a high %hit as high as ~99%. Never the less, it is firmed believed that several post processing techniques can improve performance in terms of FAs. Use of Gaussian mixtures densities will also be useful for large training/test database systems. This concludes the design and analysis of a 'Speaker dependent Bengali Keyword spotting system in unconstrained English speech.'

5.2 Future works

There are ample lot of things that can be done to improve the system and making it even more efficient. Most important thing is to note that NO database for Bengali available. It was rather created by few hundred utterances from a single speaker. Therefore, a lot of compromise had to be done regarding the system design. Further works using appropriate testing and training databases will definitely lead to a better and speaker independent Bengali word-spotter. Some discussion follows.

5.2.1 Speaker independent system design

The system designed in this project is a Speaker dependent one. This is basically due to unavailability of training data representing a wide class of Bengali speaker. The same system can be further trained with a large Bengali database. This is supposed to give keyword models optimized for a 'specific Bengali word' and independent of speaker uttering it. So, this kind of Speaker independent Bengali keyword spotter will be more useful for bi-lingual systems to be discussed soon.

5.2.2 Post processing & confidence measure

The results from already existing system can be fed to a post processing module to reduce false alarms and improve system performance. Number of confidence measure schemes can be used for this purpose. N-best out put can be used to make a 'better choice' and avoid false alarms (FAs).Duration normalized rescoring and likelihood ratio scoring (Rose and Paul, 1990) be used to use duration information to remove false alarms. Discriminative technique like support vector machines (SVM) can also be used to generate confidence measure for implementing a better rejection strategy.

5.2.3 English-Bengali bi-lingual interface

A speaker independent language specific keyword spotter can be used to find 'language boundary information' in a bi-lingual system which is very crucial for any multilingual system. So, the existing word spotter can be very useful if it is upgraded to a speaker independent system and provided with a larger keyword vocabulary. Finally, all of these and beyond are left for 'future works' to explore.

References

[1] R. Rose and D. Paul, "A Hidden Markov Model Based Key- word Recognition System," 1990 IEEE ICASSP, pp. 129- 132

[2] J.G. Wilpon, L.R. Rabiner, C.H. Lee, and E.R. Goldman, "Automatic Recognition of Keywords in Unconstrained Speech Using Hidden Markov Models," 1990 IEEE Trans. ASSP, Vo138. No. 11, pp. 1870-1878.

[3] R. Rohlicek, W. Russell, S. Roukos, H. Gish, "Continuous Hidden Markov Modeling for Speaker-Independent Word Spotting," 1989 IEEE ICASSP, pp. 627-630

[4] Lam Hiu Sing, " Audio search of surveillance data using keyword spotting and dynamic models," *M.Phil thesis* , the Chinese University of Hong Kong, 2001.

[5] K. Ohtsuki et al "Topic extraction with multiple topic-words in broadcast-news search," Proceedings of ICASSP, Vol 1,pp 329-332

[6] Peter Schwartz et al "Phoneme based acoustic keyword spotting in informal continuous speech," *Faculty of Technology*, Brno University, Czech Republic.

[7] S. J Young et al" Video Mail Retrieval", *Cambridge University Engineering Department*, Cambridge, UK.

[8] Bridle, J.S., "An efficient elastic-template method for detecting given words in running speech," Proc. of the Brit. Acoust.Soc. Meeting, pp. 1-4, April 1973.

[9] Mitch Weintraub et al "Neural network based measure of confidence for word recognition," *Speech Technology and research laboratory, SRI international* Menlo Park ,CA.

[10] Higgins, A. L.; Wohlford, Robert E., "Keyword Recognition UsingTemplateConcatenation,"ICASSP'1985, 3: 1233-1236, 1985

[11] J. G. Wilpon, C.-H. Lee, and L. R. Rabiner, ``Application of Hidden Markov Models for recognition of a limited set of word in unconstrained speech'', in *Proceedings of 1989 ICASSP*, Glasgow, April 1989, IEEE, vol. I, pp. 254--257.

[12] Joe Drish et all, “ A support vector machine based rejection technique for speech recognition,” *Department of Computer Science*, University of California, San Diego, CA.

[13] Silaghi, M.-C. and Bourlard H., “Posterior-Based Keyword Spotting Approaches Without Filler Models,” Swiss Federal Institute of Technology Lausanne (EPFL), Technical Report,1999.

[14] L. R. Rabiner, ``A tutorial on Hidden Markov Models and selected applications in speech recognition'', *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257--286, February 1989.

[15] Lleida, J. B. Mariño, and A. Moreno, ``Telemaco - a real time keyword spotting application for voice dialing'', in *Proceedings of EUROSPEECH'93*, Berlin, September 1993, vol. III, pp. 1801--1804.

Bibliography

1. "Automatic Speech Recognition", *Spring 2003*. Instructor Prof. Jim Glass. " MIT Open Courseware". MIT. Cambrigde,Massachussets. USA.

2. " Digital Processing of Speech signal", *Spring 2005*, Instructor Prof. Tan Lee. Department of EE, the Chinese University of Hong Kong. Hong Kong.

3. "Spoken Language Processing". Xuedong Huang, Alex Acero, Hsiao-wuen Hon., Microsoft Research. Prentice Hall. New Jersey,2001.

4. "Robustness in Automatic Speech Recognition", Jean-Claude Junqua, Jean-Paul Haton, CRIN-INRIA, France. Kluwer Academic Publishers.

5. "The HTK book." Steve Young et al , Cambridge University. Cambridge, UK.