FILLER MODEL BASED WORD LEVEL CONFIDENCE MEASURES FOR SPOKEN DIALOGUE SYSTEMS

by

AYDIN AKYOL

Submitted to the Graduate School of Engineering and Natural Sciences
in partial fulfillment of
the requirements for the degree of
Master of Science

Sabancı University

July 2003

APPROVED BY

Prof. Dr. Kemal OFLAZER .....
Department of Computer Science, Sabancı University
(Thesis Supervisor)

Assist. Prof. Dr. Hakan ERDOĞAN .....
Department of Telecommunications, Sabancı University

Assist. Prof. Dr. Hüsnü YENİGÜN .....
Department of Computer Science, Sabancı University

DATE OF APPROVAL: .....

to My Family (Şüheda, Halil, Ayhan Akyol)

Acknowledgments

This thesis is not only the result of two years of work on my part, but also two years worth of help, support and encouragement from the following people;

- Dr. Hakan Erdoğan, for his help and support throughout the project. He always guided me at every stage of my studies, and gave me great ideas and suggestions. Without him, this thesis would not have been possible.
- Prof. Kemal Oflazer, for his time, patience, insight and knowledge. His valuable suggestions brought a different perspective to my thesis.
- Dr. Hüsnü Yenigün, for his insightful feedback to various aspects of my thesis work.
- My parents and brother, for their love and support. Thank you for always believing in me.
- Esra Vural, my officemate. She was very helpful in many ways, from the long discussions we had on speech processing and various research topics.
- Imad Abdel Hafez, for his valuable introduction to HTK.
- All volunteers contributed to the data collection, for their courtesy and patience
- Cambridge University Speech Processing Group, for producing the speech recognition toolkit, HTK.

FILLER MODEL BASED WORD LEVEL CONFIDENCE MEASURES FOR SPOKEN DIALOGUE SYSTEMS

Abstract

In conversational dialogue applications it is critical to understand the requests accurately. However, the performance of current speech recognition systems are far from perfect. In order to function effectively with imperfect speech recognition, an accurate confidence scoring mechanism should be employed. To determine a confidence score for a hypothesis, certain confidence features are combined.

In this thesis, the performance of filler-model based confidence features have been investigated. Five types of filler model were defined: triphone-network, phone-network, phone-class network, 5-state catch-all model and 3-state catch-all model. First all models were evaluated in terms of their ability to correctly tag (recognition-error or correct) recognition hypotheses. Here, the best performance was obtained from triphone recognition network. Then the performance of reliable combinations of these models were investigated and it was observed that certain reliable combinations of filler models could significantly improve the accuracy of the confidence annotation.

On the practical side of the work, a dialogue management system was implemented for Turkish. For acoustic model training, a 3-hour speech data was collected. To increase the speech recogntion accuracy, some parameters were tied by using decision trees. For an accurate clustering, an efficient set of questions was constructed for Turkish triphones. With this setup we obtained %97.2 word recognition accuracy.

Keywords: *Confidence Scoring/Measuring, Filler/Garbage Models, Dialogue Systems, Confidence Measures/Features, Speech Recognition*

DİYALOG SİSTEMLERİ İÇİN KELİME BAZINDA GÜVENİLİRLİK ÖLÇÜTLERİ

Özet

Otomatik diyalog sistemlerinin yetkinliği için konuşmacının isteklerinin tam olarak anlaşılması büyük önem taşımaktadır. Buna karşın günümüz otomatik ses tanıma sistemlerinin performansı bu yetkinliği sağlamaktan çok uzaktır. Ses tanımadaki bu eksikliğe rağmen diyalog sistemlerinin yüksek performansla çalışmalarını sağlamak için ses tanımadaki güvenirliliği tam olarak tespit edebilecek mekanizmaların geliştirilmeleri gerekmektedir. Bu tür mekanizmaları oluşturmak için ses tanımanın güvenirliliği hakkında bilgi veren özellikler belirlenir ve bu bilgiler birleştirilerek tanımanın doğruluğu üzerine bir değer belirlenir.

Bu tezde, *filler/garbage model* tabanlı güvenilirlik ölçütlerinin performansları incelenmektedir. Beş tür *filler/garbage modeli* tanımlanmıştır: bağlam içerikli ses ağı, bağlamdan bağımsız ses ağı, ses sınıfı ağı, 5-durumlu yığın modeli ve 3-durumlu yığın modeli. İlk olarak tüm bu modeller ses tanımadan çıkan sonuçları, *doğru* veya *tanıma hatası* olarak doğru sınıflayabilmedeki başarılarına göre değerlendirildiler. Daha sonra bu özellikler çeşitli kombinasyonlarda biraraya getirilip, ortak performansları incelendi ve kimi kombinasyonların ses tanıma sonuçlarının güvenilirliğini tespit etmedeki doğruluğu önemli derecede arttırdığı gözlemlendi.

Tezin uygulama tarafında Türkçe için bir diyalog sistemi gerçeklendi. Bu sistem için, ses-birimlerinin modellenmesinde kullanılmak üzere 3 saatlik ses verisi toplandı. Ayrıca ses tanımadaki performansı arttırmak için bazı model parametrelerinin *decision tree* yöntemiyle birleştirilmesi yoluna gidildi. Bu noktada doğru birleştirmelere karar verebilmek için, Türkçe seslere yönelik etkin bir soru seti hazırlandı. Bu şekilde gerçekleştirilen ses tanıma sisteminde %97.2 lik kelime tanıma doğruluğuna ulaşıldı.

Anahtar Kelimeler: *Güvenilirlik Belirleme/Ölçme, Filler/Garbage Model, Diyalog Sistemleri, Ses Tanıma, Güvenilirlik Ölçütleri/Özellikleri*

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivational Consideration

Sophisticated interaction with speech understanding system require the use of dialogue management systems. Human-computer dialogue systems guide the user by strategies to provide necessary elements of a *task* description that the system knows and considers practically feasible and try to accomplish the required *task* by performing the appropriate (communicative and/or non-communicative) actions [34].

Today dialogue systems usually are employed in restricted domains, since this makes speech recognition more robust. The speaker responds to questions or asks her questions to a proposed menu by using a small set of allowable words. Current heavy research in this area aims to make dialogue systems more friendly by allowing the user speak with more freedom.

For such dialogue applications it is critical to understand the user's requests accurately, since the rest of the system acts based on this recognized utterance. On the other hand, the performance of current speech recognition systems are far from perfect. In order to function effectively with imperfect speech recognition, dialogue management systems should have an accurate confidence scoring mechanism to evaluate each recognized utterance. A robust recognition confidence estimator improves the efficiency of the dialogue system by requesting confirmation or repetition of uncertain words. For example, assume that the user asks such a question to the dialogue system;
What is the name of the instructor for the course, EE201?

It is possible for the speech recognizer to hypothesize this question;
What is the name of the instructor for the course, ME201?
In such a case if we fully trust the recognizer's results, then the next action of the dialogue manager would be searching the instructor for the course ME201. On the contrary the user had requested the information for another course. In order to not allow such kind of misinformative dialogues, each recognition should be evaluated in terms of certain recognition confidence measures.

In this work first a dialogue management system for Turkish was developed to be used as the baseline system for this research. The dialogue system is developed to automatically respond to enquiries about the university courses. Then, certain confidence features were defined and the performance of these and their combination in confidence annotation were investigated.

1.2 Problem Definition

Robust confidence measures, which provide information about the accuracy of the recognition, are critical to make speech applications more accurate and usable since recognition performance is less than perfect. Even when the accuracy is high, they are needed to detect out-of-vocabulary words or non-speech sounds. Thus, an accurate confidence scoring technique should take into account various factors which can contribute to misrecognitions and give reliable estimate of which words in the output from the recognizer are likely to be correct and which can probably be disregarded as incorrect.

A typical approach for confidence scoring includes two steps. First, a confidence feature vector is formed by combining one or more basic features correlated with word confidence. Then one of a variety of classifiers is applied to this vector to determine confidence for the word. Quality of the extracted features is the main determinant of the performance of a confidence annotator. So by defining informative features and forming a reliable set of such features, it is possible to increase the performance of the confidence annotation. In literature, many features have been defined [17,18,19,20,25,27]. They have been grouped with respect to their extrac-

tion sources, like acoustic model, language model, semantic, recognition lattice, or n-best list [25]. The use of acoustic features can be very useful for most speech recognition systems. In this thesis we have investigated the performance change in confidence annotation by defining and combining certain acoustic features based on filler-models with differing level of acoustic details.

1.3 Review of the Literature

The correctness of recognition results has been broadly studied by several authors. From many different perspectives this field has been addressed. These works can be grouped mainly into two broad classes: those based on verifying keywords, in other words keyword spotting applications, and those who build confidence measures out of feature compilation procedure on grammar based conversation systems. Since this thesis is included by the second category, in this section we present some works from literature related with confidence measuring using certain combinations of features on grammar based systems.

The normalized decoder score is a milestone in the generation of confidence measures [35,39], but it is not the only way to reliably label recognition results as correct or incorrect. To attack this problem a set of confidence features are extracted from the computations performed during recognition. The extraction of multiple confidence features has been investigated in many research efforts [17,26,19,18,20,25]. These features can be obtained; before recognition (a priori, e.g. language model) [17,20,27], after recognition (a posteriori, e.g. semantic parsing) [18,19], or during recognition process (e.g. acoustic scores, N-best hypotheses or word lattices) [27,25,20]. All this accumulated information can be compiled and combined to generate confidence measures on recognition. Here the key problem is the selection of effective features, since it is not desirable to include any knowledge source as feature because many of them do not provide any further information with the disadvantage of increasing the complexity of the system.

The widely used way to discriminate the useful features is to consider those whose magnitudes show a high correlation level to the recognition certainty. In this way

Schaaf and Kemp [27] extracted a 20 dimensional feature vector for each hypothesized word. In this work, the correlation matrix of this 20-dimensional feature space was computed and the features were eveluated as to the absolute value of this correlation coefficient. If the coefficient was high, then the corresponding feature was regarded as good. By combining the good features they have managed to decrease the error rate %27.2 as compared to the baseline system. Also Chase [17] provides us a wide variety of confidence features (he defined about 16 features for confidence annotation) to examine. After constructing predictor variables, he compared various combinations in their ability to assign probabilities of error class membership to words hypothesized. With a base rate of %84 of words in class *correct*, he achieved a %20.9 reduction in the labeling of this class. A recent study was conducted by Cox and Dasmahapatra [19]. Their main approach was to attempt to decouple the language modelling and acoustic modeling in the recognizer in order to generate independent information from these two sources for estimation of confidence. To isolate these two sources they have used a phone recognizer working parallel with the word recognizer. They have used *metamodels*, which generate alternative word hypotheses for an utterance, to estimate confidence measures using the phone recognizer output in conjunction with the word recognizer output. With metamodels they obtained %31.1 confidence gain. Although the question about which features to use is answered, the underlying question of how to combine the knowledge sources into effective confidence measures remains unresolved. A simple Bayesian classifier has been employed in [36] to build up a post-classifier of correct and incorrect results. Some early works [23,27,25] trained a neural network (more specifically, a multilayer perceptron) under the back propagation scheme as a useful means to combine different types of knowledge. Also, a similar unification system can be obtained from linear discriminant analysis [37]. Although effective, these systems exhibit a common drawback: their performances depend on the amount of samples given for their training and, therefore, they need databases of previous recognition examples to adjust their parameters and in consequence they become application-dependent.

Another way is analyzing a decision tree in which each of the features is represented as a branch. It is possible to count which are the relevant features to be taken into

account when considering a recognition result as correct. This is a schematic way to represent multiple choices among multiple values of variables. With a tree like this, it is possible to estimate the a posteriori probability that a speech segment has been correctly recognized. Taking advantage of this knowledge, in [38,23] several systems have been proposed that, with remarkable results, apply the decision trees to the generation of confidence measures.

1.4 Discussion

The previous studies showed that the most important confidence feature is the normalized decoder score [19,20,33,35,39]. The confidence of an utterance is determined by a comparison in between the best path from the acoustic unit network, or called as filler network, and the best path from the regular decoder. Filler model recognition is important since it matches all units belonging to the observation with models in the network without any dependency. Even when the observation sequence is not in the grammar or in the LM, filler network can generate the correct transcription for the input utterance by concatenating the hypothesized sub-unit transcriptions. However the normal decoder makes its matching by considering the contectivity of the trained models. Then it can be concluded that filler models can provide valuable information to detect misrecognitions. In this work we investigate the effect of filler networks in different detail on confidence estimation. Also combining such kind of filler models could provide better estimates for the hypothesis confidence. Although there are some overlapping information between the models, there could also be discriminative information specific to that model. To make both information useful in the decision process we look for the the best combination in between these models and the other acoustic features, not related with filler models, in addition to their individual contributions.

1.5 Introduction to Dialogue Systems

Automated dilaogue systems require the use of many different subsystem with a wide variety of functions. A typical dialogue system combines continuous speech recognition, confidence scoring, natural language understanding, natural language

generation and text to speech systems [30].

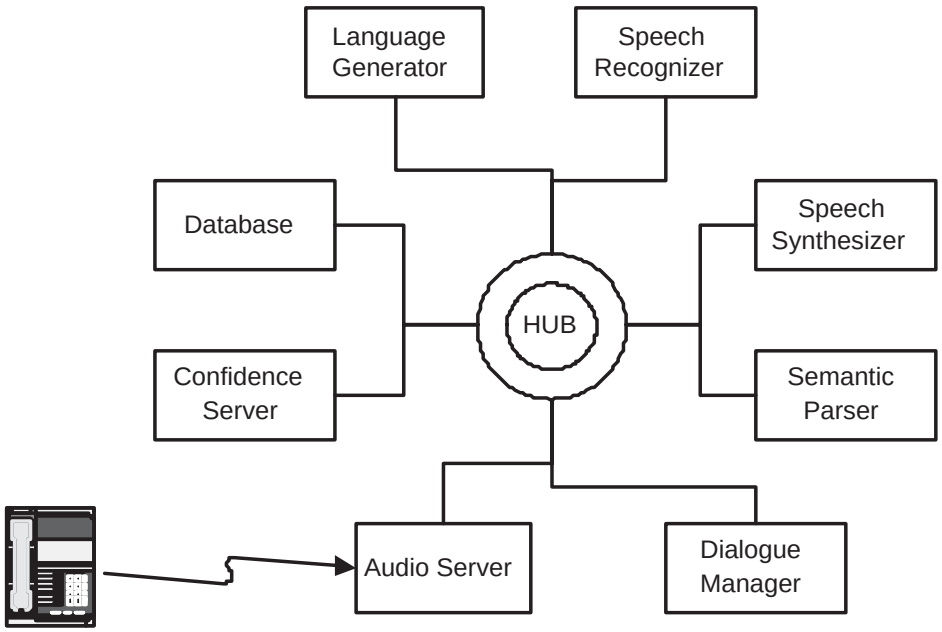

Figure 1.1: Block diagram of the functional components of a typical automatic dialogue sytem

Figure 1.1 depicts the main components of a dialogue system. A central hub controls all of the system servers through a scripting language. This kind of dialogue system structure brings flexibility to the system. If you need to change a component in the current net of the components, other components don't need to be modified. Some commercial applications provide this flexibility by a GUI supported hub script [29]. Below we present overview for each component but here, we note that in this thesis, the main focussed components are *speech recognizer* and *confidence server*.

Audio Server

The audio server is responsible for answering the incoming call, playing prompts and recording user input. This uses special hardware components such as *Dialogic* hardware environment.

Speech Recognizer

The speech recognizer receives the input vectors of the acoustic data from the audio server. In general, the recognizer produces a word lattice from which a single best

hypothesis is picked and sent to the hub for processing by the dialogue manager. In this work we used the HMM toolkit HTKv3.0, from Campbridge University [8].

Confidence Server

Current speech recognition systems are far from perfect. To be able to predict whether a hypothesis provided by automated speech recognition (ASR) system is correct or not, some confidence measures are extracted during recognition or during a separate post-processing phase. These measures can be extracted by using acoustic data, or language model or recognizer's output lattice or semantic. In Chapter 3 we examine confidence measures in more detail.

Language Understanding

The language understanding (LU) performs a syntactic and semantic analysis of an utterance using linguistic and world-knowledge constraints. For example, in our application we assume that the utterances come from a large but finite set of possible sentences. We represent such a regular set of sentences with an annotated structure, which is used to map it to a meaning representation.

The main idea of a typical semantic parser is semantically tagged context-free grammar(CFG) [31]. The nodes in a syntactic parse tree are tagged with semantic class information like Course Name or Question Word, and so on. After such a semantic parse for each recognized utterance a meaning representation is obtained. In addition to CFG, some recent parsers includes additional features like trace mechanisms to handle movement and syntactic constraints [31].

Dialogue Manager

Typical dialogue manager (DM) controls the system's interaction with the user and the application server. The DM is responsible for deciding what action the system will take at each step. The first function of a dialogue manager is that of resolving the ambiguities in the interpretation. In our application this is not very hard, since the domain of the application is quite restricted and in this restricted domain there are not many alternatives. Other functions of the DM are as:

- Extracting confidence information as an input to the confidence server.
- Interpreting the current information with previously obtained user or dialogue specific information.
- Building database queries and sending this information to Language Generation component to prompt the user.

An other point related with DM is their mode. The mode of the DM can be user-initiative or system-initiative or mixed. Deciding the mode of the DM before the construction is important since the structures and effciency of the methods used in DM varies according to the purpose of the system.

The simplest way of managing a dialogue is building a dialogue script. Another way is building a dialogue network which covers the whole dialogue framework [9]. Also Pellom *et al.* [30] made some works on event-driven dialogue managers in Colorado University. It uses the current context to decide to the next step. In this approach context consists of a set of frames and a set of global variables. The system provides a general purpose routine library to manipulate frames.

Language Generation

Language generation uses templates and information that comes from database queries to generate text based on dialog speech acts. Then the generated text is sent to the TTS system via hub.

Speech Synthesizer

Text-to-Speech (TTS) systems are used for audio output. They take text as input and synthesize an audio for this text. In this work we used the pre-implemented TTS system by Vural [32]. During the implementation of the TTS system the Festival toolkit had been used.

1.6 Outline of the Thesis

In Chapter 2, we provide a review of necessary theoretical background in some detail. This chapter includes an introduction to the main issues in signal processing

for speech recognition.

Chapter 3 gives a detailed discussion of the theory behind of the confidence scoring mechanism. The two main topics discussed are the confidence feature types and confidence feature selection.

In Chapter 4, we present practical implementation issues as well as information regarding the scope of the system. We also describe the construction of the speech corpora used for the experiments, filler models and the confidence scoring mechanism.

Chapter 5 forms the central part of this thesis in presenting filler model types developed to extract acoustic confidence information. For each filler model, training issues, model topology and performance evaluation are presented.

In Chapter 6, we introduce the extacted features to be used for confidence annotation. We discuss issues regarding classfiers used during feature combination and confidence score determination together with the results of experiments by which the performance of combined filler-model confidence features are evaluated.

In Chapter 7, we present a discussion of our results. Also some future plans to improve the performance of the suggested confidence vector are presented.

Chapter 2

Speech Recognition

2.1 Overview

Over the last few decades there has been increasing research activity in speech recognition (SR) technology. Research in automatic speech recognition (ASR) aims to develop methods and techniques that enable computer systems to accept speech input and to transcribe the recognized utterances into orthography.

Current SR systems are based on the principles of statistical pattern recognition [1]. The speech signal is a time-varying and therefore nonstationary signal. When the speech signal is divided into block of samples (called frames) it can be considered stationary and some analysis can be performed with this frame methodology. The first step of speech recognition is to convert an unknown speech waveform into a sequence of acoustic vectors, $O = o_1, o_2, ..., o_t$ (here each vector denotes the parametric equivalent of a frame in the acoustic data). If we assume that the acoustic data is produced by a sequence of words, $W = w_1, w_2, ..., w_n$, then we can define the aim of a speech recognition system as determining the most probable word sequence, $\hat{W}$, given the observed acoustic signal, O. This is represented by the Bayes' equation (2.1);

$$\hat{W} = \arg\max_w P(W|O) = \arg\max_w \frac{P(O|W)p(W)}{P(O)} = \arg\max_w P(O|W)P(O) \quad (2.1)$$

This equation indicates that the most probable sentence, W, given some observation sequence, O, can be found taking the product of two probabilities for each sentence, and chosing the sentence for which this product is greatest. These two probabilities are $P(W)$, the prior probability, computed from a language model, and, $P(O|W)$,

the observation likelihood, computed from an acoustic model.

Figure 2.1 shows the main components of a typical speech recognition system. The digitized speech signal is first transformed into a sequence of acoustic vectors at a fixed rate, typically once every 10 milliseconds. These vectors are then used to search for the most likely word candidate, making use of constraints imposed by the acoustic, lexical, and language models. Throughout this process, training data are used to determine the values of the model parameters.

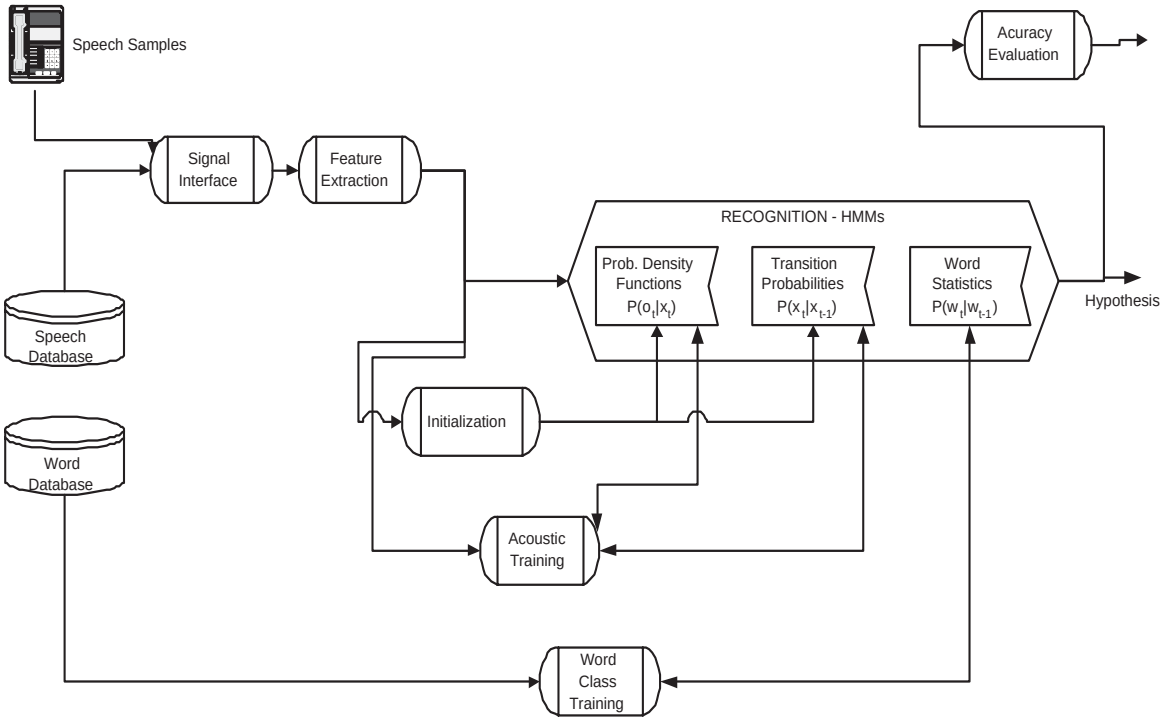

Figure 2.1: Components of a typical speech recognition system

2.2 Front-End Signal Processing and Acoustic Feature Extraction

Speech signal is identified by two properties, frequency and amplitude. Frequency determines the pitch value of the speech signal and amplitude determines the intensity and the amount of energy in the signal. The speech signal is the superposition of many sinusoidal signals of different frequencies. Each of these sinusoidal signals are the *harmonics* and in general, ASR systems benefit from the analysis and decom-

posion of these overlapped harmonics. The harmonic which has the same frequency with the speech signal is called as Fundamental Frequency and is denoted by *F0*. *F0* is quiet important in the process of boundary determination.

This section presents the whole processing of speech signal features. Although there are many other processing schemes in literature [44], in this work Mel-Frequency Cepstral Coefficients [MFCC] was used. MFCC has generally obtained a better accuracy and a minor computational complexity with respect to the other methods [44]. Figure 2.2 depicts the MFCC processing of the speech signal.

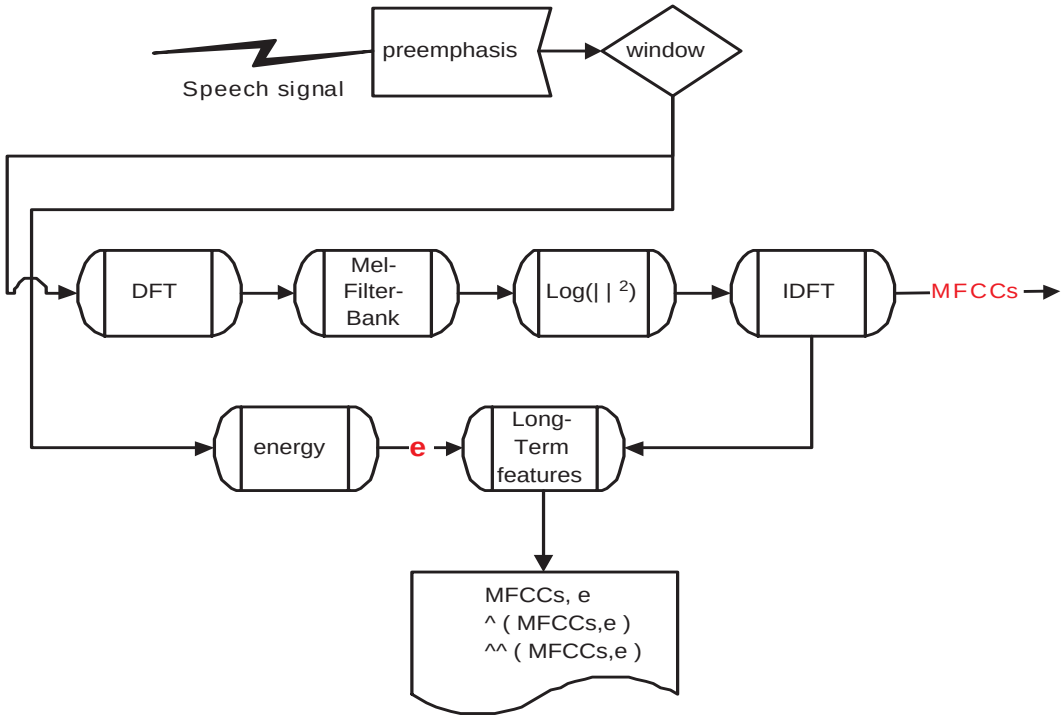

Figure 2.2: MFCC processing of the speech signal

2.2.1 Speech Signal Processing

Acoustic feature extraction follows the front-end processing of the signal. There can be different signal processing techniques for different feature extraction methods. The first stage of signal processing is converting the acoustic signal to its digital equivalent. In this conversion the sampling rate is chosen as at least two-times of the highest frequency harmonic in the signal by regarding the Nyquist criterion [9].

After the sampling, the signal is coded using a coding method like PCM(Pulse Code Modulation), Log-PCM, APCM(Adaptive PCM), DPCM(Differential PCM), ADPCM(Adapted Differential PCM) or DM(Delta Modulation) [9]. The coded signal is then processed by some additional processes to increase the distinguishability of the feature vectors. The main processes are the following:

- **Windowing:** Windowing is applied on the acoustic data which is used in parameter calculation during feature extraction. When the acoustic data is directly used in parameter calculation process, this means that rectangle windowing is chosen. In other windowing methods, the acoustic data is multiplied by certain coeffients before processing. The determination of these coefficients done with some windowing methods. While window functions such as Barlett, Triangular, Prolate Spheroidal, Blackman, Kaiser and Hanning appear in literature, the Hamming Window is the most widely used [9]. Also for this thesis the Hamming Windowing method has been used. This method is used to emphasize the central information of the signal and defined by the equation 2.2;

$$w(n) = 0.54 - 0.46 \cos \frac{2\pi n}{N-1}, 0 \leq n \leq N-1 \tag{2.2}$$

  Here, $w(n)$ denotes the impulse response and N is the period.

- **Preemphasis:** To cancel the spectral contributions of the larynx and the lips to the speech signal, the speech signal is preemphasised to increase the relative energy of the high-frequency spectrum. Typically a first order FIR filter,

$$H(z) = 1 - az^{-1} \tag{2.3}$$

  is used with $a = 0.97$. The minimum-phase component of the glottal signal which serves as the excitation for the vocal tract, can be modelled by a two-real-pole filter whose poles are near $z = 1$. By adding the preemphasis filter, the spectral contribution of the other pole is effectively canceled, eliminating the effect of the larynx and the lips on the speech signal [9].

- **DC Removal:** DC offset in a speech wave is typically an artifact of the recording process. This can easily be removed by subtracting the mean of the speech signal. If this is done in real time, a short time estimate of the mean must be used. This can

be done with a simple first oder FIR (finite-impulse response) or IIR (finite-impulse response) low pass filter [34].

- **Filtering:** Filtering is used to distinguish the data from the unwanted frequencies.
- **Zero-Crossing-Rates:** Zero-Crossing-Rates are used to observe frequency variations.
- **Center Clipping:** Center Clipping is a method used in noise cancelation. In this method it is assumed that noise signals have high frequencies and low amplitudes. With this assumption zero near signals are eliminated from the signal.

2.2.2 Speech Signal Feature Extraction

The purpose of the feature extraction stage is to derive a set of parameters from the speech signal suitable for the subsequent recognition stage. This set of parameters forms a feature vector. A feature vector is the smallest fundamental unit of the speech data and generally includes 15-39 features. There are two kinds of feature vectors, short-time and long-time feature vectors. Long-time feature vectors consist of the combination of many successive short-time feature vectors. There exists many possiblities to extract short-time feature vectors [2]. Commonly used ones are the following;

- Spectrum
- Cepstrum
- Linear Predictive Coding(LPC)
- Perceptual Linear Prediction(PLP)
- Relative Spectra(RASTA)
- RASTA+PLP

Spectrum

To use the spectrum content of the speech signal is one of the important methods in speech recognition. FFT is one of the efficient methods in spectrum calculation. Spectrum calculation is done by measuring the intensity of frequency in short time

periods. Thus in addition to FFT, it is possible to use a filterbank. The selected filter should reflect all characteristics of the sound data by a vector in an efficient way. Now, the best performance is obtained from Mel-Scale filterbanks. Mel-Scale filterbank consists of 100Hz length triangular filters in between 0-1000Hz. After 1kHz logarithmically increasing filters are used, as in Figure 2.3.

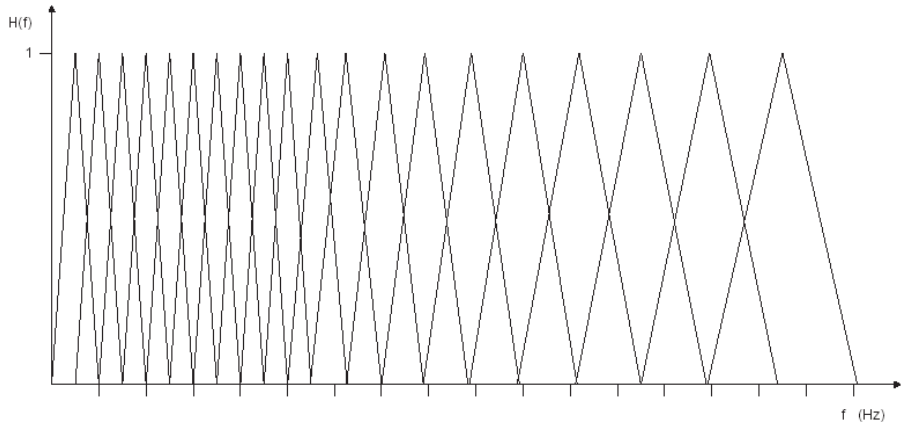

Figure 2.3: Mel-Scale filterbank for 8kHz signal

An approximate conversion to mel-scale is given in (2.4);

$$F_{mel} = \frac{1000}{\log 2}\left[1 + \frac{F_{Hz}}{1000}\right] \tag{2.4}$$

In addition to mel-scale also it is possible to build filterbanks by using bark-scale. Bark-scale conversion can be done by using (2.5);

$$F_{bark} = 13a\tan\left(\frac{0.76F_{Hz}}{1000}\right) + 3.5a\tan\left(\frac{F_{Hz}^2}{7500^2}\right) \tag{2.5}$$

Each unit in mel/bark scale is called as the critical bandwidth. To be able to build a filterbank with mel/bark scale the critical bandwidths should be known. Equation (2.6) helps to find these critical bandwidths.

$$BW_{critical} = 25 + 75[1 + 1.4(f/1000)^{2.069}] \tag{2.6}$$

The filterbanks constructed by mel/bark scale are called as critical band filterbanks.

The other method of obtaining spectrum is FFT Based Spectral Analysis. Its implementation is relatively easier than the previous method. FFT based spectral analysis bases on applying a Fourier transformation on the speech signal and selecting samples for some certain values. Furthermore to obtain better spectral intensity, weighted summation of each spectral value is calculated. During the calculation of this summation mel-scale is used for determining the neighbourings.

Cepstrum

In this method the speech signal is represented as a linear equeation; $S(f) = V(f).G(f)$. Here $S(f)$ denotes the acoustic data, $V(f)$ models the acoustic channel, namely larynx, and $G(f)$ models the original pure signal produced by vocal cords. Thus it becomes possible to cancel the effects of larynx on the pure sound signal. Cepstral coefficients are generally complex but most of the contemporary systems use real cepstrum coefficients. Computation of real cepstrum coefficients using Discrete Fourier Transform (DFT) is given in Figure 2.4.

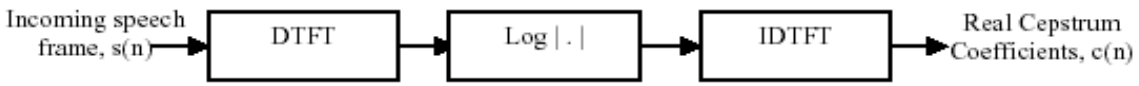

Figure 2.4: Computation of real cepstrum coefficients by DFT

Short-term cepstral coefficients computed either directly from FFT spectra or from LP(Linear Prediction) coefficients. In *Eq.* (2.7) FFT based calculation is formulated.

$$c(n) = \frac{1}{N_s} \sum_{k=0}^{N_s-1} \log_{10} |S(k)| e^{(2\pi/N_s)kn} \quad , 0 \leq n \leq N_s - 1 \tag{2.7}$$

Here, $S(k)$ denotes the Fourier transform of the frequency interval and N_s is the wide of the current frame. Also $c(n)$ is called as the n^{th} Cepstrum Coefficient. These coefficients are called as fourier transform based cepstral coefficients. Another set of cepstrum coefficients can also be computed from linear predictive (LP) parameters using the recursion given in *Eq.* (2.8) [4];

$$c_{LP}(0) = \ln \sigma^2 \qquad c_{LP}(i) = -a_{LP}(i) - \sum_{j=1}^{i-1} (\frac{i}{j}) c_{LP}(j) a_{LP}(i-j) \tag{2.8}$$

$$c_{LP}(0) = -a_{LP}(1) \qquad c_{LP}(i) = \sum_{j=1}^{i-1} (\frac{i}{j}) c_{LP}(j) a_{LP}(i-j)$$

At first LP modelling was the most popular method for spectral representation. But it has some drawbacks, such as having sharp peaks at formant frequencies, which degrades the performance.

In general first 20 cepstrum value are used in feature vectors. Since cepstrum values are calculated by non-linear equations it can be said that they are noise-tolerant, for noisy environment speech recognition applications this method is not suggested.

If the frequency used in Fourier transformation is sampled in mel scale then the obtained cepstrum coefficients are called as Mel-Cepstrum values. Furthermore to calculate mel-cepstrum values mel-scale filterbank can be used instead of Fourier transform [3].

$$melc(i) = \sum_{k=0}^{K} \log |Sk| \cos \left[i(k - \frac{1}{2}) \frac{\pi}{K} \right] \tag{2.9}$$

Linear Predictive Coding (LPC)

Linear predictive modelling is one of the most accurate analysis techniques and most of today's speech recognizers support LPC to parameterize the speech signal. The fundamental idea of LPC is that a speech sample can be obtained approximately as a linear combination of past speech samples. The distance between the current signal and the linearly predicted one is minimized by finding certain coefficients. They are called as LP coefficients. The effect of vocal tract (spectral shaping) is modelled by an all-pole filter. Generally an order (number of poles) of 8-10 is chosen as the filter order for telephone applications, where a sampling frequency of 8kHz is used and the filter coefficients are determined by minimizing the MSE(mean-squared error) between the speech samples and predicted ones [4]. The vocal tract model can be

given as in (2.10).

$$H(z) = \frac{S(z)}{X(z)} = \frac{1}{1 - \sum_{k=1}^{p} a_k z^{-k}} \tag{2.10}$$

Here X(z) is the z-transform of the excitation driving the source filter model of speech production, S(z) is the z-transform of the speech signal, a_ks are the linear predictor coefficients and p is the order of the LPC analysis. The linear estimation of the next speech sample can be calculated by using the weighted summation of the past samples as in (2.11).

$$\hat{s}_n = \sum_{i=1}^{p} a_i s_{n-i} \tag{2.11}$$

The transfer function of a lossless tube can be modelled by using all-pole model. Although in speech processing context larynx is assumed as a lossless tube, actually it is lossness and its shape can not be thought as a complete cylinder. Larynx contains some small holes. Also some consonants are formed with lips without any effect of vocal chords. Despite these drawbacks, there are some approaches to approximately estimate the actual parameters of the speech signal by using enough number LP coefficients.

Let's assume that the speech signal consists of N samples. The ultimate goal is estimating the a_i coefficients which produce the most accurate result. The accuracy is tested by finding the minimum-square error, (2.12).

$$e_n = s_n - \hat{s}_n = s_n - \sum_{i=1}^{p} a_i s_{n-i} \tag{2.12}$$

And the summation of the squares of errors is calculated by (2.13).

$$E = \sum_{n=0}^{N-1} e_n^2 = \sum_{n=0}^{N-1} \left(s_n - \sum_{k=1}^{p} a_k s_{n-k} \right)^2 \tag{2.13}$$

The minimum value of E can be found by making its derivative equal to zero as in

(2.14).

$$\frac{\partial E}{\partial a_j} = 0 = -\sum_{n=0}^{N-1}\left(2(s_n - \sum_{k=1}^{p} a_k s_{n-k})s_{n-j}\right) = -2\sum_{n=0}^{N-1} s_n s_{n-j} + 2\sum_{n=0}^{N-1}\sum_{k=1}^{p} a_k a_{k-j} s_{n-j} \tag{2.14}$$

If we arrange the 2.14, then we obtain (2.15).

$$\sum_{n=0}^{N-1} s_n s_{n-j} = \sum_{n=0}^{N-1}\sum_{k=1}^{p} a_k a_{k-j} s_{n-j} \tag{2.15}$$

This equation determines the LP coefficients, a_k, for the speech sample. Robinson et al. [5] propose three ways to solve this equation;

- Autocorrelation Method
- Covariance Method
- Lattice Method

For speech recognition generally autocorrelation method is preffered. LPC Cepstrum values are obtained from the direct use of predicted LP coefficients. The equation for LPC cepstrum values are given in (2.16).

$$c_k = a_k + \frac{1}{k}\sum_{i=1}^{k-1} i c_i a_{k-i} \tag{2.16}$$

Perceptual Linear Prediction (PLP)

PLP method is the combination of DFT(Discrete Forurier Transform) and LP methods. In LP method it is assummed that all sounds in all frequencies are equal. But this assumption does not reflect the actual behaviour of human ear. Under 800Hz hearing decreases in parallel with the frequency. Human ear is more sensitive for middle frequencies of the hearing range. One of the proposed solutions for this problem predicts the LP coefficients in mel-scale. Another method is finding power spectrum of speech signal before the application of LP technique [6] and PLP method is based on this approach.

Relative Spectra (RASTA)

The main idea of RASTA is modelling the environmental effects. This method is an effort to filter out distortions and noises caused by unexpected environmental factors so as to improve speech recognition performance. This method can be obtained by adding a noise modelling technique to the aforementioned PLP method. In RASTA it is assumed that human perception strictly depends on the previously heard sounds. In other words perception is dependent to the spectral difference between current sound and the past sound [9].

2.3 Acoustic Modelling and Training

Modelling of the acoustic data is generally performed in a statictical framework. An inventory of elemantary probabilistic models of basic linguistic units (in this work, triphones) is used to build word representations. Acoustic Models (AM) are stochastic models used with language models to find the correct transcription for the given acoustic data. There are mainly three methods used in acoustic modelling.

- Hidden Markov Models(HMM)
- Dynamic Time Warping(DTW)
- Artificial Neural Networks(ANN)

The most effcient and widely used method for the past decade is HMM method and also in this work HMMs are used for acoustic modelling.

2.3.1 Hidden Markov Models

In ASR, HMMs were first used by Baker and Jelinek in 1975 [11]. Rabiner gave a detailed application methodology of HMMs for ASR in 1989 [7]. The fundamental assumption of HMM is that the data samples can be well characterized as a parametric random process, and the parameters of the stochastic process can be estimated in a precise and well-defined framework.

An HMM model is defined by three parameters; $M = (A, B, \Pi)$. Letting $o \in O$

be a variable representing observations and $i, j \in X$ be variables representing model states, the model parameters can be represented as in (2.17);

$$\begin{aligned} A &\equiv \{a_{i,j} | i, j \in X\} \quad & transition\ \ probabilities \\ B &\equiv \{b_{i,j} | i, j \in X\} \quad & output\ \ distributionprobabilities \\ \Pi &\equiv \{\pi_i | i \in X\} \quad & initial\ \ probabilities \end{aligned} \tag{2.17}$$

and they can be defined as in (2.18);

$$\begin{aligned} a_{i,j} &\equiv p(X_t = j | X_{t-1} = i) \\ b_{i,j}(o) &\equiv p(O_t = o | X_{t-1} = i, X_t = j) \\ \pi_i &\equiv p(X_0 = i) \end{aligned} \tag{2.18}$$

A graphical representation of an HMM is given in Figure 2.5.

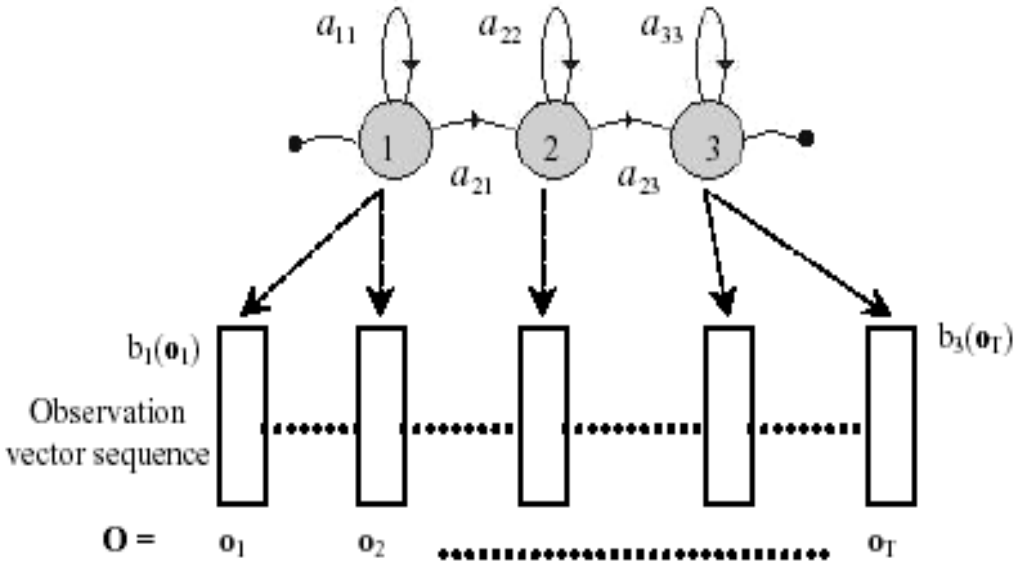

Figure 2.5: A three-state left to right HMM with observation vectors [8]

According to the definition set of the observation vector, two kinds of HMM structure can be defined; Discrete HMM (DHMM) and Continuous Density HMM (CDHMM). The difference betwwen the discrete and continuous HMM lies in a different form of output probability functions. In discrete HMMs, distributions are defined on finite

spaces. Observations, such as acoustic feature vectors, are vectors of symbols in a finite alphabet of N different elements. For each one of the vector components, a discrete density $w(k)/k = 1 \ldots N$ is defined, and the distribution is obtained by multiplying the probabilities of each component. In continuous density HMMs, distributions are defined as probability densities on continuous observation spaces. In this case, functional form of the distributions has to have certain characteristics, in order to have a manageable number of statistical parameters to estimate. The density functions are usually Gaussian or Laplacian as in (2.19). The statistics can be characterized by the mean vector and the covariance matrix.

$$b_j(o) = \sum_{k=1}^{M} c_{jk} N(o, \mu_{jk}, \Sigma_{jk}) = \sum_{k=1}^{M} c_{jk} b_{jk}(o) \tag{2.19}$$

Here $N(o; \mu; \Sigma)$ denotes a single Gaussian density function with mean vector μ , and covariance matrix Σ for state j, M denotes the number of mixture components and c_k is the weight of the k^{th} component. The Gaussian probability density function $N(o; \mu; \Sigma)$ is defined as in 2.20.

$$N(o; \mu; \Sigma) = \frac{1}{\sqrt{(2\pi)^n |\Sigma|}} e^{-\frac{1}{2}(o-\mu)\Sigma^{-1}(o-\mu)^t} \tag{2.20}$$

For speech recognition, continuous density HMMs are used since the extracted feature vectors are defined in an N-dimensional space. But in order to model complex distributions in this way, a rather large number of base densities have to be used in every mixture. This may require a very large training acoustic data for the estimation of the distribution parameters. Problems arising when the available data is not large enough can be alleviated by sharing distributions among transitions of different models. In semicontinuous HMMs, for example, all mixtures are expressed in terms of a common set of base densities. Different mixtures are characterized only by different weights. This method is called as tied-mixture modelling.

A common generalization of semicontinuous modeling consists of interpreting the input vector o as composed of several components $o[1] \cdots o[Q]$, each of which is associated with a different set of base distributions. The components are assumed

to be statistically independent, hence the distributions associated with model transitions are products of the component density functions.

Computation of probabilities with discrete models is faster than with continuous models, nevertheless it is possible to speed up the mixture densities computation by applying Vector Quantization (VQ) on the gaussians of the mixtures.

It is necessary to estimate accurate parameters $M = (A, B, \Pi)$ to describe the acoustic data effectively. Actually there is no analytical method that maximizes the joint probability of the acoustic data in a closed form. Instead the eproblem can be solved by the iterative Froward-Backward or Baum-Welch algorithm [9], a special case of the Expectation Maximization (EM) algorithm.

Before we describe the forward-backward algorithm, we first define a few useful terms. Consider the forward variable, $\alpha_t(i)$, defined as in 2.21.

$$\alpha_t(i) = P(o_1 o_2 \ldots o_t,\ x_t = i \,|M) \tag{2.21}$$

Here the $\alpha_t(i)$ is the partial observation sequence, $o_1 o_2 \ldots o_t$, and state i at time t , given the model M. $\alpha_t(i)$ can be found inductively as in 2.22.

$$\begin{aligned} \alpha_t(1) &= \pi_i b_i(o_1) \qquad 1 \le i \le N \\ \alpha_t(j) &= \left(\sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} \right) b_j(o_t) \qquad 2 \le t \le T \quad 1 \le j \le N \\ P(O|M) &= \sum_{i=1}^{N} \alpha_T(i) \end{aligned} \tag{2.22}$$

In a similar manner, the backward variable $\beta_t(i)$ can be defined as;

$$\beta_t(i) = P(o_{t+1} o_{t+2} \ldots o_T,\ x_t = i \,|M) \tag{2.23}$$

$\beta_t(i)$ is the probability of the partial observation sequence from $t+1$ to the end, given the state i at time t and the model M. $\beta_t(i)$ can be solved inductively by 2.24.

$$\begin{aligned} \beta_T(i) &= 1/N \qquad 1 \le i \le N \\ \beta_t(i) &= \left(\sum_{j=1}^{N} a_{ij} b_j(o_{t+1}) \beta_{t+1}(j) \right) \qquad t = T-1 \ldots 1 \quad 1 \le i \le N \end{aligned} \tag{2.24}$$

Now we define $\gamma_t(i,j)$, which is the probability of taking the transition from state i to state j at time t, given the model, M, and the observation sequence O.

$$\gamma_t(i,j) = \frac{P(x_{t-1}=i, x_t=j, O_1^T \,|M)}{P(O_1^T|M)} = \frac{\alpha_{t-1}(i) a_{ij} b_j(o_t) \beta_t(j)}{\sum_{k=1}^{N} \alpha_T(k)} \tag{2.25}$$

The equation above can be illustrated as shown in Figure 2.6.

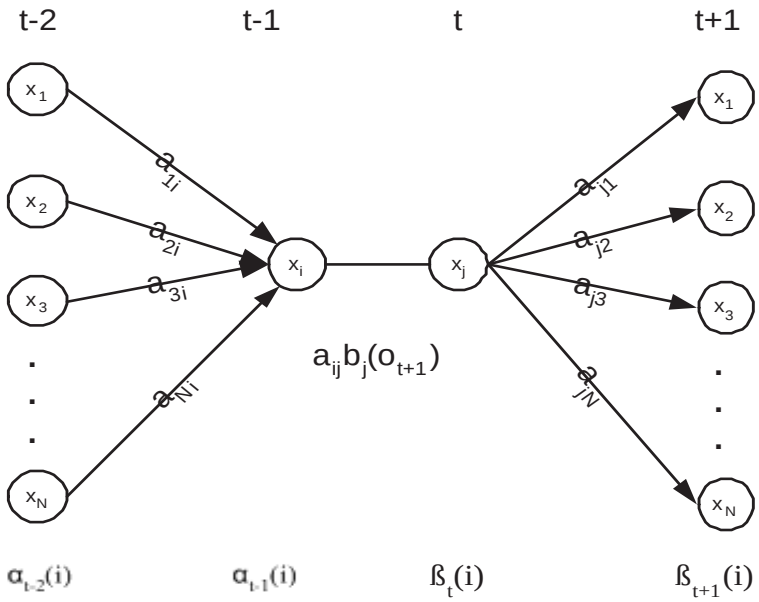

Figure 2.6: Illustration of the operations required for the computation of $\gamma_t(i,j)$

Lastly, we define the a posteriori variable $\delta_t(i)$ as the probability of being in state i at time t, given the observation sequence O and model M,

$$\delta_t(i) = P(x_t = i \,|\, O, M) = \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{N} \alpha_t(i)\beta_t(i)} = \sum_{j=1}^{N} \gamma_t(i,j) \tag{2.26}$$

Using this formation the estimated models can be defined as follow;

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T} \gamma_t(i,j)}{\sum_{t=1}^{T} \sum_{k=1}^{N} \gamma_t(i,k)} \tag{2.27}$$

$$\hat{b}_j(o_t) = \sum_{k=1}^{M} \hat{c}_{jk} N(o_t; \hat{\mu}_{jk}; \hat{\Sigma}_{jk}) \tag{2.28}$$

$$\hat{\mu}_{jk} = \frac{\sum_{t=1}^{T} \delta_t^k(j) o_t}{\sum_{t=1}^{T} \delta_t^k(j)} \tag{2.29}$$

$$\hat{\Sigma}_{jk} = \frac{\sum_{t=1}^{T} \delta_t^k(j)(o_t - \hat{\mu}_{jk})(o_t - \hat{\mu}_{jk})_t}{\sum_{t=1}^{T} \delta_t^k(j)} \tag{2.30}$$

$$\hat{c}_{jk} = \frac{\sum_{t=1}^{T} \delta_t^k(j)}{\sum_{t=1}^{T} \sum_{k=1}^{M} \delta_t^k(j)} \tag{2.31}$$

The forward-backward algorithm guarantees a monotonic likelihood improvement on each iteration, and eventually the likelihood converges to a local maximum. Although the forward-backward algorithm described above is based on one training observation sequence, it can easily be generalized to multiple training observation sequences. To train an HMM from N data sequences is equivalent to finding the HMM parameter vector M that maximizes the joint likeliood [9];

$$\prod_{i=1}^{N} P(O_i \mid M) \tag{2.32}$$

The training procedure performs the algorithm on each dependent observation sequence to calculate the expectations in Equations 2.27 and 2.28 [9].

Decoding of an HMM is the process of finding the best path, which generates the certain observation sequence, in the HMM framework of hidden states. The most widely used criterion is to find the state sequence that has the highest probability of being taken while generating the observation sequence. In other words, we want to find the most likely state sequence $X = x_1 \dots x_t$ for a given sequence of observations,

$O = o_1 \ldots o_t$, and a model $M = (A, B, \Pi)$, (2.33).

$$V_t(i) = \underset{x_1 \ldots x_{t-1}}{\arg\max} P(x_1 x_2 \ldots x_{t-1}, x_t = i, o_1 o_2 \ldots o_t | M) \tag{2.33}$$

This problem is very similar to the optimal-path problem in dynamic programming. As a consequence, a formal technique based on dynamic programing, known as Viterbi algorithm[10] can be used to find the best path for an HMM. Viterbi algorithm computes the observation likelihood of the total of the all paths through the HMM and searches for the optimal path simultaneously. In addition to Viterbi algorithm, there is another search method, A* Decoding or Stack Search [11] used in some speech recognizers. A* decoding algorithm is a kind of best-first search for the lattice or tree which implicitly defines the sequence of allowable words in a language. The algorithm performs a search from the root of the tree to the towards the leaves, looking for the highest probability path, and hence the highest probability sequence. In A* decoding, a path probability is defined as the product of its language model probability and its likelihood. It does this by keeping a priority queue of partial paths. A simple decoding process of this method is illustrated in Figure 2.7.

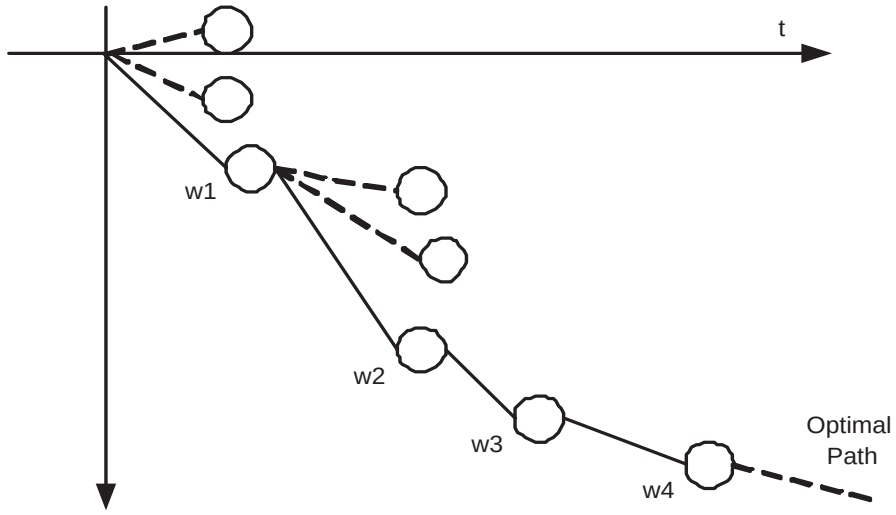

Figure 2.7: Illustration for the optimal path with stack decoder

2.3.2 Artificial Neural Nets

Neural Networks can be seen as a succesful modelling technique since they produce accurate recognition results with some of the most difficult phoneme identifications, such as stop consonant pairs, nasals and sonarants and fricatives [12]. The best performance has been obtained from Recursive or Time-Delay network approaches, which are well suited to ignoring the temporal variability that is undesirable in speech, while capturing the temporal features that mark prosodic distinctions within words [13]. Although neural network algorithms provides significant progress in phoneme-based recognition, they need further improvement to be as accurate as HMMs in modelling of speech signals.

In ANN approach there exist several layers and each layer includes several nodes (Each layer can have different number of nodes). The coded speech enters the network and the network is updated synchronuously at some time intervals, receiving new inputs and propagating information through the rest of the network. The output nodes represent output status, or response, which signals that a particular event has been recognized. Between the input and output nodes there are one or more hidden layers of nodes. Connections between nodes may go forward towards output nodes, sideways within hidden layers, or backwards, and can be recursive or have delays built in.

2.3.3 Dynamic Time-Warping

The two speech signal of an utterance produced by the same speaker can be quite different. The length of the utterance can distort in time non-linearly. DTW [14] is used to derive the overall distortion between two speech templates. In these template-based systems, each speech template consists of a sequence of speech vectors. The overall distortion measure is computed from the accumulated distance between two feature vectors that are aligned between two speech templates with minimal overall distortion. Indeed DTW is a special kind of Dynamic Programing Technique. Let's consider two array in time domain; $A = a_1, a_2, ..., a_M$ and $B = b_1, b_2, ..., b_N$. If we assume that the start time is same for both array, $a_1 = b_1$, then the aim is to overlap the end times, a_M and b_N. To make the and times equal

a funcion is defined by using dynamic programming techniques. And this function is applied on the sound data iteratively untill the boundaries are equal.

2.4 Speech Recognition Toolkits

There are toolkits available for building speech recognition systems. The most important,widely accepted and publicly available speech recognition toolkits are HTK and SPHINX. HMM is the common modelling structure for both systems but they differ in many details. In this work HTK has been chosen as the speech recognition toolkit.

HTK

HTK [8] has been developed in Cambridge University as an integrated suite of software tools for building and manipulating continuous density HMMs. It consists of a set of library modules and a set of more than 20 tools. It is written in ANSI C. Although HTK is general-purpose, it also encourages a particular approach to building speech recognition systems. Firstly, HTK is restricted to continuous density HMMs for the purpose of practical application believed to be more robust. Secondly, HTK provides a generalised mechanism which allows tying at all levels. Thirdly, HTK fosters an incremental approach to model building whereby a system of HMMs is refined through a number of stages involving interleaved model manipulation and model re-estimation. Lastly, it provides a rich set of integrated tools to manipulate a diverse range of data.

SPHINX

CMU Sphinx [15] is a large vocabulary, speaker independent speech recognition codebase and suite of tools. The code is available for download and use. Sphinx is implemented using a 5-state phonetic model; each phone model has exactly five states. At run-time, frames of the input audio are compared to the distributions in the states to see which ones the sound could have come from.

Chapter 3

Confidence Scoring

3.1 Overview

For many practical applications of speech recognition systems, it is desirable to have an estimate of confidence for each hypothesized word since the current speech recognition systems are far from perfect. With an accurate confidence measure for each recognized word, the language understanding and generating part can repair potential speech recognition errors, can reject out-of-vocabulary (OOV) words, and can identify key words (for word-spotting applications) that are relevant to the language understanding and generation. Also this information can be used in word selection for unsupervised adaptation schemes like MLLR to better match acoustic channel variations [16].

In Bayes' rule, Eq. (3.1), the denominator is the same for any sequence of recognized words, so that most practical speech recognition systems simply ignore $P(O)$.

$$P(W|O) = \frac{P(W)P(O|W)}{P(O)} \tag{3.1}$$

This means that the recogniser likelihoods are not absolute measures for the probability of O but relative measures with respect to different decodings of O. Equation 3.2 provides a solid framework for measuring confidence levels [9]. It is the ratio between the score for the word hypothesis, W, $P(W)P(O|W)$ and the acoustic probability sum, $\sum_{\hat{w}} P(\hat{w})P(O|\hat{w})$, (3.2), where the summation is over all possible word sequences.

$$P(W|O) = \frac{P(W)P(O|W)}{\sum_{\hat{w}} P(\hat{w})P(O|\hat{w})} \tag{3.2}$$

3.2 Confidence Features

Chase [17] proposes a framework for incorporating a confidence measure into a conversational dialog system:

- At what level should the confidence annotation be made?
- What is the right way to define what is a recognition error and what isn't?
- What features are useful and how useful?
- How to build a model combining the various features to create a confidence annotation?
- How to measure the goodness of the feature and model?

Although the answers to these questions depend on the particular application that incorporates the confidence annotator, in this section generic approaches for each question are presented.

The confidence annotations (error predictions) consist of labeling either *phones, words, or utterances* with probabilities that correspond to the likeliness that the *phone, word or utterance* in question belongs to each of the classes under consideration, namely correct or recognition error.

Mainly, there are three types of levels on which confidence features can be defined, phonetic level, word level, and utterance level. Phonetic level features are the raw acoustic scores produced by the recognizer's acoustic model at phonetic level. Because the raw acoustic scores are usually not particularly useful as confidence measures [18], these scores are normalized by some methods[19,20,21]. Utterance level features are the features which have been observed to provide information about the correctness of an utterance hypothesis and they are extracted for each utterance. And similiarly word level features are derived from each word to provide information about the correctness of a word hypothesis.

Another grouping on confidence measures/features can be done as to the information source; acoustic, language model (LM), N-best list, word lattice or semantic. All except the last one are based on the information derived from the decoder. The last one is extracted during language understanding or parsing.

All these features (in the remaining of the thesis, *features* refers to confidence features, not to speech signal features) unavoidably overlap in the information that they use. The performance achieved by all the features together isn't much better than that with only the best feature. So it is important to decide that which features should be used and where the thresholds should be placed, namely how to combine the knowledge sources into effective confidence measures. A simple gaussian-mixture-model (GMM) classifier can be employed to build up a post-classifier of correct and incorrect results. In some early works a neural network (NN) has been trained under the back propagation scheme as a useful means to combine different types of knowledge [23]. Also, a similar unification system can be obtained from linear discriminant analysis [24]. A recent technique, support-vector-machine (SVM) [22] has been shown as an effective post-classifier to find the best combination of features. Lastly, Zhang et al. [25] have proposed several systems that, with remarkable results, apply the decision trees to the generation of confidence measures. Although effective, these systems, including ours, exhibit a common drawback: their performance depend on the amount of samples given for their training and, therefore, they need databases of previous recognition examples to adjust their parameters and in consequence they become application-dependent. In this work, we experiment with two kinds of classifiers, GMM and ANN to observe the variations in performance with different classifiers.

P(O) in Bayes' equation (3.1) can be approximated by general-purpose recognizers. These kind of recognizers should be able to recognize anything, so that they can fill the holes of the grammar in the normal speech recognizer. One of the most widely used is all-phone network. All possible phonetic and nonspeech models are trained and connected in parallel to each other and a loop is added, thus any word

sequence can be recognized. Also, for word-spotting applications a network of antiword models trained with all the data that are not associated with the key words of interest can be used as a filler model. In this thesis, new kind of filler models are constructed and are used to derive robust confidence estimators in combinations with other extracted acoustic information.

Another method of determining confidence information is based on the impacts of different phones on human perception of words [9]. Then the confidence measure can be defined as in (3.3) for the word, w;

$$CS(w) = \sum_{i=1}^{N} \wp_i(x_i)/N \tag{3.3}$$

Where x_i is the recognition score of phone i and N denotes the number of frames for the word, w. $\wp(x)$ is the mapping function and it determines the weight for each phone score.

$$\wp_i(x) = a_i x + b_i \tag{3.4}$$

The parameters a_i and b_i can be optimized by using discriminative training.

The most common way of obtaining confidence information is using the combinations of features. In Appendix A a long-list of confidence features compiled from the related work in the literature is presented. The features are categorized according to their information source and the level of the usage. Information sources can be defined in five goups;

- **Acoustic Features:** Many confidence features focus on an examination of the scores produced by the recognizer's acoustic models at different levels. Because the raw acoustic scores are usually not particularly useful in confidence annotation [18]. So before the use, they are normalized by using various methods [19,20]. Normalizing the score against a filler model is one of the widely used techniques.

- **Languge Model Features:** The knowledge of the pragmatics of language (what people are likelly to say in particular contexts) can be important to achieving the goal of confidence annotation.

- **N-Best List Features:** The decoder can generate results until it finds N complete paths, namely sentences, and these N complete sentences form the N-best list. The fundamental idea is to maintain separate records for paths with distinct histories. The history is defined as the whole word sequence up to the current time t and word w. When two or more paths come to the same state at the same time, paths having the same history are merged and their probabilities are summed together; otherwise, only the N-best paths are retained for each state [9].

- **Lattice-Based Features:** The set of hypotheses considered by the recognizer is represented as a directed graph with exactly one start node and one end node. Each arc carries a word along with its log-likelihood score and each possible path in the graph denotes one hypothesis.

- **Semantic Features:** Features of the parse of the decoding hypothesis suggest themselves as a new information source for confidence annotation. The correct recognition results would be more grammatical than the incorrect recognition result, given the assumption that the speaker is cooperative.

Although there are three levels to define a feature; phone,word, and utterance, while grouping the fetaures at the phone level is not considered as a separate level since its impact is realized on word level. So they are included in word-level features. (In the list level categorisation is shown by W, refers to word-level, and U, refers to utterance-level.)

3.3 Feature Selection and Classification

A classifier can be viewed as a network or machine that computes many discriminant functions and selects the category corresponding to the largest discriminant [26]. A network representation of a classifier is illustrated in Figure 3.1.

In this representation each $g_i(x)$ is called as discriminant function and gives the probability of observation belonging to the class i. As to the Figure 3.1, the choice of discriminant functions is not unique, all the functions are multiplied by some constants. But for most cases in speech the costs are chosen equal.

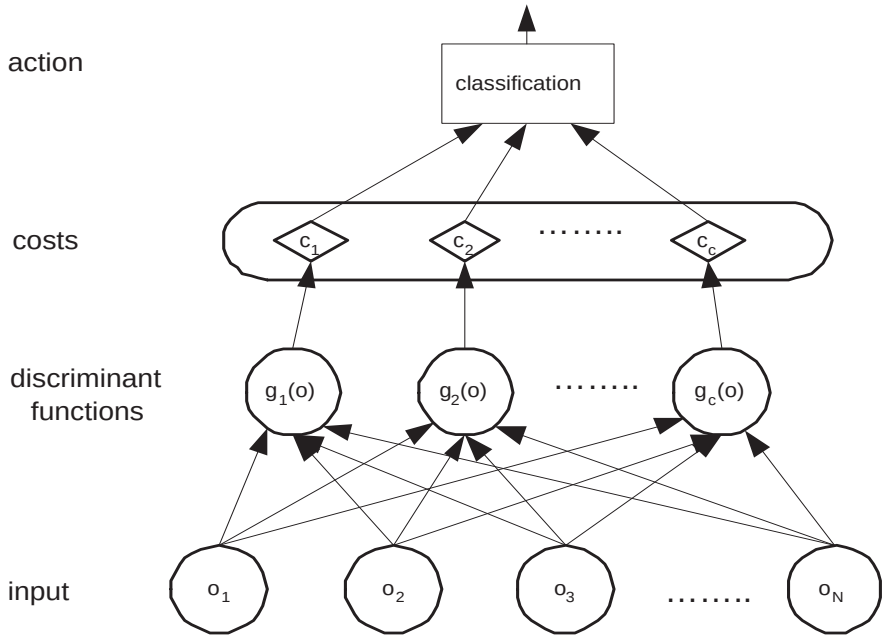

Figure 3.1: Functional structure of a general statistical classifier[26]

For two class case, the decision is taken according to the comparison of the results of discriminant functions as in Eq. (3.5). Equation (3.5) is also called as Bayesian Decision Rule.

$$g_i(o) = P(h_i|o) = \frac{P(o|h_i)P(h_i)}{\sum_{j=1}^{c} P(o|h_j)P(h_j)} \cong P(o|h_i)P(h_i) \quad (3.5)$$

Several kinds of classifiers can be used to compute the confidence scores from a set of features. The widely used ones in literature are classifiers based on GMMs, Neural Networks (NN), Linear Discriminant Analysis and Decision Trees. Previous research has shown that although in general the difference between classifiers is insignificant, for some specific applications certain classifiers improve the performance much more than the others. We have used GMM and NN classifiers for our experiments.

3.3.1 GMM-Based Classifier

A Gaussian classifier is a Bayes' classifier where conditional probability density function $P(x|h_i)$ for each class, h_i, is determined by Gaussian distributions. Each Gaussian density function, $p(x)$, also called as Gaussian component, is determined by three variables, c_k the weight of the component, μ, the mean and σ^2, the variance.

A gaussian density function is represented by the formula in (3.6).

$$p(x) = \frac{1}{\sqrt{2\pi\sigma}} exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right] \tag{3.6}$$

$$\mu \equiv \int_{-\infty}^{\infty} xp(x)dx \tag{3.7}$$

$$\sigma^2 \equiv \int_{-\infty}^{\infty} (x-\mu)^2 p(x)dx \tag{3.8}$$

In a GMM classifier, the discriminant function of the class is calculated as the weighted summation of gaussian density functions, (3.9).

$$P(x|h=i) = \sum_{k=1}^{M} c_k(i) N(x, \mu_k(i), \Sigma_k(i)) \tag{3.9}$$

In our experiments we have used a GMM-based classifier. (The classifier was used to decide on the input vector of confidence features, maximum 12 features.) Also all covariance matrices have been assumed as diagonal matrices. The number of components for each class have been determined by a set of experiments with different component combinations. Maximum-likelihood-estimation (MLE) method [8] has been used for the training of the GMM parameters.

3.3.2 Neural-Net Classifier

In confidence measure calculation, it is popular to combine different features with multi-layer neural networks. In Figure 3.2 a two-layer perceptron classifer is illustrated.

There are works that conduct research on determining the best neural network topology for the discrimination of confidence features [27] and they report us that there is no significant difference between simple topologies and more complex topologies.

In Figure 3.2, the inputs of the network are the confidence features relative to a word. The hidden layers can be viewed as feature extractors and it is not requried to have the same number of perceptrons in each layer. The value of each

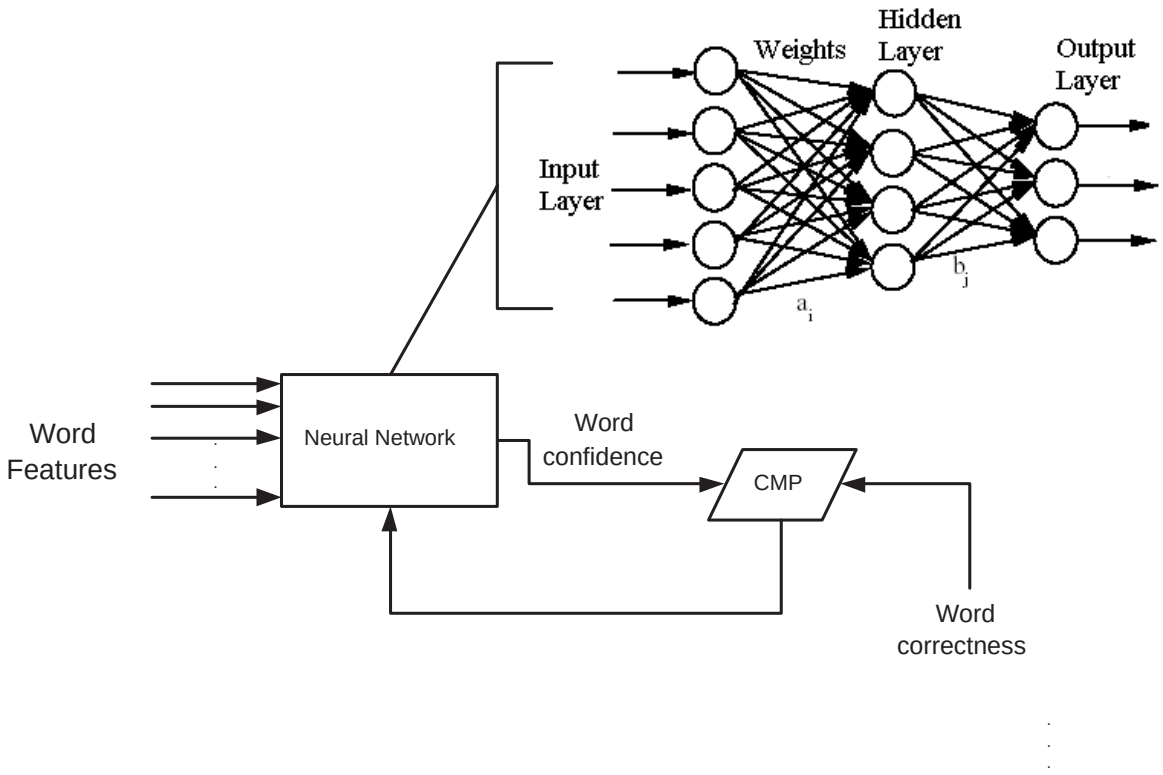

Figure 3.2: Illustration of a Neural Net classifier

node/perceptron in the network is computed as a linear weighted sum of the input nodes and passed to a sigmoid type of threshold function. In Equation (3.10) the computational model for each level in the network is formulated.

$$h_l = sigmoid(g_{hl}(h_{l-1})) \tag{3.10}$$

$$sigmoid(x) = \frac{1}{1+e^{-x}}$$

$$g_j(x) = w_{0j} + \sum_{i=1}^{N} w_{ij}x_i$$

Here, N is the number of inputs labeled as $x_1, x_2, ..., x_N$ and $w_{0j}, w_{1j}, ..., w_{Nj}$ denotes the weights for the class j.

3.4 Performance Evaluation Methods

Although there is no reliable way of measuring and comparing performance of confidence annotation systems, since the performance of such systems depends on the

application specific parameters like the quality of the data, in literature various methods for the evaluation of confidence measuring systems have been proposed.

A confidence measure is used to determine whether a recognized word or sequence of words should be trusted or rejected. Since words are accepted or rejected according to a threshold, ranks are used. The quality of the confidence measure resides in its ability to distinguish correct and wrong words [42]. Here, two different types of errors can occur, Flase-Acceptance and False-Rejection. In a false-accept case, a false word is tagged as correct. Similiarly, in a false-reject case, a correct word is tagged as false. There is a trade-off between the two types of errors, and this trade-off depends on the choice of the tagging threshold.

$$FR\,Rate = \frac{Number\ of\ false\ assigned\ tags\ for\ class\ CORRECT}{Number\ of\ total\ tags\ for\ class\ CORRECT} \tag{3.11}$$

$$FA\,Rate = \frac{Number\ of\ false\ assigned\ tags\ for\ class\ FALSE}{Number\ of\ total\ tags\ for\ class\ FALSE} \tag{3.12}$$

The simplest measure is the confidence error rate (CER). It is the number of correctly tagged words divided by the total number of words as seen in Equation (3.13). The baseline for CER is the ratio of the total number of recognition errors (substitutions and insertions) and the number of recognized words. The main drawback of the CER is that it depends on the prior probability of the two classes correct and incorrect [43].

$$CER = \frac{Number\ of\ correctly\ tagged\ words}{Number\ of\ words} \tag{3.13}$$

Another criterion is the equal-error-rate (EER). It is the operating point where the False-Accept (FA) rate is equal to the False-Reject (FR) rate. It is the point of the reciever operating characteristic (ROC) curve closest to the origin of the axes. The ROC curves are used to represent the performance of a confidence measure. (The ROC curve intersects the axes of FA and FR and plots the EER over decision threshold.) In this work to evaluate the confidence measures, EER criterion was used, since

it contains a high amount of information for different operating points of the system.

The normalized cross entropy (NCE) is another method and it is defined as the relative decrease in uncertainty brought by the confidence measure about the status of the words (correct or false). For a detailed explanition of the criterion, the reader is referred to [27,42].

These are the most common methods to evaluate confidence measures. The other methods can be listed as:

- NMCE, a variation of NCE [42]
- Mean Square Error (MSE) [23]
- Normalized NCE [23]
- Net recognition performance (NERP) [23]

Chapter 4

Experimental Setup

In this chapter, we present a detailed description of the dialogue system at which the performance of the investigated confidence vectors are evaluated.

4.1 Task Description

The dialogue system developed in the thesis responds to enquiries about courses. To increase the recognition performance and overall robustness of the dialogue system we defined a restricted word network, grammar, for both the speech understanding and speech generation parts. Although regular grammar was used, we tried to cover all possible question types in this domain. A detailed explanation of the grammar is presented in Section 4.3.

In Figure 4.1 a sample scenario defined in this handcrafted grammar is illustrated.

Main components of a typical dialogue system had been introduced in Section 1.4. In a typical system, each task is managed by separate sub-systems like, speech recognizer, speech synthesizer, language generator, language understanding component, audio server, dialogue manager, and all these sub-systems are connected to each other by a hub script. For our system, we have used this flexible design scheme. In the remaining of this chapter, details for each component in the main system are presented.

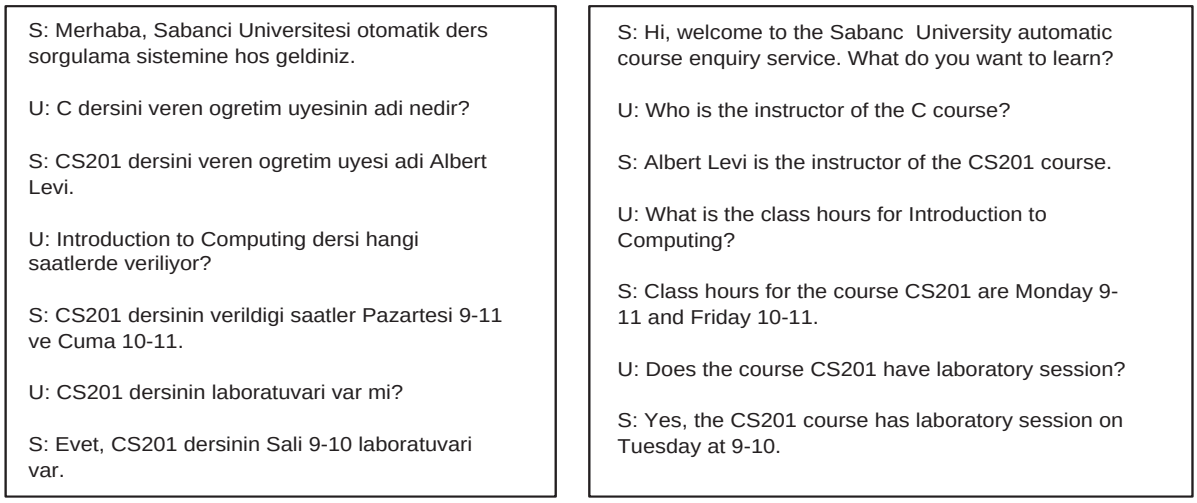

Figure 4.1: A scenario defined on the task grammar. S denotes the system and U refers to the user.

4.2 Data Collection

To train the acoustic models we have used two separate data-sets. Since research on ASR for Turkish is relatively new, there is no common speech database like TIMIT or SWITCHBOARD for Turkish yet. But in a study by Ümit Yapanel [4] about an hour telephony speech data, called TurTel, was collected for Turkish. It is a general-purpose database and based on statistical modelling of Turkish triphones. In our experiments we used this database and additionally, since the content of our domain is a bit far from the scope of the generic model, we needed to collect a new set of speech data. But at the end, we used both data to train our acoustic models.

4.2.1 TurTel

Turtel is a general purpose Turkish speech database collected over the public telephone network. Its corpus is based on statistical triphone modelling. It assumes that %80 of Turkish languge can be represented by 1000 triphones. Table 4.1 presents the language coverage rates of some triphone sets. Another assumption in the database is that the pronunciation of words in continuous speech do not dramatically differ from isolated ones. So the mentioned 1000 triphones are captured using 15-sentence and 343-isolated-word corpus.

The design structure of the database collection system for TurTel is given in Figure

Table 4.1: Number of triphones included vs. covered language

Num. of Triphones	**Coverage of the Lang.**
100	31.80
500	64.44
1000	80.88
1500	87.80
2000	92.15
3000	96.66
4000	98.51
5000	99.33

4.2.

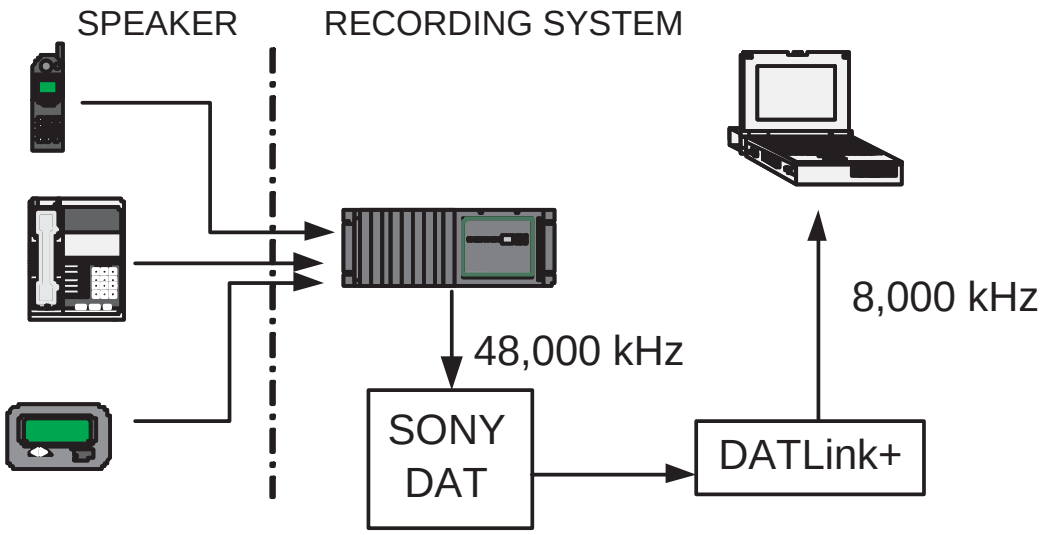

Figure 4.2: Data collection structure of Turtel

The database includes speech of people from many different ages. The speaker set consists of 57 male and 36 female from different age groups and origins in Turkey. Speaker age distributions of the database is given in Figure 4.3.

To provide an even distribution of different microphone types, three different tele-

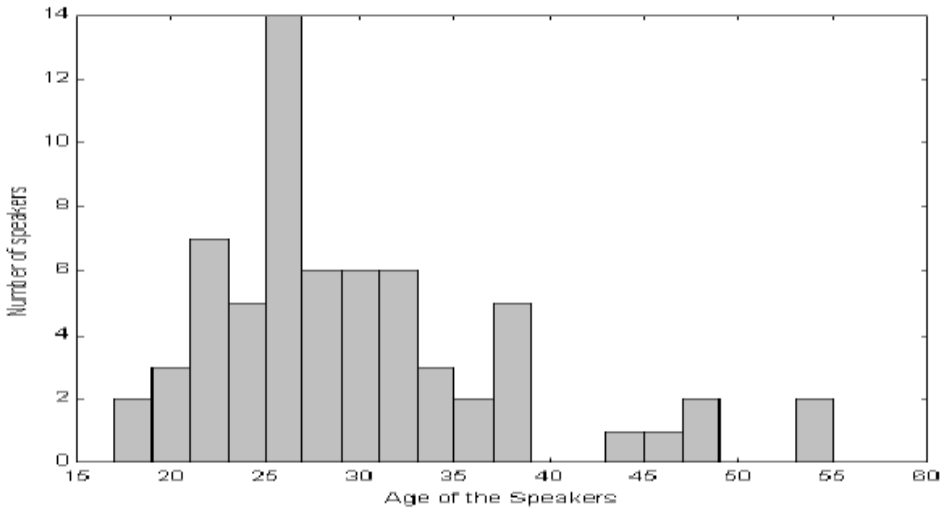

Figure 4.3: Age structure of the speakers in TurTel

phone machines, normal local phones, hands-free phones and cellular phones have been used. Each of these phones have different type of microphones with considerably different specifications. In Table 4.2, the distribution of telephone types in TurTel is illustrated.

Table 4.2: Telephone type distribution of TurTel

Gender	**Normal**	**Hands-Free**	**Cellular**
Male	25	8	7
Female	12	6	7

4.2.2 Collected Data

After an analysis we determined that about 500 different triphones used in the task grammar are not included by TurTel corpus. So we have decided to collect a set of separate data in addition to TurTel. We collected a 3-hour speech data specific to our system. First we have constructed a 1000-sentence corpus by using our word network. Then we have separated the corpus into 10 different sentence groups. Each group was read by a different speaker. We have used 45 speakers, 31 male and 14 female. Since the dialogue system is designed for use of students, all speakers are selected from students in different levels, undergraduate, masters and Phd. In our

campus all students have the same kind of telephone machine, so that in our recordings we have used only a such kind of telephone microphone.

From the 45 speakers, 4500 different sentences were recorded. %50 of the database was used for training the acoustic models. For confidence vector classifier training %30 of the data was used and the remaining %20 was reserved for overall system test. In addition to this 4500 sentence database, we also collected another database of 2000 sentences from only two speakers to use in other projects.

4.2.3 Construction of the Corpus

At first we have tried to train our acoustic models by using the TurTel database. After the training we have observed that this database was insufficient to model our specific domain. This was expected because it is quite probable for TurTel not to cover many triphones from our grammar. The corpus of TurTel had been constructed after a statistical modelling study on some novels and a newspaper, so that it mostly covers daily words and context dependent phones. But in our case there are lots of different context-dependent phones because the grammar includes many course codes, English pronounciations, and other similar exceptional cases. For example we need models of such triphones, *n-t+r, y-z, s-i+e*, that are too rare to be seen in a generic Turkish text corpus. The abundance of such kind of unseen triphones forced us to build our own corpus and speech database.

First we produced 100,000 random sentences from our grammar. Then we applied a statistical method to cover as many triphones as possible using 1000 sentences. The Maximum Entropy criteria was used in the selection of these sentences. The concept, Entropy describes the quantity of information and is defined as the amount of information after seeing an event.

Suppose S denotes a sentence from the set of 100,000 sentences and $X = x_1, x_2 \ldots x_n$ denotes the set of uncovered triphones. Then the entropy $H(X)$ of this random vari-

able S can be defined by the equation in (4.1).

$$H(S) = \sum_{x_i \in S} P(x_i) \log \frac{1}{P(x_i)} \tag{4.1}$$

Here the entropy $H(S)$ is the amount of information required to specify what kind of triphone has occurred on average and $P(x_i)$ is the probablity for that symbol x_i.

At the end we have determined 1000 sentences to cover the remaining of the triphones, not included in TurTel. After that we separated these sentences into 10 equal sets since it is not suitable to give all 1000 sentences to only one speaker for reading. During data collection each 100-sentence group was used in equal amounts.

4.2.4 System Setup

After the determination of training corpus we built a data collection system. To create equivalant training conditions with the real use environment we have designed our data collection system to work over telephone lines without using any other interfaces like tape recorders or filters. For the system we have used Intel Dialogic telephone cards and a GUI was designed for easy use. Also by considering future needs we have designed the system to work in background and to be available to collect data without an administrator. For each speaker, we assigned a user-id and a sentence-group number. The speaker invokes the application by calling the system over public telephone systems. Then the system prompts the instructions for robust recording and after from this prompt the speaker begins to record by dialing his id and sentence set-id. The overall algorithm of the system is given in Figure 4.4.

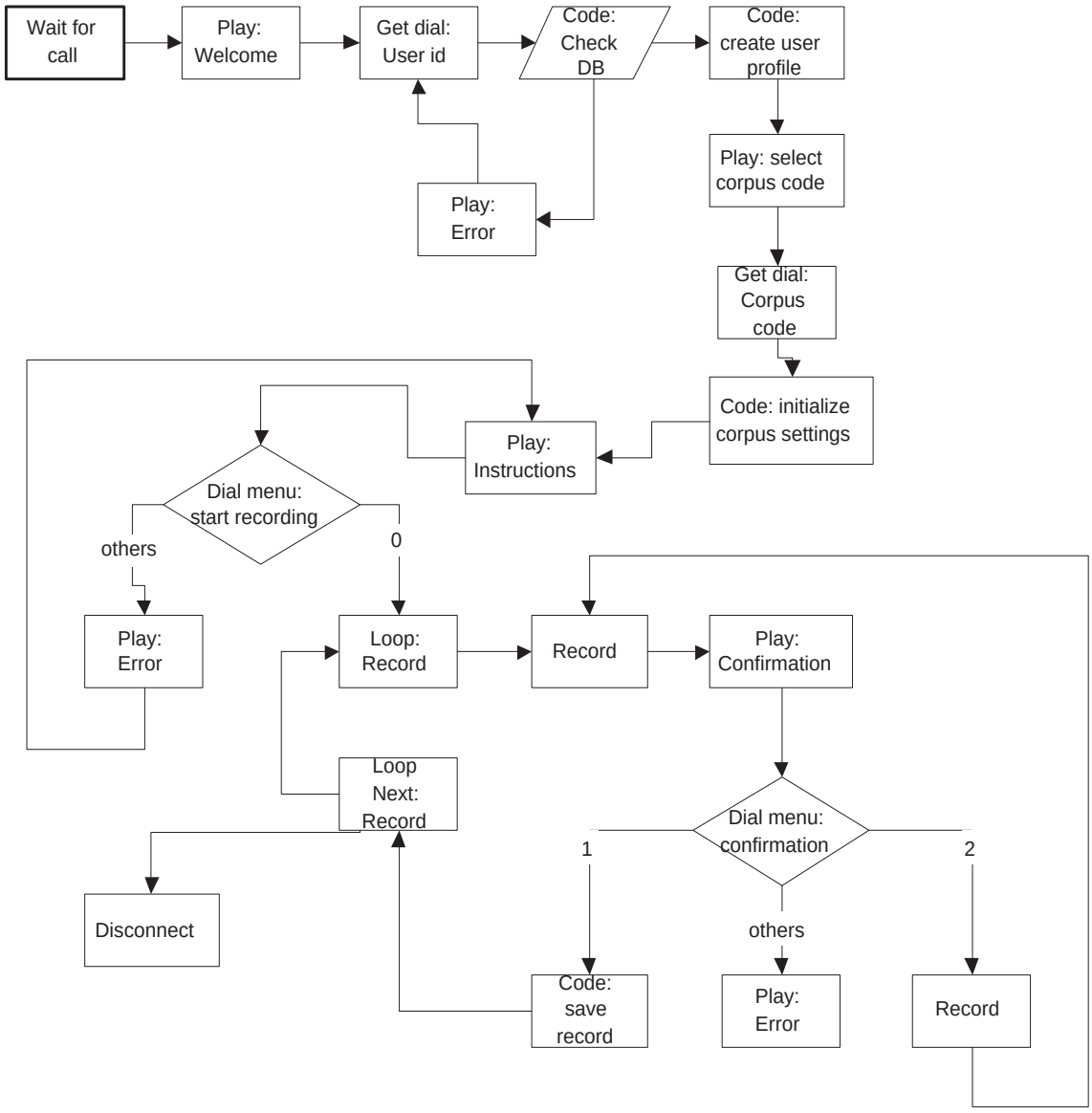

Figure 4.4: Flow diagram of the data collection system, implemented in Dialogic CT-ADE environment

All speech was recorded in 8kHz sampling rate with 8 bit resolution in mono quality. By a post-processing procedure all recordings have been checked and the erroneous parts were removed from the database.

4.3 Training Issues

In this study, triphones were chosen as sub-word units for modelling the acoustic data. Because triphone modeling considers both left and right context, it models the most important coarticulation, namely the transitions at the beginning and end of each phone.

Mel-Frequency Cepstral Coefficients (MFCCs), introduced in Section 2.1.3, are the parameterisation choice for this application. The parameter set for each frame contains 39 features; C0 as the energy component, 12 MFCCs, 13 delta coefficients and 13 acceleration coefficients as in Figure.4.5. Details about the extraction of these coeffcients has been presented in section 2.1.

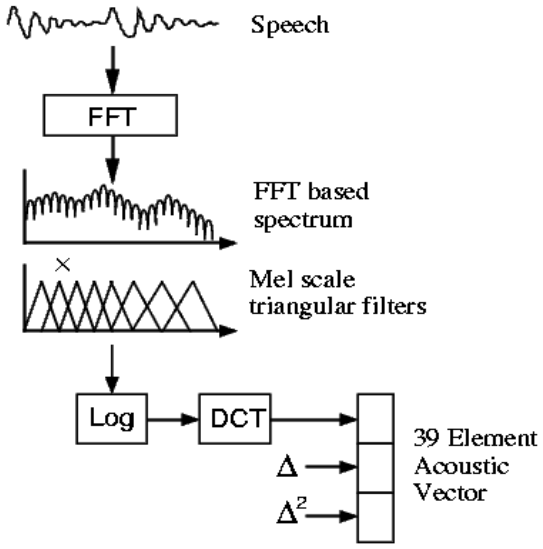

Figure 4.5: Feature Extraction by using MFCCs

All triphones were modelled by using 5-state (with 3 emitting states) and 5-mixture semi-continuous HMMs. Gaussian distribution function was used to define the distribution of each mixture. In Figure 4.6 the state topology of a triphone HMM is

illustrated.

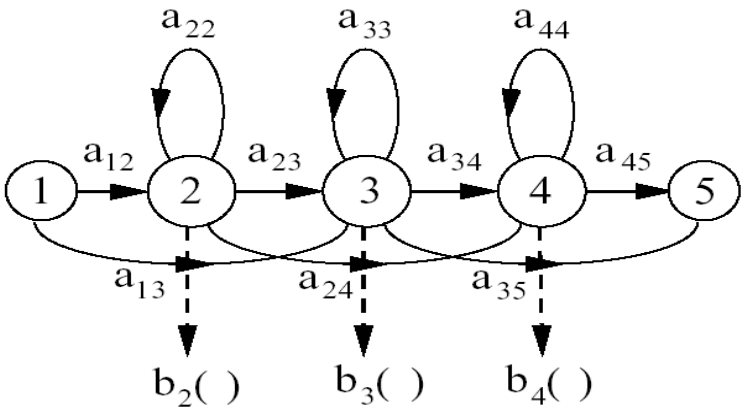

Figure 4.6: 5-state left-to-right HMM model topology

One of the important issues in speech recognition with HMMs is training. Here, Baum-Welch re-estimation has been adopted for training of sub-word HMMs. But before training a task-grammar, to be used in decoding, and a pronounciation dictionary were defined. The task grammar is constructed for user convenience. Actually the recogniser requires a word network to be defined using low-level representations in which each word instance and each transition is listed explicitly. And this low-level representation can be easily derived from a high level representation automatically. Below the details about the task grammar and pronounciation dictionary are presented.

4.3.1 Task Grammar

For our application, inputs to the system are generally questions related with courses. So in the grammar we have tried to cover all types of information requests. Mainly 7 types of questions were covered and for each question type, we have defined various styles of asking. Figure 4.7 provides some idea about the design scheme of the grammar in some extent.

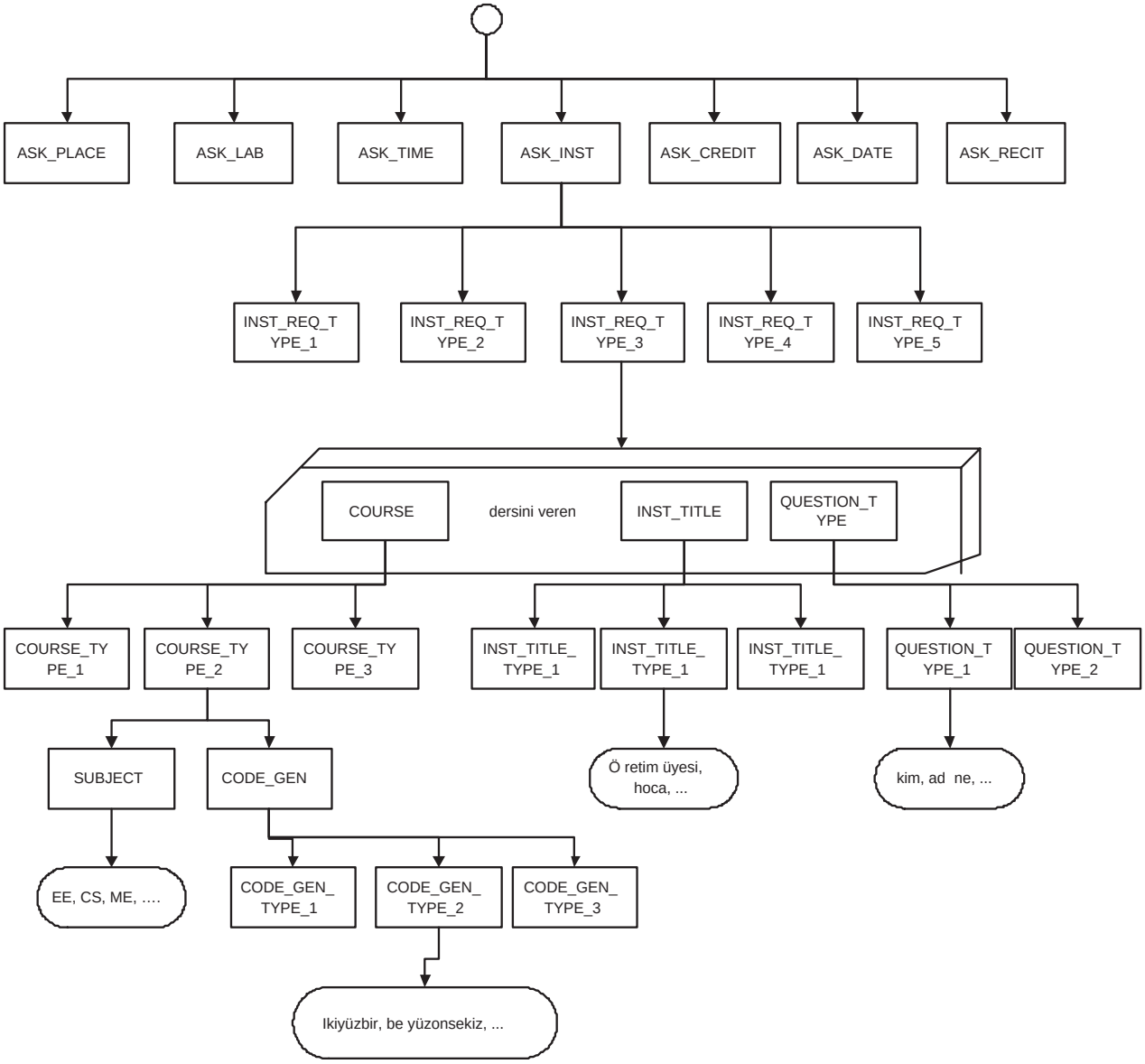

Figure 4.7: Representation of the task grammar, in some extent.

While designing the grammar, we had to consider some task-specific problems. The most important one is the pronounciation of course names. Between students it is common to describe a course by using its code like *CS201*, its formal name like *Introdcution to Computing*, its Turkish formal name like *Bilgisayara Giris* or its possible slang names like *C dersi* or *Bilgisayar* or *C++*. Another problem was the pronounciation of the course codes. The most popular way to pronounce a course code is saying the department-info (ie. CS) by using its English pronounciation and saying the number part by Turkish pronounciation. In addition to this also there are people who pronounce the code by using only the Turkish pronounciation. The variability of the pronounciation of the number part in the course codes is another significant problem in grammar design. For example there are many ways to pronounce such three-numeral numbers like 201 as illustrated in Figure 4.8.

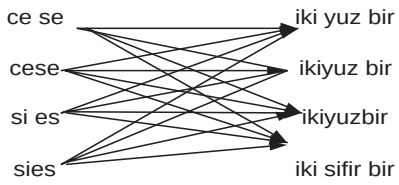

Figure 4.8: 4*4=16 possible pronounciations of CS201 in Turkish. Also the network can be more complicated for the numbers which don't include zero, like 211.

By considering all these problems we have prepared a grammar by using HTK grammar definition languge. This simple language provides the ability of defining variables and writing regular expressions on these variables. After from the construction of the grammar via an HTK tool, the grammar is converted into a word network for use in training and recognition.

4.3.2 Pronounciation Dictionary

Pronounciation Dictionary is a generic database of words together with their phonetic transcriptions. It is used in finding matches between phone sequences and words. The first step in building a dictionary is to create a sorted list of the required words. We have created this list from the sentences used in the training via a simple script. Finding the phonetic transcriptions for each word can be hard for

some languages but for Turkish it is not. Because Turkish is a phonetic language, each phone is read as it is written. For languages that are not phonetic, either each word is processed manually or some rules, if it is possible, should be extracted to find the correct phone transcriptions.

Although we use context-dependent phones to model acoustic data, during the construction of the pronounciation database we use context-independent phone units. The main idea of this behaviour underlies the philosophy of system construction. In HTK all HMMs are refined incrementally. Thus, a typical progression is to start with a simple set of single Gaussian context-independent phone models and then iteratively refine them by expanding them to include context-dependency and use multiple mixture component Gaussian distributions.

4.3.3 Parameter Tying

The single biggest problem in building context-dependent HMM systems is always the insufficiency of the training data. The more complex the model set, the more data is needed to have robust estimates of its parameters, and since data is usually limited, a balance must be struck between complexity and the amount of available data. For continuous density systems, this balance is achieved by tying parameters together. For parameter tying, HTK provides two methods, Data-Driven clustering and Tree-Based clustering. The main idea of data-driven clustering is tying equivalent states across different models [8]. For example one way to reduce the total number of parameters without significantly altering the models' ability to represent the different contextual effects might be to tie all of the centre states across all models derived from the same monophone as in Figure 4.9.

The limitation of data-driven is that it does not deal with triphones which were not seen in the training data.

All data corresponding to a monophone is gathered together and a tree is constructed where at each node we ask questions about left and right contexts. The best question that reduces the entropy is chosen at each node. This provides a

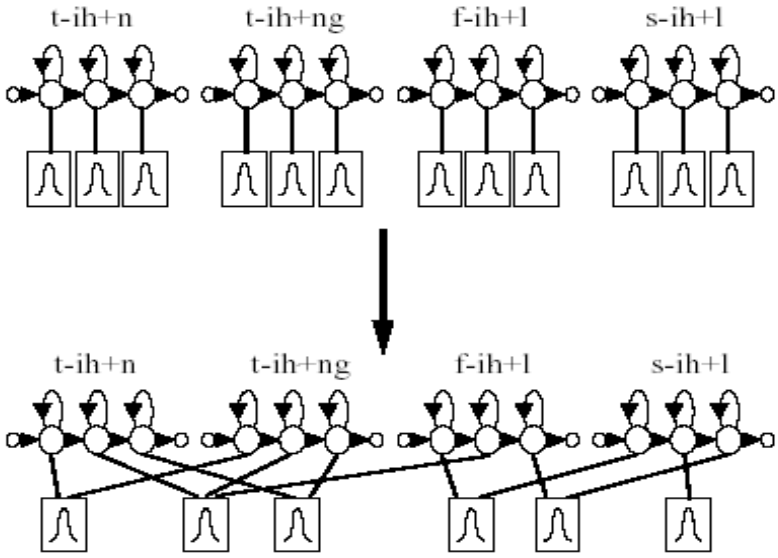

Figure 4.9: Data-Driven clustering of triphone model states

natural clustering of triphones. Tree-based clustering can provide a better quality of clustering if an efficient set of questions can be defined. For English, qualified questions had been defined to cluster triphones. Also for Turkish there are works about clustering of phones [28]. By regarding these works we have derived a set of triphone-based rules for decision tree clustering in Turkish. The proposed question set is given in Appendix B. To measure the quality of the rules we have made two experiments. In the first one, called *exp1*, we have used a basic set of questions during clustering. Then we have improved the question set and made an other experiment, called *exp2* by using this proposed question set. In Table 4.3 the improvement in the recognition is illustrated.

Table 4.3: Recognition results obtained from exp1,for which a basic set of questions was used, and exp2, for which the proposed set of questions was used.

Recognition Type	**exp1**	**exp2**
Utterance Recognition Rate	% 82	% 91
Word Recognition Rate	% 91	% 97

Table 4.3 shows the quality of the proposed question set and it can be used for any kind of speech recognition application in Turkish.

4.4 Filler Models and Confidence Measuring

One of the most important components of a dialogue management system is Confidence Server and for this thesis it forms the main interested area. Since recognition errors can not yet be avoided, it alternatively becomes important for a system to be able to detect when recognition errors have occurred and take appropriate actions to recover from these errors. For example, suppose that the system received such a question;

EE662 dersini veren öğretim üyesinin adı nedir?

(What is the name of the instructor for the course EE662?)

And again let's suppose that *EE662* is not included by the task grammar. Although the word *EE662* is not in the grammar, the recognizer produces its best guess for this word and it hypothesizes a sentence;

CS201 dersini veren öğretim üyesinin adı nedir?

(What is the name of the instructor for the course CS201?)

After passing from the confidence server, the course code *CS201* would probably be tagged as incorrect. Here, the language understanding component would need to be able to determine that for which course the user asks the instructor name. To make clear the missing information it sends a message to the Language Generator to prompt the user by asking a targetted question, like;

Hangi ders için öğretim üyesi ismini öğrenmek istiyorsunuz?

(For which course do you request the instructor name?)

As to the answer the Language Understanding component searchs for the database and sends the related information to the Language Generator to return the requested information.

The most common way to determine the confidence of a hypothesis is extracting some features either during recognition or during a post-recognition process like filler model recognition. By using such features a confidence classifier is trained. In this thesis 12 word-level candidate confidence measures were extracted for each hypothesis. Most of the features are derived from a number of different filler models. In this thesis, performances of different *filler-mode-based* confidence features were investigated. To determine a reliable decision boundary, a classifier was trained .

We have tried two different classifier types, GMM-based classifier and ANN classifier, to observe the effects of using different type of classifiers. In the following chapter, more detailed information about the used features, the filler models and the classifier design issues is presented.

4.5 Testing Issues

In our experiments we used a set of 1080 utterances from 12 speakers and 10 of them were not used neither in recognizer training nor in classifier training. All sentences in this data were recorded over public telephone lines in 8kHz and none of them was an out-of-grammar request.

4.6 Issues about the Other Dialogue Components

As audio server, we used the Dialogic hardware environment. This interface card receives the calls and records the requests. Also after from the speech synthesis this card plays the answer to the caller. Dialogic hardware set includes a function library written in C language. So that it can be easily managed by the Hub script.

The hub script of the system, coded in C++, connects the other components. Mostly it consists of system calls to invoke the other components.

Although there is heavy research in language generation and various complex techniques have been proposed, since our grammar is not so complex, we did not need to build such complex methods for language generation. Instead we used a predefined set of sentence prototypes and according to the request we search for the best prototype. After the determination of the best prototype, we adapt the prototype sentence as to the answer and we send it to the speech synthesizer.

For speech synthesis we used a pre-implemented TTS system by Vural [32]. This system uses the Festival speech recognition toolkit in the background.

Simply, semantic parser of the system searchs for some contextual clues in the re-

ceived request and tagged each recognized word with a semantic representation.

Dialogue manager receives semantic tags from the semantic parser and confidence information from the confidence server. Then it evaluates the recognition accuracy of the input utterance and it decides on the appropriate action. If there are misrecognized words the action would be requesting this misrecognized information from the user. But if the user request was understood then it searches the course database for the requested information and sends this information to the language generation component to produce the reply.

Chapter 5

Filler Models

5.1 Overview

As mentioned before, in theory, the confidence information for an hypothesis can be computed from the denominator part, $P(O)$, of the fundamental formula of Bayes' theory (3.1).

$$P(O) = \sum_{U=u_1 \cdots u_M} P(O|U)P(U) \approx \arg\max_U P(O|U)P(U) \approx \arg \max_{U_l = u_{l1} \cdots u_{lN}} \prod_{i=1}^{N} P(O_{li}|u_i).1/M \tag{5.1}$$

In Eq. (5.1), U represents the all unit sequences and U_l denotes the units used in the null-grammar unit decoder and M, N are the numbers of units respectively. Also O_l represents the observation sequence that match with that unit model.

In most recognisers the denominator, acoustic probability of the observation is assumed equal for all observations and it is not considered during the calculation of the likelihood. The aim of filler models is giving an accurate calculation for the acoustic probability of the observation.

The design philosophy of filler models should be consistent with their common purpose, filling the holes of the grammar in the normal speech recognizer. In order to do this the acoustic world is modelled in some units and these unit models are connected to each other within a loop structure (null-grammar). Thus the recognition process is made free from any effect of constrained unit networks like, grammar or language model. In other words, filler models output the best unconstrained acoustic path from the unit networks. In this thesis we have defined filler models

consisting of units in different details.

In general, the confidence of an utterance is determined by a comparison in between the best path from the acoustic unit network and the best path from the regular decoder. The desired ratio between the two should be equal to one or greater than one. This means that the filler model result is more trustworthy than that of the regular decoder. This is because filler model matches all units belonging to the observation with models in the loop without any dependency. However the normal decoder makes its matching by considering the connectivity of the trained models. Even if the observation sequence is not in the grammar or in the LM, according to the dependency of the models it generates an output in any way. For example let's assume that the word *work* is not included in our grammar or LM but the word *verb* is included. Also let's design our filler model as all-phone network, in which all phones are connected to each other without considering any dependency. For the acoustic observation of the word *work*, the normal decoder will give a transcription such as "verb". However in filler model decoding each phone is matched with a phoneme model and the concatenation of these best models are returned as "work" even if the *work* is not in the grammar/LM. Then it can be concluded that filler models are well on detecting the out of vocabulary errors. Also in grammar-based cases, they can successfully detect the out of grammar sentences or divisions.

In this work five filler models, each of which have different structures, are built and their contributions, either individually or in combinations with other models or features, on the robustness of speech recognition are investigated. In general, as filler model, all-phone networks or catch-all models are used and the output is used in the normalisation of the normal decoder's result. In our approach, we propose to use an all-triphone network, a phone-class decoder in addition to them with different scoring techniques. It is expected that with the increasing detail used in the unit models, we approximate $P(O)$ better. For example all-triphone network will have much more confidence information than the all-phone network. Although there are some overlapping information between the models, there are also discriminative information specific to that model. To make both information useful in the deci-

sion process we look for the the best combination in between these models and the other acoustic features, not related to filler models, in addition to their individual contributions.

5.2 Triphone Recognition Network

Context-dependent phonemes, called triphones, have been widely used for speech recognition and they significantly improve the accuracy and trainability. A triphone model is a phonetic model that takes into consideration both the left and the right neighboring phones. Triphone models are powerful because they capture the most important coarticulatory effects. However, as context dependent models generally have increased parameters, trainability becomes challenging. If we have less data to train, then the models become overfit to the data and thus the trained models can not be sensitive to the variations.

To construct a filler model which contains all triphones used in our training data, first, all different triphones were counted. Then, in order to decrease data-overfitting, a unit clustering technique was applied and the models were clustered as told in Section 3.3. Thus the number of parameters to estimate was decreased. After that, a null-grammar recognition network was built as in Figure 5.1. The network consists of 1673 triphone models, a generic silence model and a generic pause model.

Each word belongs to the input utterance is passed from this network and a filler model score is produced by summing all per-frame scores of the matched triphone models. Now let's look at this score calculation in more detail; In traditional filler model score calculations the word score is computed between the frame boundaries without normalising the unit durations. But in this approach we calculate per frame likelihood score for each phone of the word and then these scores are summed up and the result is assigned to that word as its filler model score.

Assume that the normal decoder search hypothesized $word_i$ with a log-likelihood score $S(word_i)$. Although in literature there are some works which use this score, it is more common to use the per-frame scores $S_{PF}(word_i)$ as in (5.2). Here f_e and

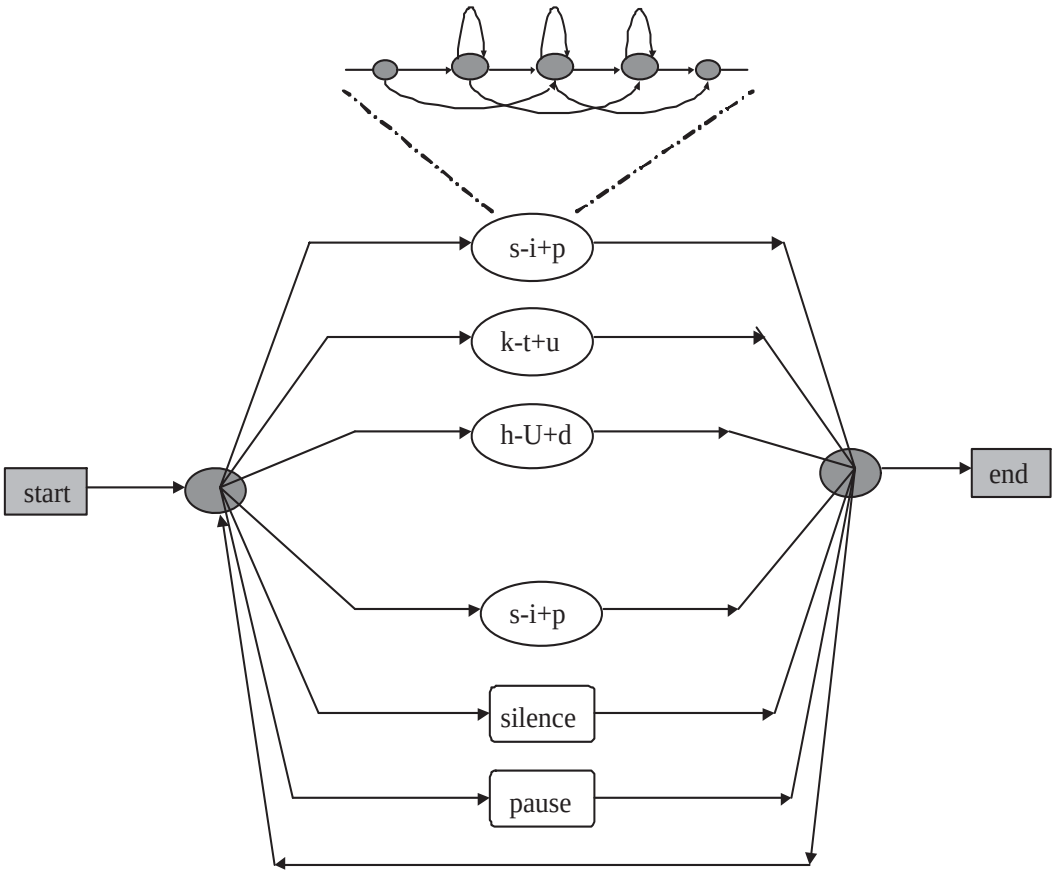

Figure 5.1: Triphone-Filler Netwok

f_s denote the end and start frames of the hypothesis.

$$S_{PF}(word_i) = \frac{S(word_i)}{f_e - f_s} \tag{5.2}$$

Filler model score for the same acoustic part of the input utterance is calculated as the summation of the per-frame scores of each sub-unit. Thus, first we normalise included phones and then we use these normalised phones. It is important to normalise the data in phone level since while speaking people emphasize some phones much more than the other and this can badly effect the score. Assume that the filler network found an alignment for the same input utterance as seen in the right part of the Figure 5.2. The equations (5.3) and (5.4) are used to calculate the filler model score, $S_{FS}(word_i)$ for the word $word_i$ and here, $S_{PF}(tri)$ denotes the normalized viterbi score for the triphone, tri, obtained from the triphone recognition network.

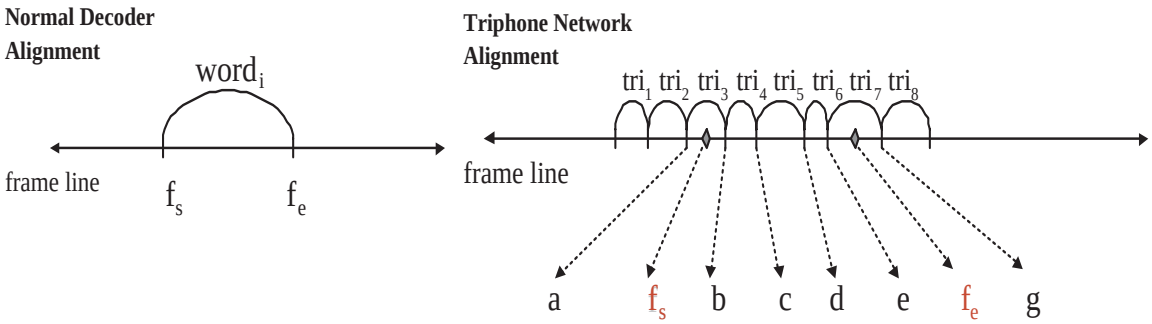

Figure 5.2: Scoring

$$S_{PF}(tri_i) = \frac{S(tri_i)}{f_{end_{tri}} - f_{start_{tri}}} \tag{5.3}$$

$$S_{FS}(word_i) = \frac{\sum_i^N S_{PF}(tri_i)}{N} \tag{5.4}$$

Here, N is the number of triphones in between hypothesized word boundaries. For this example, filler model score of $word_i$ can be calculated as below;

$$S_{FS}(word_i) = \frac{1}{4}\left[S_{PF}(tri_4) + S_{PF}(tri_5) + S_{PF}(tri_6) + \frac{1}{2}\left(\frac{S(tri_3)(b - f_s)}{b - a} + \frac{S(tri_4)(f_e - e)}{g - e}\right)\right]$$

In traditional filler score calculations, only the scores of included models are summed and then this summation is normalized by the the number of frames as in (5.5). However considering the effect of the emphasis on phones in the word could be helpful to increase the robustness of the estimate. For this filler model and the following filler models we used this proposed calculation scheme (5.4).

$$S_{FS}(word_i) = \frac{1}{f_e - f_s}\left[S(tri_4) + S(tri_5) + S(tri_6) + \frac{S(tri_3)(b - f_s)}{b - a} + \frac{S(tri_4)(f_e - e)}{g - e}\right] \tag{5.5}$$

The derived confidence features from the all-triphones filler network are; pure all-triphone loglikelihood score, $S_{FS}(word_i)$, and the ratio between the normal decoder's likelihood score and the filler model's score (5.6), called as log-likelihood ratio score.

$$S_{RS}(word_i) = S_{PF}(word_i) - S_{FS}(word_i) \tag{5.6}$$

The given scatter plot in Figure 5.3 shows the distribution of the test data according to the triphone filler score. The vertical axis denotes the values of the filler model score and the other axis denotes the normal decoder score. The blue **o** represents correctly recognized words and the red **x** is used for recognition errors.

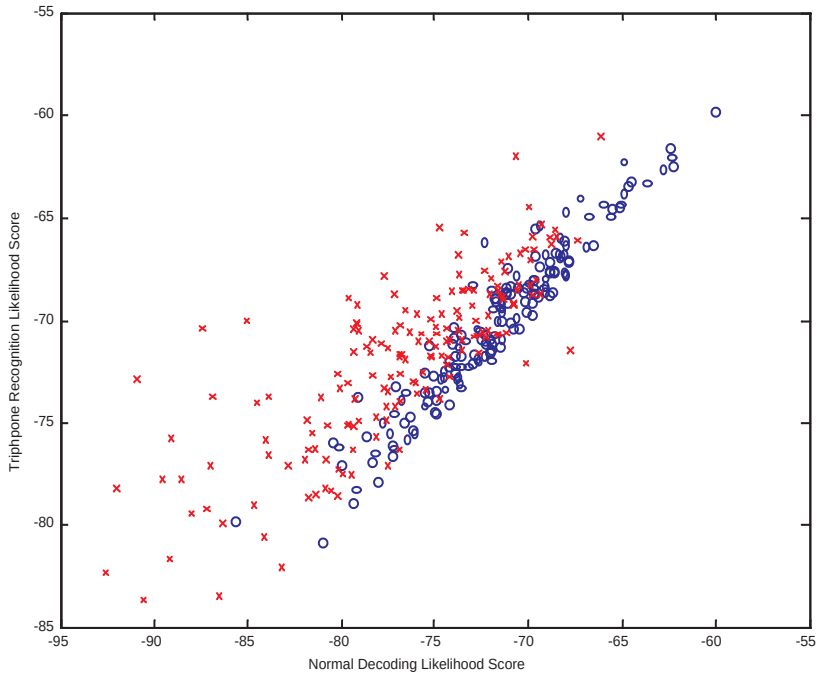

Figure 5.3: Distribution of the test data. X axis denotes the per frame viterbi score of the normal decoder, and Y axis denotes the triphone filler score.

As to the Figure 5.3 it seems that there is not an obvious separation between the two classes of hypotheses. This result was an expected one because we can not trust fully to the used triphone models in the filler model. However, it is also clear that this information carries some discriminative information.

In this chapter, only the individual contributions of filler models are presented. But in Chapter 6 the results of some reliable combinations are investigated. All performances in this thesis are compared according to the EER (defined in Section 2.3). And the test data consists of 1080 utterances collected from 12 speakers

and ten of them were not used neither in decoder training nor in confidence training.

The ROC curve (defined in Section 2.3) in Figure 5.4 was obtained from the system which uses the log-likelihood ratio score for triphone filler network as the single confidence feature.

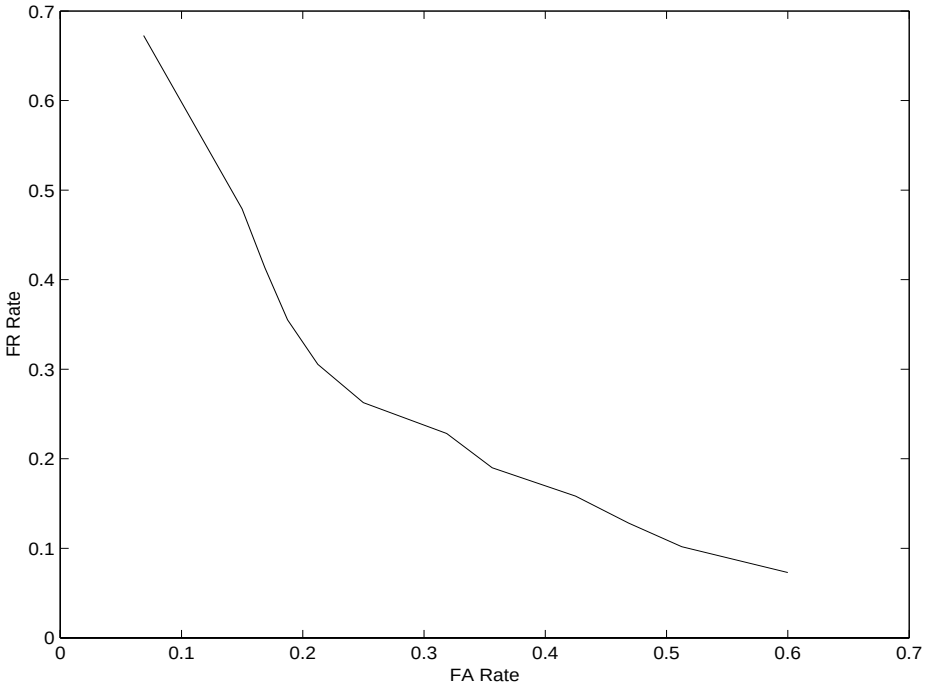

Figure 5.4: ROC curve for the single feature (triphone log-likelihood ratio score) confidence tagger. Obtained EER value : % 27.12

During the test, we have used two types of classifiers, GMM-classifier and ANN classifier. In Table 5.1, the EERs obtained from both classifiers for the mentioned feature, log-likelihood ratio score, are presented.

Table 5.1: EERs obtained from GMM and ANN classifiers for the feature, triphone log-likelihood ratio score.

GMM Classifier	**ANN Classifier**
% 27.82	% 27.12

In theory the best performance can be obtained from the network of all possible

unit models as in Eq. (5.1). But in practice, it is impossible to build and use such a network. In this case, using detailed sub-unit models can be seen as a good solution.

In this section, we have proposed to use the most detailed practical sub-units, triphones with a new score calculation scheme as a practical alternative to the all-words network. By using only the triphone filler score we have obtained an acceptable determination. But we expect that using this score, in a combination with other scores will give much better results. In the remaining of this chapter we follow to define some filler models and to evaluate their individual performances, and in the next chapter we will present the results of their combinations either together with other filler models, or together with other feature types.

5.3 All-Phones Network

In the previous case, in order to increase the detail of acoustic modeling, we used the network of context-dependent phones as the filler model and its result was the highest EER in between individual confidence contributions of filler models. But together with this case, during the remaining filler models, we continuously decrease the number of models. For example in this case we have only 29 monophone models and 2 more models to represent pause and silence, instead of having 1673 separate triphone models, as seen in Figure 5.5.

All-phone network is one of the most widely used filler models. In word-spotting applications it is added to the word network as an alternative word model and it works parallel with the normal decoder. But, in other cases like in our approach, a separate decoding process is used to evaluate the filler model score. We derive two kind of confidence features from this separate decoding: (*i*) all-phone network score and (*ii*) phone log-likelihood ratio score. To increase the discriminatory effect of the features, we use the proposed scoring scheme, finding per frame scores for each phone and then summing this scores to calculate the word score. As mentioned before, the main purpose of this scheme is to handle the effect of abnormal duration lengths of some phones. However, in literature there are results claiming that phone lengths include confidence information [39,17,41]. If we want to add the

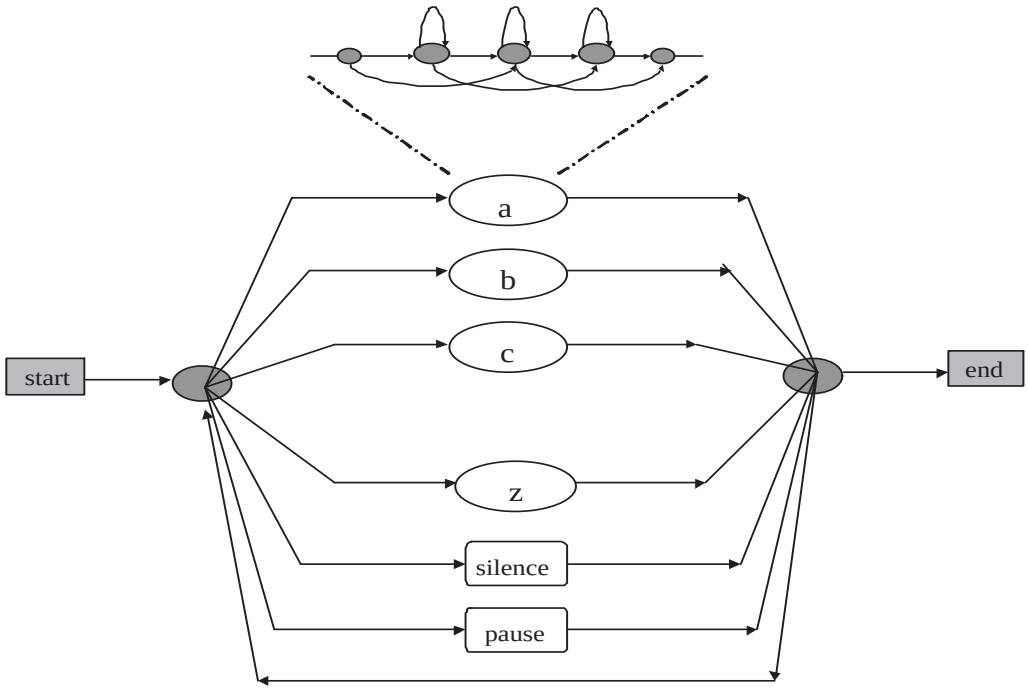

Figure 5.5: All-Phones Filler Netwok

phone length information to our confidence score, then the best way is to define the phone-duration information as a separate confidence feature in the feature vector. In our work, we derive the all-phone network score from the summation of per-frame scores as defined in Equation (5.4) and use the average phone duration information as a separate feature. The performance of the feature derived from the all-phone filler network is evaluated by the ROC curve in Figure 5.6.

In Table 5.2 results obtained from different classifiers are given. Such an EER:% 33.49 means that this feature gives us partially useful information about the confidence of the hypotheses. But as it will be seen later, the best performance will be obtained from a reliable combination of such features. Although all features include some overlapping information, they also include some discriminatory information. A good combination aims to decrease the overlap and increase the discriminatory information.

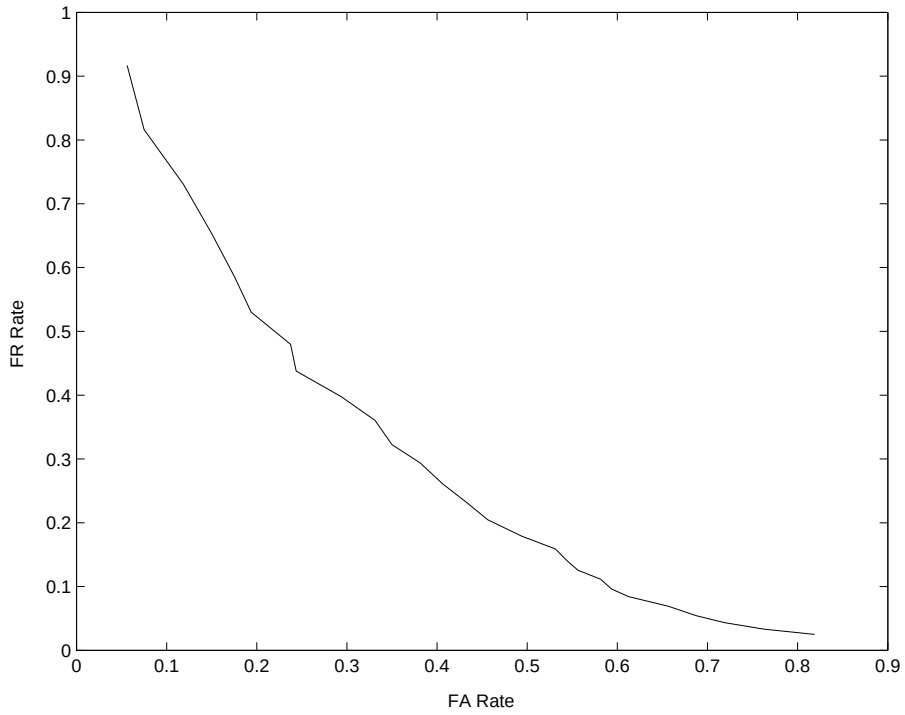

Figure 5.6: ROC curve for the single feature (phone log-likelihood ratio score) confidence tagger. Obtained EER value : % 33.49

Table 5.2: EERs obtained from GMM and ANN classifiers for the feature, phone log-likelihood ratio score.

GMM Classifier	**ANN Classifier**
% 33.49	% 33.62

5.4 Monophone Class Filler Model

In this filler model, we decreased the acoustic detail in the model a bit more. In all-phone filler network, we had used 31 phone models. Now we cluster these phones into six groups by considering their lingusitic characteristics [8] as in the Table 5.3.

The models of the groups were trained and they were placed in a fully connected network with two more models, silence and pause as in Figure 5.7.

Because of the reduction in the acoustic detail, the expected discrimination will

Table 5.3: Phone classes used as garbage models

Name of Phone Class	**Phones Included**
Stops	p,t,C,k,b,d,c,g
Fricatives	f,s,S,v,z,j
Nasals	m,n
Liquids	l,r
Glides	y, G, h
Backvowels	a, I, o, u
Frontvowels	e, i, O, U

hopefully be less than the previous filler models. However, less detailed modelling can provide extra information about the confidence of the word. Sensitivity to the variations is one of them. It is quite possible to have variations on the acoustic data and these variations can cause to the rejection of the data by its own model. Such kind of cases increase the number of False-Rejects. By more general models this problem can be handled in some extent.

Figure 5.8 is given to show the relationship between the detail in filler models and FA and FR rates. To prove the above claim, *by more general models FRs can be decreased*, the desired behaviour should be that the curve of the less detailed models, *monophone filler network*, should have a better performance after the EER point and it should be more nearer to the FR axis. In Figure 5.8, this desired behaviour is shown in some extent, but here, while interpreting the figure, the effect of the test data characteristic should also considered.

In this point it can be said that in some extetnt acoustically detailed filler model features would be useful in decreasing the False Accepts and similiarly, generic models would be helpful in decreasing the False-Rejects. Then it would be reasonable to claim that the best trade-off between the False-Accepts and False-Rejects would be obtained from an optimal combination of generic and detailed models. In Chapter 5 the performance of filler model feature combinations is investigated.

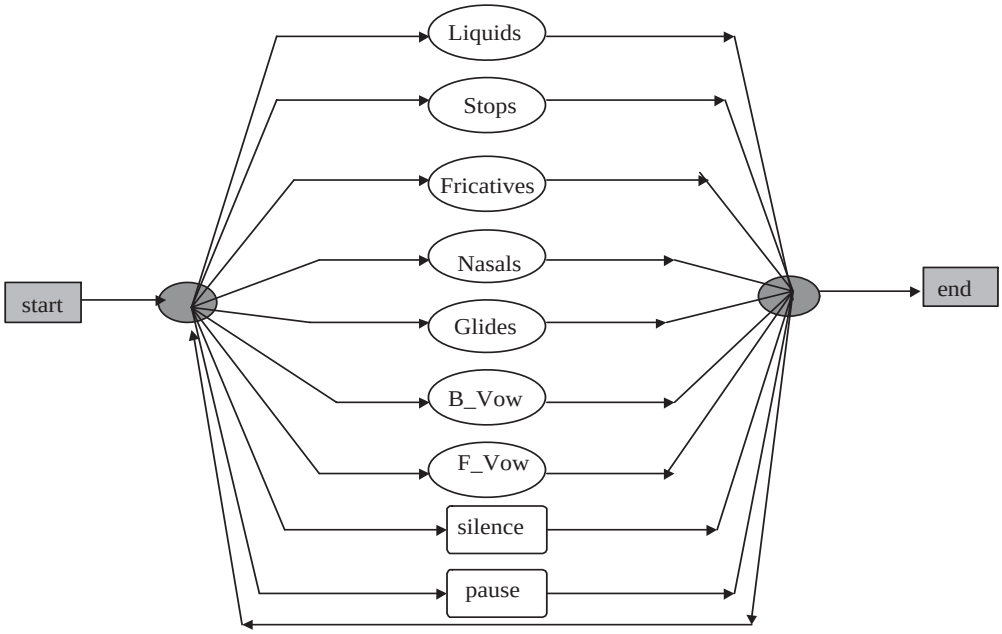

Figure 5.7: Phone Class Filler Netwok

The ROC curve in Figure 5.9. is obtained from the confidence evaluation system which uses only phone-class filler model score as the confidence measure. The EER is found as %38.78.

5.5 Catch-All Model

This filler model is the simplest and the widely used one among the filler models. The idea is to model all acoustic variability inside only one model. First such a model is trained and then it is placed in a loop structure as the only element of the loop, Figure 5.10.

A catch-all model can be designed with different number of states and different number of mixtures. In this work two different topologies were tried. In the first one 5-state topology, seen in Figure 5.10, was used and for the other we used 3-state structure, seen in Figure 5.11. Two types of confidence features were derived from these topologies: (*i*) per-frame recognition score of the 5-state catch all model, and (*ii*) per-frame recognition score of the 3-state catch-all model.

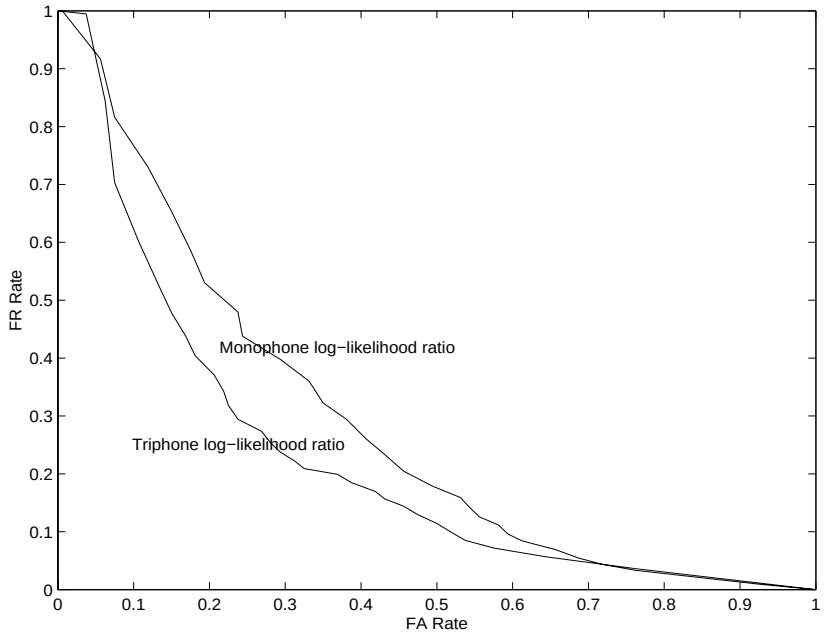

Figure 5.8: Comparison of two model types in the same ROC curve

Again in the evaluation of the system performance, we use ROC curves as in Figure 5.12. and Figure5.13. For 5-state structure the EER was found as %38.37 and for the 3-state structure it was found, %42.78.

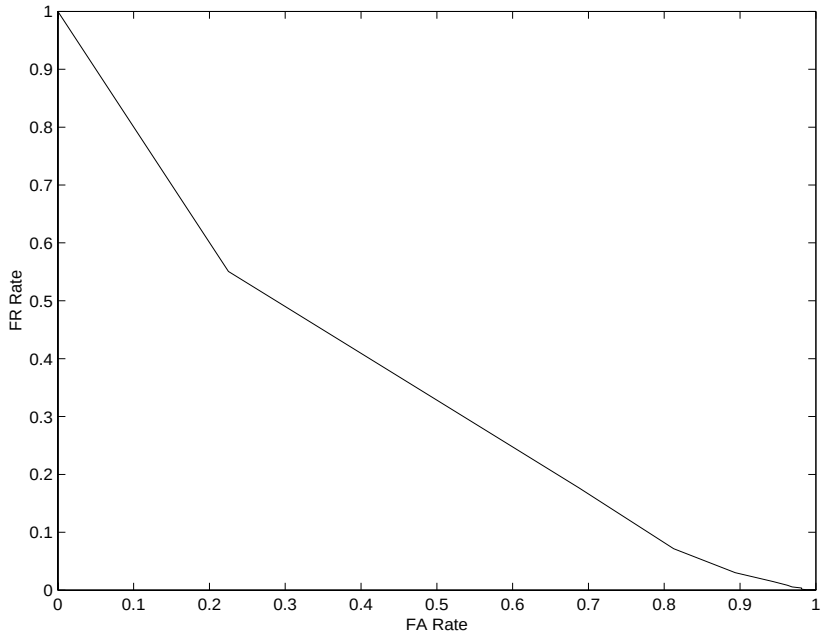

Figure 5.9: ROC curve for the single feature (phone-class filler score) confidence tagger. Obtained EER value : % 38.78

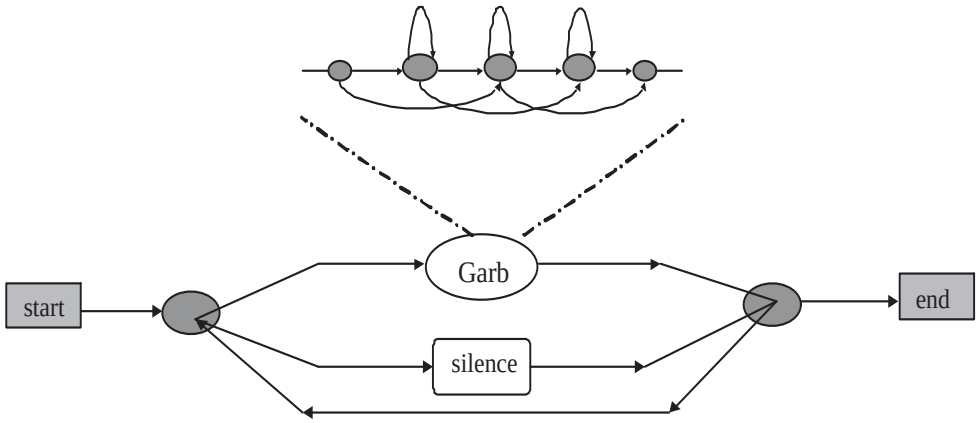

Figure 5.10: Catch-All Filler Netwok, 5-state topology

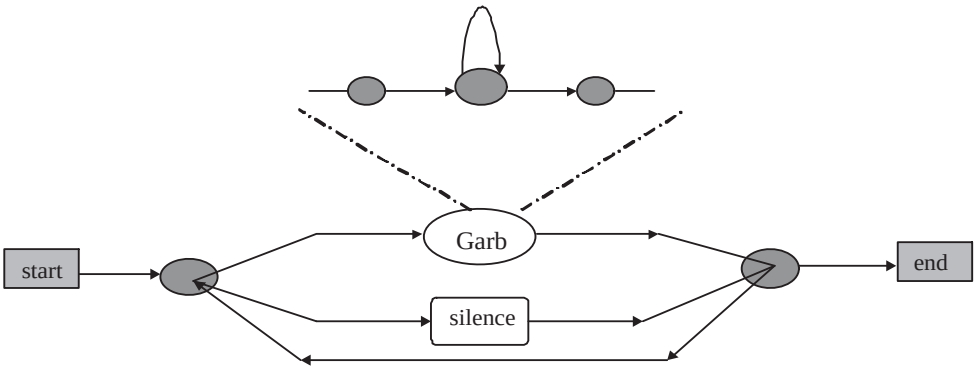

Figure 5.11: Catch-All Filler Netwok, 3-state topology

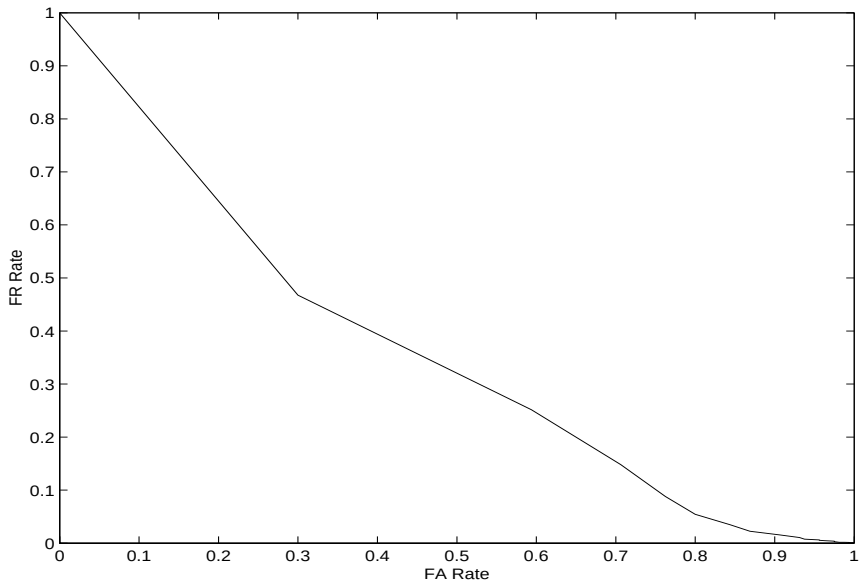

Figure 5.12: ROC curve for the single feature (5-state catch-all score) confidence tagger. Obtained EER value : % 38.37

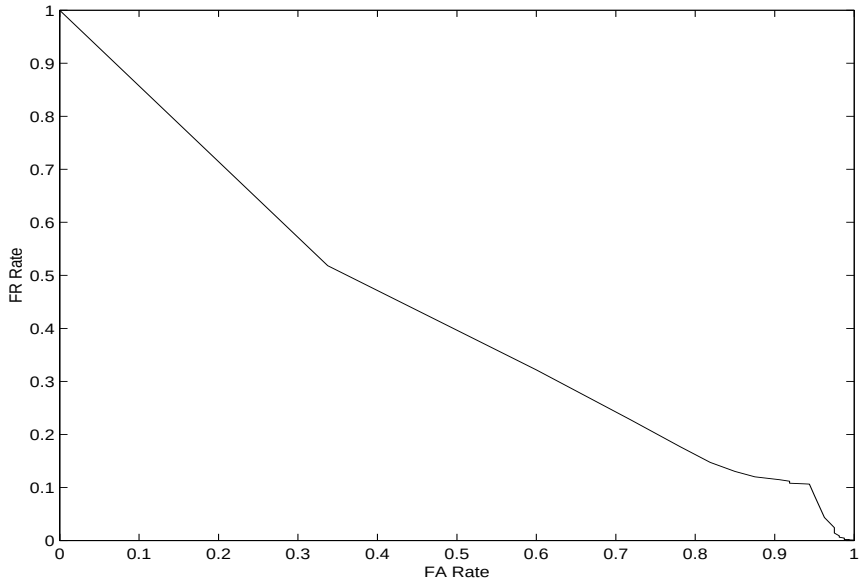

Figure 5.13: ROC curve for the single feature (3-state catch-all score) confidence tagger. Obtained EER value : % 42.78

Chapter 6

Confidence Vector Extraction and Results

6.1 Overview

In the previous chapter, the individual performance of certain filler models were investigated. It was concluded that with more detailed acoustic modelling better results, namely lower EERs, could be obtained. Also it was emphasized that despite the decrease in performance, each filler model type could provide different information about the data. In other words, all filler model types can contain overlapping information but on the other hand they can also contain some specific information, which can not be obtained from the other filler model types. For example, detailed models could be successful on decreasing the False-Accept rate. However generic models could be relatively more useful in decreasing the False-Reject rate than the detailed ones. Then combining both information would be a reasonable attempt to reach better EERs.

In this chapter, the performance of different combination schemes are investigated to understand the trade-off between the overlapping and discriminating information. First we introduce the features extracted and then the results on the analysis of their combinations are presented.

6.2 Extracted Features

We extract 12 different candidate features for each hypothesis. All features are acoustic word-level features. We focused especially on acoustic features, because they are generic ones so that they can be used in any kind of application. In Chapter

3 about 30 acoustic features were presented. Most of these features were derived from the normal decoding process. However, most of our features are derived from filler models. The reason for giving higher importance to filler models underlies the fact that filler models provide the pure, unconstrained acoustic information. The comparison of this information with the actual decoding result would give us a quite reliable measure. In addition to filler model features, we have also used normal decoder based features like the standart deviation of frame scores. Below we list the features used in our experiments;

1. Per-Frame log-likelihood Score (LL)
2. Per-Frame log-likelihood Score of the triphone recognition network (TL)
3. Per-Frame log-likelihood Score of the monophone recognition network (PL)
4. Per-Frame log-likelihood Score of the phone-class recognition network(CL)
5. Per-Frame log-likelihood Score of the 5-state catch-all model (CF)
6. Per-Frame log-likelihood Score of the 3-state catch-all model (CT)
7. Triphone log-likelihood ratio score (TR)
8. Monophone log-likelihood ratio score (PR)
9. Maximum frame score (MA)
10. Minimum frame score (MI)
11. Standart deviation of frame scores in the hypothesis (SD)
12. Number of phones in the hypothesis (NP)

6.3 Feature Combinations

After the evaluation of features in terms of their ability, in this part these features are combined into feature vectors and classified by a vector classifier into tags for each hypothesis to decide if the recognition is correct or not. The selection of features can be divided into two steps:

1. Selection of appropriate knowledge sources

2. classification of these feature vectors into the classes recognition-error and correct

In the remainder of this section, we first discuss some issues about the classifiers used. Then we present experiment on different candiadate feature combinations with different classifiers.

6.3.1 Classifier Issues

To train the classifier and confidence scoring mechanism, a set of confidence training data must be used and this data should be independent of the data used to train the recognizers. In our experiments the confidence training data consists of 1620 sample utterances. Each utterance was passed through the normal decoder and the filler models. For each word of the utterance a feature vector was extracted. Then each word hypothesis is labeled as *correct* or *incorrect*. The label *correct* means this hypothesis should be accepted, while the label *incorrect* means the hypothesis should be discarded. For utterance level tagging, certain measures are defined in language understanding part of the dialog system and they are not based on such kind of probabilistic methods. Although there are prior results [17,18,19,27] that propose the use of probabilistic methods, we use only semantic weights for each word and the summation of these wieghted scores helps us to determine whether the confident words are enough to understand the meaning of the utterance or not.

We designed two classifiers, a classifier based on GMMs, and a multilayer perceptron classifier. For each type of classifier we have two classes to be trained, *correct* and *incorrect*.

In the GMM classifier, we first trained both the correct hypotheses class and incorrect hypotheses class by using the confidence training data. For each class we determined different number of Gausssian components since in most cases the number of training data for incorrect hypotheses is much less than the number of correct ones. While determining the optimum numbers for classes we used a long list of component number combinations (see in Appendix C, 40 combinations). For each component combination, a new training process with a testing process was employed and the best performance component combination was found. During tests, we used

another independent data, which consisted of 1080 utterances, and we compared the best EERs found in each test. For each test process, since we did not yet known the reliable thresold value, we used a list of threshold values(140 values between -40 and 8) and for each threshold a separate point on the ROC curve was produced. At last, the EER and the threshold value was found and assigned as the performance for that combination of component numbers. To find the EER value for a feature combination, 40 different mixture combinations are tried, and for each mixture combination, we tested by trying 140 different thresholds.

In addition to the GMM classifier, a 2-layer neural network classifier with sigmodial transfer function units was trained, using a mean square error performance function and standart backpropagation. There are works[27,18] which report that such a simple classifier using one single unit in the hidden layer could not be significantly outperformed by more complex topologies. After training, to determine the threshold for the EER value, we have made decision experiments on the test data as described for the GMM classifier.

6.3.2 Combinations and Results

The problem of determining the best feature combination is two folds, curse of dimensionality and information overlapping. Having many features can be seen as an advantage if the amount of the training data is not limited. But in most cases the data is limited and so it is not sufficient to accurately define classes on high dimensional spaces. To combat the curse of dimensionality, the size of the feature vector should be reduced as much as possible. The other problem is information overlapping. Only the features providing different discriminating information should be combined in the same vector, because the overlapped information could decrease the joint discrimination of the feature vector.

We have determined 59 logical combinations of the 12 candidate features, described earlier, to get a sense of the tradeoff between the best EER and the other constraints like, computation time, curse of dimensionality, information overlapping, etc. In Table 6.1, EERs obtained for each feature combination by using GMM classifier are

presented. To find the EER value for a feature combination, 40 different mixture combinations were tried, and for each mixture combination, 140 different threshold values were tested.

Below some interpretations on the results are listed;

The Feature TR, *triphone log-likelihood ratio*, provides the best EER, % 27.82, among the individual performance. The difference between the best result in the table, Combination 33 (TR,PR,NP): % 24.32, is only %3.50, and also it is included by the best combination. Then it is reasonable to say that the most of the discriminating information in the best feature combination comes from this feature. It is an expected result as told in Section 5.1 since the feature TR uses the information of the most detailed modelling of unconstrained speech. The substantial contribution of Feature TR can be also seen from the good performance of the other combinations which include TR. From this interpretation, it can be concluded that for a reliable measure, the Feature TR must definitely be included.

Also the Feature PR provides valuable contribution for its combinations. It is included by almost every reliable combination in the table. However its single contribution is less than the one of the Feature TR since it uses less detailed acoustic models.

Another interpretation can be made on the relation between the number of features included by a combination and the increase on the performance. For the Combination 6 (TR), a single feature combination, the EER is 27.82 and the best EER in the table is %24.32, Combination33 (TR,PR,NP). The improvement obtained by adding extra two feature is about % 3.5. This improvement is significant if we think that the cost of extracting these extra features is much less than the one of the baseline feature. For triphone log-likelihood ratio score, Feature TR, we search on a network of 1675 models but for phone log-likelihood ratio score, Featre PR, we search on a network of only 31 models. Now let's define a case for which % 3.5 improvement could not be considered as significant. For this case assume that

the EER of the most efficient single feature combination (a) is % 33.49 and the best EER, % 30, is provided by the combination of a,b and c. Again assume that we extract the information a from a network of 31 models. In such a case, if we need a search on a network of 1600 models for the feature b, then the improvement can be considered as insignificant.

Although the Combination 58 (CL,CF,TR,PR,SD,NP) includes the best combination (TR,PR,NP), its performance (% 26.73) is less than the performance of its subset, (% 24.32). This means that the Combination 58 includes some overlapping information and this overlapping badly effects the overall confidence of the feature vector. Another reason for this case could be the curse of dimensionality problem. It is possible that the classifier training data is insuffcient to determine a boundary on such a 6-dimensional space.

Lastly, let's consider the Combination 50 (CF,TR,PR,SD,NP). Its EER is % 25.15 and it is one of the best performances in the experiment Table 6.1. This combination can be seen as the ideal combination of all features. Because it includes the normal decoder information (TR,PR), a generic filler model (CF), two detailed filler models (TR,PR), distribution of frame scores (SD) and the average phone duration information (NP), and thus it covers the whole candidate feature set. Actually it is expected that this ideal vector would have a better score than the set of (TR,PR,NP), since it includes features from all possible source types. Curse of dimensionality could be considered as the probable reason of the decrease in performance . Although it does not have the highest score, we have adapted this vector into our dialogue system.

In Table 6.1, for all experiments a GMM classifier has been used. We have determined 12 reliable combinations from the results in Table 6.1, and these combinations were also experimented by a ANN classifier. The results are compared in Table 6.3.

By considering the results in Table 6.3, it can be concluded that ANN classifier becomes more efficient with the increase in the dimension of the vector. The parallelism between the feature number and the performance of the ANN classifier

justifies the assumption that each examined feature represents distinct information about the confidence of the hypothesis and the combination of these distinct information improves the confidence annotation accuracy.

Table 6.1: Feature combination results for GMM classifier. Feature codes are given in Section 6.2

Combination Code	**Included Features**	**EER(%)**
1	TL	49.87
2	PL	49.53
3	CL	47.01
4	CF	45.61
5	CT	50.00
6	TR	27.82
7	PR	33.49
8	MA	46.40
9	MI	38.72
10	SD	44.24
11	NP	44.16
12	LL,TL	43.22
13	LL,PL	41.18
14	LL,CL	38.78
15	LL,CF	38.37
16	LL,CT	42.78
17	LL,TR	26.98
18	LL,PR	34.42
19	LL,MA	44.53
20	LL,MI	36.22
21	LL,SD	38.21
22	LL,NP	40.10
23	TR,PR	25.62
24	CL,CF	46.68
25	LL,TR,PR	26.52
26	LL,TL,PL	44.11
27	LL,MA,MI	37.03
28	LL,CL,CF	41.45
29	TL,PL,CL	45.79

Table 6.2: Table 6.1-Continued

Combination Code	**Included Features**	**EER(%)**
30	TL,PL,CF	46.47
31	PL,CL,CF	46.01
32	TR,PR,SD	25.35
33	TR,PR,NP	24.32
34	TL,TR,PR	26.41
35	CL,TR,PR	25.35
36	CF,TR,PR	25.34
37	LL,CF,TR,PR	26.47
38	LL,CL,TR,PR	27.47
39	LL,CT,TR,PR	30.04
40	LL,CL,CF,TR	27.81
41	LL,CL,CF,SD	41.45
42	LL,TR,PR,SD	25.55
43	CL,CF,TR,PR	26.23
44	TR,PR,MA,MI	26.43
45	CL,CF,MA,MI	37.57
46	TR,PR,SD,NP	26.19
47	TL,PL,TR,PR	27.75
48	LL,TL,SD,NP	39.83
49	LL,CL,CF,TR,PR	28.57
50	CF,TR,PR,SD,NP	25.15
51	CF,TR,PR,MA,MI	26.71
52	CL,TR,PR,MA,MI	27.39
53	LL,TR,PR,MA,MI	28.81
54	CL,CF,TR,PR,MA,MI	27.64
55	LL,CF,TR,PR,MA,MI	27.94
56	LL,CL,CF,TR,PR,SD	27.78
57	LL,CL,CF,TR,PR,NP	28.98
58	CL,CF,TR,PR,SD,NP	26.73
59	CL,CF,TR,PR,MA,MI,NP	27.43

Table 6.3: Experimental results with GMM and ANN classifiers

Combination Code	**Included Features**	**GMM(%)**	**ANN(%)**
6	TR	27.82	27.12
17	LL,TR	26.98	27.03
23	TR,PR	25.62	27.11
25	LL,TR,PR	26.52	27.08
32	TR,PR,SD	25.35	27.23
33	TR,PR,NP	24.32	28.50
43	CL,CF,TR,PR	26.23	27.21
50	CF,TR,PR,SD,NP	25.15	25.34
51	CF,TR,PR,MA,MI	26.71	24.90
54	CL,CF,TR,PR,MA,MI	27.64	24.88
58	CL,CF,TR,PR,SD,NP	26.73	24.92
59	CL,CF,TR,PR,MA,MI,NP	27.43	23.42

Chapter 7

Conclusion and Future Work

7.1 Conclusions

On the practical side of this thesis a dialogue management system was implemented for Turkish, described in Chapter 4. In this system we focused especially on the speech recognition and the confidence annotation parts. For speech recognition we obtained %97.2 word recognition rate. Although an objective comparison is not feasible for such kind of applications, since the result depends on many application-dependent parameters, this result is quite satisfactory when compared with other results in the literature.

On the theoretical side, we investigated the performance of filler models in different details for grammar-based continuous speech recognition systems. After the evaluation of features in terms of their ability to increase the confidence annotation accuracy, we investigated the performance improvement when they were combined in the same feature vector.

The first important contribution of the thesis is the use of a null-grammar triphone recognition network as a filler model. Among the filler model types investigated, the best individual performance was obtained from this filler model, triphone recognition network, with a significant improvement, %5.67 as compared to the all-phone network, a popular filler model type in literature. The main reason of the improvement is that the triphone network uses much more detailed acoustic models than the other filler model types. Although this detailed modelling brings some problems like trainability or computation cost, these disadvantages can be alleviated by some

implementation tricks like parameter-tying and pruning.

Another contribution is the evaluation of the combinatory performances of filler models in different details. In general, one type of filler model is used to define a reliable confidence information for the hypothesis. But it was observed that combinations of different filler model types increased the confidence annotation performance, %3.5 as compared to the best filler model performance (triphone recognition network).

Implementation of a dialogue system for Turkish is another important contribution of this thesis. A framework was defined to build Turkish dialogue systems. In this framework, a set of decision questions were derived for efficient triphone clustering in Turkish. Also a 3-hour speech data was collected. The database consists of 45 speakers, 31 male and 14 female, therefore it can also be used for speaker identification purposes. For data collection an automatic mechanism was implemented by using Dialogic interface card, and CT-ADE software environment.

7.2 Recommendations for Future Work

This thesis has opened up many new research possibilities on increasing the confidence annotation performance. First, alternative filler model types can be constructed like all-syllable network or null-grammar uni-gram network. Since these topologies also provide detailed modelling of the acoustic data, improvement on the determination of the confidence on the hypothesis can be expected. Also the performances of reliable combinations of filler models and other type of scores, listed in Appendix A, can be investigated.

To increase the accuracy in Turkish speech recognition systems, maybe using a more detailed phone-set can be useful. Oflazer et al. [40] proposed a new phone-set for Turkish, which consists of 45 different phones instead of standart 29 phone definitions. To be able to observe the real contribution of this phone set, the experiment should be done on a sufficiently large training data-set.

Appendix A

List of Some Confidence Features

(In the list level categorisation is shown by W, refers to word-level, and U, refers to utterance-level, letters.)

- **Acoustic Features**

  1. **Per-Frame LogLikelihood Score[35]-(W):** If we denote $W = q_1, q_2, , q_K$, then the basic measure which is provided by the Viterbi Decoding is defined as;

     $$P(W) = \frac{1}{N} \sum \log(P(qk|Xk)))$$

     where $P(qk|Xk)$ is the posterior probability of being in state q_k for acoustic vector X_k and N is the number of frames of the hypothesized word.
  2. **Mean Acoustic Likelihood Score[18]-(W):** The mean of the acoustic likelihood scores(not the log scores) across all acoustic observations in the word hypothesis.
  3. **Minimum/Maximum Acoustic Score[18]-(W):** The minimum log likelihood score across all acoustic observations(frames) in the word hypothesis. Also it is possible to use the maximum one.
  4. **Acoustic Score Standart Deviation (W):** The standart deviation of the log likelihood acoustic scores across all acoustic observations.
  5. **Mean Difference From Maximum Score[18]-(W):** The average difference, across all acoustic observations in the word hypothesis, between the acoustic score of a hypothesized phonetic unit and the acoustic score of highest scoring phonetic unit for the same observation.
  6. **Utterance Score[18]-(W):** The utterance confidence score generated from the utterance features.
  7. **Number of Acoustic Observations[18]-W:** The number of acoustic observations within the word hypothesis.
  8. **Mean Catch-All Score[18]-(W):** Mean score of the catch-all model across all observations in the word hypothesis.
  9. **Top-Choice Total Score[18]-(U):** The total Score from all models(acoustic, language, pronunciation models)
  10. **Top-choice Average Score[18]-(U):** The average score per word from all models for the top-choice hypothesis.

11. **Top-Choice Number of Words[18]-(U):** The number of words in the hypothesized choice
12. **LogAWE-end[27]-(W):** The logarithm of the number of active final word states in the search, averaged over a three-frame window araound the last frame of the hypothesized word.
13. **NscoreQ[27]-(W):** The log-score of the word divided by the log a-priori probability of the time segment Tw.
14. **NScore[27]-(W):** The normalized version of NscoreQ. The log-score of the word minus the log a-priori probability of time Tw.
15. **LogNPhones[27]-(W):** The log of the number of phones of the word.
16. **Top-choice Total N-gram Score[18]-(U):** The Total Score of the n-gram model for the top choice.
17. **Top-choice Average N-gram Score[18]-(U):** The average score per word of the N-gram model for the top choice.
18. **Top-Choice Total Acoustic Score[18]-(U):** The total acoustic score summed over all acoustic observations.
19. **Top-Choice Average Acoustic Score[18]-(U):** : The average acoustic score per acoustic observation for the top-choice
20. **LogAWE-beg[27]-(W):** It is the log of the number of active final word states in the search, averaged over a three-frame window around the first frame of the hypothesized word.
21. **NDisfluent[27]-(W):** The number of surronding non-word entities to the left and to the right of the word W.
22. **A-Entrop[27]-(W):** The average score per word of the N-gram model for the top choice.
23. **PercPhAll-Frame[17]-(W):** The percentage of frames in the hypothesized word which base phones match the base phones in the phone-only decoding.
24. **PerchPhAll-Phone[25]-(W):** Similar to the previous feature except the percentage is computed for phones rather than frames.
25. **Phone loglikelihood ratio score[25]-(W):** The ratio between the acoustic scores from the normal decoding and phone-only decoding.
26. **SpkRate[27]-(W):** Speaking rate computed by the quotient of the length of Tw and the expected word length. The expected word length is computed on the acoustic training set.
27. **SNR[27]-(W):** maximum SNR(speak to noise ratio) value with in Tw.
28. **SNR-MinMax[27]-(W):** The difference of the minimum and maximum SNR per frame within the Tw.
29. **Log-Train[27]-(W):** The number of times the word was observed in the training material.
30. **Duration[27]-(W):** Reciprocal of SpkRate
31. **N-active-leaf[27]-(W):** The average number of active final word atates in the search during the Tw, into which the word was aligned by the search.
32. **Triphone loglikelihood ratio score (W):** The ratio between the acoustic scores from the normal decoding and triphone-only decoding.

- **Languge Model Features**

  1. **LM-Ngram[27]-(W):** The number of times a backoff in the language model occurs
  2. **Lexical Score Drop[18]-(U):** The drop in the total Ngram score between the top hypothesis and the second best.

- **N-Best List Features**

  1. **A-stabil[27]-(W):** A number(typically 100) of alternative hypothesis with different weighting between acoustic scores and language model scores is computed. Each of these hypotheses is aligned against the reference output of the recognizer, where the reference output is defined as the output with the best weighting between acoustics and language model. For each word of the reference output, the number of times the same word occurs in the set of alternative hypothesis, normalized by the number of alternative hypotheses, is taken as feature value. A-stabil-before is the same feature, computed on the hypothesis before vocal-tract normalization.
  2. **Total Score Drop[18]-(U):** The drop in the total score between the top hypothesis and the second hypothesis in the N-best list
  3. **Acoustic Score Drop[18]-(U):** The drop in the total acoustic score between the second best hypothesis.
  4. **Top-Choice Average N-Best Purity[18]-(U):** The average N-best purity of all words in the top-choice hypothesis. The N-best purity for a hypothesized word is the fraction of N-best hypotheses in which that particular hypothesized word appears in the same location in the sentence.
  5. **Top-Choice High N-best Purity[18]-(U):** The fraction of words in the top-choice hypothesis which have an N-best purity of greater than one half
  6. **N-Best-Homogeneity[25]-(W):** The raito between the score of the paths containing the hypothesized word to the total path score of N-Best list
  7. **N-Best Word-Rate[25]-(W):** The ratio between the number of the paths containing the hypothesized word to the total number of paths in the N-Best list
  8. **Average N-best Purity[18]-(U):** The average N-best purity of all words in all of the N-best list hypothesis
  9. **High N-Best Purity[18]-(U):** The percentage of words across all N-best list hypotheses which have an N-best purity of greater than one half
  10. **Number of N-best Hypotheses[18]-(U):** The number of sentence hypotheses in the N-best list. This number is usually N but sometimes it can be less if fewer than N hypotheses are left after the search prunes away highly unlikely hypotheses.
  11. **N-best Purity[18]-(W):** The fraction of the N-best hypotheses in which the hypothesized word appears in the same position in the utterance

- **Lattice-Based Features**

1. **Lattice-PWP-Acoustic / Link Probability [43]-(W):** The log posterior word probability computed from the word lattice by summing and normalizing the scores of paths passing through the hypothesized word. It is computed using only acoustic score.
2. **Lattice-PWP-LM[43]-(W):** The log posterior word probability computed from the word lattice but using only the language model score
3. **Lattice-PWP[43]-(W):** Lattice-PWP-Acoustic + Lattice-PWP-LM
4. **Hypothesis Density[45]-(W):** High number of hypothesis with similar likelihood implies a higher probability of error. The number of links that span the time segment of a word in the most likely hypothesis should be strongly correlated with the word error. For each word three numbers of competing links are computed: at the word beginning, at the word end, and at the average number over the time segment into which the was aligned. To capture the effects of high or low confidence of the neighbouring words, the hypothesis density at the last frame and the first frame of the successor word are also calculated

- **Semantic Features**

1. **Parsing Mode [25]-(W):** This feature indicates if a word is parsed by the grammar. For a parsed word, it further indicates the position of the word within the slot, either edge or middle.
2. **Slot-Backoff-Mode [25]-(W):** Using a bigram language model for the slots, each parsed word is assigned the back-off mode of the slot that it belongs to. This feature is computed on a two-word window that contains both the current word and the next word.

Appendix B

List of Decision-Tree Questions for Triphone Clustering in Turkish

"R_NonBoundary" { *+* }
"R_Silence" { *+sil }
"R_Consonants" { *+b,*+c,*+d,*+g,*+G,*+j,*+l,*+m,*+n,*+r,*+v,*+y,*+z,
+C,+f,*+h,*+k,*+p,*+s,*+S,*+t }
"R_Resonance" { *+b,*+c,*+d,*+g,*+G,*+j,*+l,*+m,*+n,*+r,*+v,*+y,*+z }
"R_Non_Resonance" { *+C,*+f,*+h,*+k,*+p,*+s,*+S,*+t }
"R_Stop" { *+p,*+t,*+C,*+k,*+b,*+d,*+c,*+g }
"R_Nasal" { *+m,*+n }
"R_Fricative" { *+f,*+s,*+S,*+v,*+z,*+j }
"R_Liquid" { *+l,*+r }
"R_Glide" { *+y,*+G,*+h }
"R_Vowel" { *+a,*+I,*+o,*+u,*+e,*+i,*+O,*+U }
"R_Back_Vowel" { *+a,*+I,*+o,*+u }
"R_Front_Vowel" { *+e,*+i,*+O,*+U }
"R_Non_Round_Vowel" { *+a,*+e,*+I,*+i }
"R_Roun_Vowel" { *+o,*+O,*+u,*+U }
"R_W_Vowel" { *+a,*+e,*+o,*+U }
"R_N_Vowel" { *+u,*+U,*+I,*+i }
"R_Voiced" { *+b,*+d,*+g,*+v,*+z,*+j,*+c }
"R_UnVoiced" { *+p,*+t,*+k,*+f,*+s,*+S,*+C }
"R_Double_lip" { *+p,*+b,*+m }
"R_Up_teeth" { *+f,*+v }
"R_Back_teeth" { *+t,*+d }
"R_Gum" { *+s,*+z,*+n,*+r,*+l,*+C,*+c }
"R_Hard_Palate" { *+S,*+j,*+y }
"R_Palate " { *+k,*+g }
"R_Larynx " { *+h }
"R_Burst_Palate " { *+b,*+p }
"R_Gum_Palate " { *+s,*+z }
"R_Leakeage_Palate " { *+S,*+j }
"R_Burst_Gum_Palate " { *+c,*+C }
"R_a" { *+a }
"R_b" { *+b }
"R_c" { *+c }
"R_C" { *+C }
"R_d" { *+d }
"R_e" { *+e }
"R_f" { *+f }
"R_g" { *+g }
"R_G" { *+G }

"R_h" { *+h }
"R_I" { *+I }
"R_i" { *+i }
"R_j" { *+j }
"R_k" { *+k }
"R_l" { *+l }
"R_m" { *+m }
"R_n" { *+n }
"R_o" { *+o }
"R_O" { *+O }
"R_p" { *+p }
"R_r" { *+r }
"R_s" { *+s }
"R_S" { *+S }
"R_t" { *+t }
"R_u" { *+u }
"R_U" { *+U }
"R_v" { *+v }
"R_y" { *+y }
"R_z" { *+z }
"L_NonBoundary" { *-* }
"L_Silence" { sil-* }
"L_Consonants" { b-*,c-*,d-*,g-*,G-*,j-*,l-*,m-*,n-*,r-*,v-*,y-*,z-*,C-*,f-*,h-*,k-*,p-*,s-*,S-*,t-* }
"L_Resonance" { b-*,c-*,d-*,g-*,G-*,j-*,l-*,m-*,n-*,r-*,v-*,y-*,z-* }
"L_Non_Resonance" { C-*,f-*,h-*,k-*,p-*,s-*,S-*,t-* }
"L_Stop" { p-*,t-*,C-*,k-*,b-*,d-*,c-*,g-* }
"L_Nasal" { m-*,n-* }
"L_Fricative" { f-*,s-*,S-*,v-*,z-*,j-* }
"L_Liquid" { l-*,r-* }
"L_Glide" { y-*,G-*,h-* }
"L_Vowel" { a-*,I-*,o-*,u-*,e-*,i-*,O-*,U-* }
"L_Back_Vowel" { a-*,I-*,o-*,u-* }
"L_Front_Vowel" { e-*,i-*,O-*,U-* }
"L_Non_Round_Vowel" { a-*,e-*,I-*,i-* }
"L_Round_Vowel" { o-*,O-*,u-*,U-* }
"L_W_Vowel" { a-*,e-*,o-*,u-* }
"L_N_Vowel" { u-*,U-*,I-*,i-* }
"L_Voiced" { b-*,d-*,g-*,v-*,z-*,j-*,c-* }
"L_UnVoiced" { p-*,t-*,k-*,f-*,s-*,S-*,C-* }
"L_Double_lip" { p-*,b-*,m-* }
"L_Up_teeth" { f-*,v-* }
"L_Back_teeth" { t-*,d-* }
"L_Gum" { s-*,z-*,n-*,r-*,l-*,C-*,c-* }
"L_Hard_Palate" { S-*,j-*,y-* }
"L_Palate " { k-*,g-* }
"L_Larynx " { h-* }
"L_Burst_Palate " { b-*,p-* }
"L_Gum_Palate " { s-*,z-* }
"L_Leakeage_Palate " { S-*,j-* }
"L_Burst_Gum_Palate " { c-*,C-* }

"L_a" { a-* }
"L_b" { b-* }
"L_c" { c-* }
"L_C" { C-* }
"L_d" { d-* }
"L_e" { e-* }
"L_f" { f-* }
"L_g" { g-* }
"L_G" { G-* }
"L_h" { h-* }
"L_I" { I-* }
"L_i" { i-* }
"L_j" { j-* }
"L_k" { k-* }
"L_l" { l-* }
"L_m" { m-* }
"L_n" { n-* }
"L_o" { o-* }
"L_O" { O-* }
"L_p" { p-* }
"L_r" { r-* }
"L_s" { s-* }
"L_S" { S-* }
"L_t" { t-* }
"L_u" { u-* }
"L_U" { U-* }
"L_v" { v-* }
"L_y" { y-* }
"L_z" { z-* }

Appendix C

List of Used GMM Component Combinations

Table C.1: Experimented combinations of mixture numbers for the classes, *correct* and *incorrect*

Combination Code	**Class, *Correct***	**Class, *Incorrect***
1	80	10
2	60	20
3	60	10
4	60	5
5	50	25
6	50	15
7	50	10
8	50	5
9	40	25
10	40	15
11	35	20
12	30	20
13	30	15
14	30	10
15	30	5
16	30	1
17	25	25
18	25	20
19	25	10
20	25	5
21	20	20
22	20	10
23	20	5
24	15	15
25	15	10
26	15	5
27	10	10
28	10	5
29	10	3
30	10	2

Combination Code	**Class, *Correct***	**Class, *Incorrect***
31	10	1
32	8	5
33	8	3
34	8	1
35	5	3
36	5	1
37	3	2
38	3	1
39	2	1
40	1	1

Bibliography

[1] Young, S. Large vocabulary continuous speech recognition. IEEE Signal Processing Magazine, 13(5):45-47, 1996.

[2] Picone J. W. Signal modelling techniques in speech recognition. Proceedings of the IEEE, vol. 81, no. 9, pp. 1215-1247, 1993.

[3] Davis S.B., Mermelstein P. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. IEEE Transactions on Acoustic, Speech and Signal Processing, vol. 28, no. 4, pp. 357-366, 1980.

[4] Yapanel U. Garbage modelling techniques for a Turkish keyword spotting system, Msc. Thesis, Boğaziçi University, 2000.

[5] Robinson T. Speech Analysis. Lecture Notes, http://mi.eng.cam.ac.uk/ ajr/SA95, 1995.

[6] Hermansky H. Perceptual Linear Predictive (PLP) Analysis of Speech. Journal Of Acoustic Society of America. vol. 87, no. 4. pp. 1738-1752, 1990.

[7] Rabiner L. R. A tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. Proceedins of IEEE, vol. 77, no. 2, pp. 257-286, 1989.

[8] Young S. et al. The HTK Book v3.0. Cambridge University, 1999.

[9] Huang X., Acero A., Hon H.W. Spoken Language Processing. Prentice Hall, Upper Saddle River, NJ, 2001.

[10] Jurafsky D., Martin J. H. Speech and Language Processing. Prentice Hall, Upper Saddle River, NJ, 2000.

[11] Jelinek F. Statistical Methods in Speech Recognition. Prentice Hall, Upper Saddle River, NJ, 2000.

[12] Waibel A., Hanazawa T., Hinton G., Shikano K., Lang K. Phoneme recognition using time-delay neural networks. IEEE Trans. ASSP, ASSP-37, 328-339, 1989.

[13] Port R. Representation and recognition of temporal patterns. Connection Science (2) 1-2, 151-176, 1990.

[14] Sakoe H., Chiba S. Dynamic Programming Algorithm Optimization for Spoken Word Recognition. IEEE Trans. on Acoustic, Speech and Signal Processing, 26(1), pp. 43-49, 1978.

[15] Huang X., Alleva F., Hon H., Hwang M., Rosenfeld R. The Sphinx-II speech recognition system: an overview. Comp. Speech and Lang, 2 :137-148, 1993.

[16] Legetter C.J., Woodland P.C. Maximum likelihood linear regression for speaker adaptation of continuous density hidden markov models. Computer Speech and Language 9, 171-185, 1995.

[17] Chase L. Error-Responsive Feedback Mechanisms for Speech Recognition. Phd Thesis, Carnegie Mellon University, 1997.

[18] Hazen T.J., Burianek T., Polifroni J., Seneff S. Recognition confidence scoring for use in speech understanding systems. Computer Speech and Language16, 49-67, 2002.

[19] Cox S., Dasmahapatra S. A high-level approach to confidence estimation in speech recognition. Proceedings of the European Conference on Speech Communication and Technology, Budapest, Hungary, pp. 41-44, 1999.

[20] Kamppari S., Hazen T., 2000. Word and phone level acoustic confidence scoring. In Proceedings of the International Conference on Acoustics, Speech, and Signal Processing, Istanbul, June, 2000.

[21] Williams G., Renals S. Confidence measures from local posterior probability estimates. Computer Speech and " Language, 13, 395-411, 1999.

[22] Ma C., Randolph M. A., Drish J. A support vector machines-based rejection technique for speech recognition. ICASSP, 2001.

[23] Weintraub M., Beaufays F., Rivlin Z., Konig Y., Stolcke A. Neural - Network based measure of confidence for word recognition. Proceedings of 1997 ICASSP, Munich, vol. II, pp. 887–890, 1997.

[24] Katz M., Meier H., Dolfing H., Klakow D. Robustness of Linear Discriminant Analysis in automatic speech recognition. ICPR02 (III: 371-374), 2002.

[25] Zhang R., Rudnicky A.I., Word level confidence annotation using combinations of features.Eurospeech 2001 - Scandinavia, 2001.

[26] Duda R. O., Hart P.E., Stork D. G., Pattern Classification, John Willey Sons, Inc., 2001.

[27] Schaaf T., Kemp T. Confidence measures for spontaneous speech recognition. Proceedings of the International Conference on Acoustics, Speech, and Signal Processing, Munich, pp. 875-878, 1997.

[28] Artuner H. Bir Trke Fonem Kmeleme Sistemi Tasarm ve Gereklestirimi. Phd. Thesis, Hacettepe University, 1994.

[29] Bayer S., Doran C., George B. Exploring Speech-Enabled Dialogue with the Galaxy Communicator Infrastructure. Proceedings of HLT Conference, 2001.

[30] Pellom B., Ward W., Hansen J., Cole R., Hacioglu K., Zhang J., Yu X., Pradhan S. University of Colorado Dialog Systems for Travel and Navigation. Human Language Technology Conference (HLT), San Diego, California, March, 2001.

[31] Burineak T.K. Building a speech understanding system using word spotting techniques. Msc. Thesis, Massachusetts Institute of Technology, 2000.

[32] Vural E. Prosodic TTS system for Turkish Language. Msc. Thesis, Sabancı University, 2003.

[33] Gunawardana A., Hon H. W., Jiang L. Word-based acoustic confidence measures for LVSR. International Conference on Speech and Lan- guage Processing, 1998.

[34] Mori, De R. Spoken Dialogues with Computers. Academic Press, Great Britain, 1998.

[35] Weintraub, M. LVCSR log-likelihood ratio scoring for keyword spotting. in Proceedings of 1995 ICASSP, Detroit, vol. I, pp. 297–300, April 1995.

[36] Cox S., Rose R. C. Confidence measures for the Switchboard database. Proceedings of ICSLP'96, Philadelphia, vol. I, pp. 478–481, 1996.

[37] Caminero J., Torre C., Villarrubia L., Martin C., Hernandez L. On-line garbage modeling with discriminant analysis for utterance verification. ICASSP97, 1997.

[38] Neti C.V., Roukos S., Eide E. Word-based confidence measures as a guide for stack search in speech recognition. Proceedings of ICASSP'97, vol. II, pp. 883–886, 1997.

[39] Cox S., Rose R. Confidence measures for the switchboard database. ICASSP-96 1:511514, 1996.

[40] Oflazer K., Inkelas S. A Finite State Pronunciation Lexicon for Turkish. In Proceedings of the EACL Workshop on Finite State Methods in NLP, Budapest, Hungary, April 13-14, 2003.

[41] Moreno P.J., Logan B., Raj B. A boosting approach for confidence scoring, In Eurospeech, 2001.

[42] Maison B., Gopinath R. Robust confidence annotation and rejection for continuous speech recognition. Proc. ICASSP01, 2001.

[43] Wessel F., Schlüter R., Macherey K., Ney H. Confidence measures for large vocabulary continuous speech recognition. IEEE Transactions on Speech and Audio Processing, vol. 9, no. 3, 2001.

[44] Davis S. B., Mermelstein P. Comparison of parametric representations of monosyllabic word recognition in continuously spoken sentences. IEEE Trans. Acoustics, Speech and Signal Processing, 28, p. 357-366, 1980.

[45] Kemp T., Schaaf T. Estimating confidence using word lattices. Proc. 5th Eur. Conf. Speech Communication and Technology. pp. 827-830, 1997.