Fuzzy GMM-based Confidence Measure Towards Keyword Spotting Application

by

Mohamed Kacem Abida

A thesis
presented to the University of Waterloo
in fulfilment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2007

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

The increasing need for more natural human machine interfaces has generated intensive research work directed toward designing and implementing natural speech enabled systems. The Spectrum of speech recognition applications ranges from understanding simple commands to getting all the information in the speech signal such as words, meaning and emotional state of the user. Because it is very hard to constrain a speaker when expressing a voice-based request, speech recognition systems have to be able to handle (by filtering out) *out of vocabulary* words in the users speech utterance, and only extract the necessary information (*keywords*) related to the application to deal correctly with the user query. In this thesis, we investigate an approach that can be deployed in keyword spotting systems. We propose a confidence measure feedback module that provides confidence values to be compared against existing Automatic Speech Recognizer word confidences. The feedback module mainly consists of a soft computing tool-based system using fuzzy Gaussian mixture models to identify all English phonemes. Testing has been carried out on the JULIUS system and the preliminary results show that our feedback module outperforms JULIUS confidence measures for both the correct spotted words and the falsely mapped ones. The results obtained could be refined even further using other type of confidence measure and the whole system could be used for a Natural Language Understanding based module for speech understanding applications.

Acknowledgments

The author would like to thank his supervisor, Professor Fakhreddine Karray, for his guidance and support on this research work. The author would also like to acknowledge Dr. Jiping Sun for his advice and assistance. Many thanks are also due to my thesis readers Dr. Ramadan El Shatshat and Dr. Kumaraswamy Ponnambalam for taking the time and assessing my work within a short time frame. Special thanks are due as well for Arash Abgari, Dr. Jin-Myung Won, Sameeh Ullah and Abbas Ahmadi for all the fruitful discussions and feedback. The author acknowledges Vestec Inc. for providing valuable resources used in this thesis and for its courtesy to use its labs to carry out some of the reported experiments.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The value to our society of being able to communicate with computers in everyday *natural* language cannot be overstated. Imagine asking your computer *When is the next televised National League baseball game?* Or being able to tell your PC *Please format my homework the way my English professor likes it.* Commercial products can already do some of these things, and AI scientists expect many more in the next decade. One goal of AI work in natural language is to enable communication between people and computers without resorting to memorization of complex commands and procedures.

The increasing need for more natural human-machine interfaces has generated intensive research work directed towards designing and implementing natural speech enabled systems. The spectrum of speech recognition applications ranges from understanding simple commands to getting all the information in the speech signal such as words, meaning and emotional state of the user. Because it is very hard to constrain a speaker when expressing a voice based request, speech recognition systems have to be able to handle (by filtering out) *out of vocabulary* words in the users speech utterance and only extract the necessary information (*keywords*) related to the application to deal correctly with the user query.

1.1 Motivation

The major problem in speech recognition is the inability to predict exactly what a user might say. We can only know the keywords that are needed for a specific speech enabled system. Fortunately, that's all we need in order to interact adequately with

the user. The system that spots these specific terms in the user utterance is called a keyword spotting system. Most state of the art of the keyword spotting systems rely on a filler model (garbage model) in order to filter out the out of vocabulary uttered words[2, 3]. Up to now, there is no universal optimal filler model that can be used with any automatic speech recognizer (ASR) and handle natural language. Several researchers have attempted to build reliable and robust models, but when using these kind of garbage models, the search space of the speech recognizer becomes big and the search process takes considerable time. Our ultimate goal is to build a filler model free keyword spotting system. That is, without having to come up with a model to represent the words outside the speech grammar, we should be able to spot only the necessary information from an utterance that contains a mix of keywords and garbage words.

The approach we are proposing here consists of providing the ASR with only the keywords and letting it do the recognition of natural sentences and cause false mapping. That is when the ASR deals with a word that is not in the speech grammar, it will automatically wrongly map it to an in grammar keyword. By doing this, we are stretching the ASR to the limit and leading it to cause a lot of wrong recognition. As such we have an ASR output that might contain the correct uttered keywords and also falsely mapped words. Now we have to find a way to filter out and discard all these recognition errors. To do that, we will have to rely on a confidence metric to evaluate the degree of correctness for each recognized word. The easiest way to do this, is to use the ASR confidence values, but these confidences are usually not reliable. In fact, most of the time, the wrongly spotted keywords have a high confidence score. That is why we cannot rely on the ASR confidence measures. Our solution to this problem is to build a feedback module that will take the ASR output and rank it and then we base our decision making process (discarding or accepting ASR outputs) on this ranking.

1.2 Objectives

Our principal goal in this research work is to find a way for building a filler model free keyword spotting system. To achieve this, we need to integrate adequately a number of steps summarized as follows:

- Build a phoneme classifier based on fuzzy Gaussian mixture modeling, which is the fuzzy C-means based modification of the Gaussian mixture modeling.

- Evaluate the ASR outputs using the phoneme classifier integrated with some confidence measure techniques.
- Compare the ASR confidence metrics with our confidence scoring method and check if they are conclusive enough to adopt this approach.

1.3 Contributions

To achieve the stated goals, we have proposed in this work to integrate in a novel way some classifier technique for phoneme recognition with some standard confidence measure approaches. Along the way, we have analyzed the output of each one of these approaches to find ways on how to improve our proposed technique. This adequate integration will provide us with a new confidence measure for each recognizer output, that can be used to better differentiate between the correctly spotted words and the falsely mapped ones.

1.4 Thesis Organization

The remainder of this thesis is organized as follows. The chapter 2 reviews the state of the art of phoneme recognition and classification, keyword spotting techniques and confidence measures techniques used in today's speech recognizers. Then chapter 3 gives an overview of the architecture of the system and describes the various techniques used to implement our approach, mainly the Gaussian mixture modeling and the fuzzy Gaussian mixture modeling used for phoneme classification. This chapter describes as well the chosen confidence metrics for the ranking of the ASR output. Chapter 4 presents the obtained results with interpretations for the phoneme classification module and then discusses the performance of the full system in terms of the proposed ranking compared to the ASR confidence scores. Finally chapter 5 summarizes the contributions of this thesis and introduces the focus of future research.

Chapter 2

Background And Literature Review

This chapter reviews the state of the art of what has been achieved in phoneme classification and recognition. It also highlights some current approach for confidence measures in speech recognition and presents some research work in the keyword spotting field.

2.1 Phoneme Recognition

One basic question in speech processing is: what are the primary units that must be first recognized to start the recognition process? Should these units be words, syllables, phonemes or acoustic events? Practically speaking a good phonological model to be used in a speech recognizer must satisfy these conditons[4]:

- Each unit should be mapped to a minimally variant acoustic region, and distinct units should map into maximally separate acoustic regions.
- The inventory of these units should be small in order to minimize the need for the training data.

The most promising approach to the problem of large vocabulary automatic speech recognition is to build a recognizer that works at the phoneme level. Let's start by a brief introduction to the phonetics and then we will outline the phoneme recognition state of the art.

2.1.1 Phonetics

Like fingerprints, every speakers vocal anatomy is unique, and this makes for unique vocalizations of speech sounds. Yet language communication is based on commonality of form at the perceptual level. To allow discussion of the commonalities, researchers have identified certain gross characteristics of speech sounds that are adequate for description and classification of words in dictionaries. They have also adopted various systems of notation to represent the subset of phonetic phenomena that are crucial for meaning[1].

In speech science, the term phoneme is used to denote any of the minimal units of speech sound in a language that can serve to distinguish one word from another. In this report, we conventionally use the term phone to denote a phonemes acoustic realization. For example, English phoneme */t/* has two very different acoustic realizations in the words *sa**t*** and *me**t**er*. We had better treat them as two different phones if we want to build a spoken language system. Speech is produced by air pressure waves emanating from the mouth and the nostrils of a speaker. In most of the worlds languages, the inventory of phonemes can be split in two basic classes:

- *Consonants*: articulated in the presence of constrictions in the throat or obstructions in the mouth (tongue, teeth, lips) as we speak.
- *Vowels*: articulated without major constrictions and obstructions.

Vowels

The tongue shape and positioning in the oral cavity do not form a major constriction of air flow during vowel articulation. However, variations of tongue placement give each vowel its distinct character by changing the resonance, just as different sizes and shapes of bottles give rise to different acoustic effects when struck. The primary energy entering the pharyngeal and oral cavities in vowel production vibrates at the fundamental frequency. The major resonances of the oral and pharyngeal cavities for vowels are called F1 and F2 - the first and second formants[1], respectively. They are determined by tongue placement and oral tract shape in vowels, and they determine the characteristic timbre or quality of the vowel. The different vowel qualities are realized in acoustic analysis of vowels by the relative values of the formants. The acoustics of vowels can be visualized using spectrograms, which display the acoustic energy at each frequency, and how this changes with time. Figure 2.1 shows the spectrogram of the word *sees*. Notice here the difference in

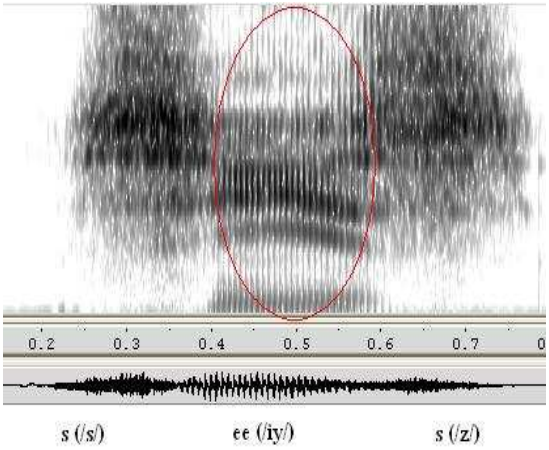

Figure 2.1: Spectrogram of the word *sees*

the characteristics of the energy of the middle vowel. We can obviously conclude here that formants play a major role in the recognition of vowels.

The major articulator for English vowels is the middle to rear portion of the tongue. The position of the tongues surface is manipulated by large and powerful muscles in its root, which move it as a whole within the mouth. The linguistically important dimensions of movement are generally the ranges [front - back] and [high - low]. Figure 2.2 shows a schematic characterization of English vowels in terms of relative tongue positions.

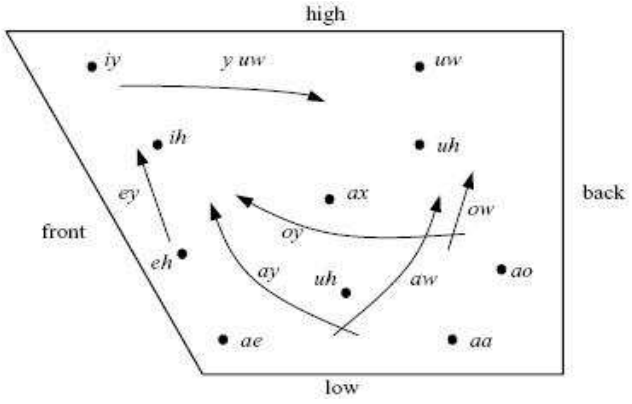

Figure 2.2: Relative tongue positions of English vowels[1]

Consonants

The word consonant comes from Latin and means *sounding with* or *sounding together*, the idea being that consonants don't sound on their own, but occur only with a nearby vowel, which is the case in Latin. This conception of consonants, however, does not reflect the modern linguistic understanding which defines consonants in terms of vocal tract constriction. Consonants, as opposed to vowels, are characterized by significant constriction or obstruction in the pharyngeal and/or oral cavities. When the vocal folds vibrate during phoneme articulation, the phoneme is considered voiced, otherwise it is unvoiced. Vowels are voiced throughout their duration. Some consonants are voiced, others are not. Each consonant can be distinguished by several features:

- The manner of articulation is the method that the consonant is articulated, such as nasal (through the nose), stop (complete obstruction of air), or approximant (vowel like);
- The place of articulation is where in the vocal tract the obstruction of the consonant occurs, and which speech organs are involved;
- The phonation of a consonant is how the vocal cords vibrate during the articulation;
- The voice onset time (VOT) indicates the timing of the phonation. Aspiration is a feature of VOT;
- The airstream mechanism is how the air moving through the vocal tract is powered;
- The length is how long the obstruction of a consonant lasts;
- The articulatory force is how much muscular energy is involved.

2.1.2 Background

The main problem in phoneme recognition is that phoneme pronunciations are different from one word to another, and also depend on the position of the phoneme in the same word. Whether this small acoustic unit is at the beginning, in the middle or at the end of a word matters when it comes to the automatic detection of these units. This is besides the variability in the phoneme pronunciation from one person to the other. Due to its important role in speech recognition[5], the problem

of phoneme recognition has been tackled over the years by many researchers. In the literature, there have been two main approaches to deal with this problem: statistical and neural network based approach.

Statistical based approaches

These type of approaches have been widely used to deal with the phoneme recognition problem. They can be split into two major categories: pattern-matching techniques and stochastic-based methods. In pattern-matching techniques, there is a reference for each class and each test pattern is compared to all of these references. Dynamic programming is employed to perform pattern-matching[6, 7]. In stochastic-based techniques, the recognition decision is made based on probabilistic rules. The Hidden Markov model is the most important and widely used method in this approach.

Neural network based approaches

The best phone recognition rate ever reached has been produced by Robinson in 1991[8]. He kept improving the system until he reached a rate of 80% and a frame-by-frame accuracy of 70.4% in 1994[9]. His system used a time delayed recurrent neural network. Since TDNNs can capture temporal relationships between acoustic events, they are powerful networks well-suited to performing phoneme recognition.

Chen and Jamieson reported almost the same results based on an extensive set of experiments[10]. They used an almost identical approach to Robinsons RNN, but included a new criterion function that enabled them to directly maximize the frame classification accuracy. Even though their phoneme recognition rate of 79.9% was slightly lower than Robinsons result and their best frame classification result was 74.2%.

There is as well new work emerging using a Support Vector Machine (SVM) type of classifier. Clarkson in [11] created a multi- class SVM system to classify the phonemes. The reported results of 77.6% are good and show the potential of SVM in speech recognition. In [12], the practical issues of training SVMs in the context of speech recognition were examined. Problems with non-converging training algorithms were solved, and two multi-class systems were created and shown to produce good results on a handpicked subset of the TIMIT speech corpus.

2.2 Confidence Measures In Speech Recognition

A confidence measure (CM) is a number between 0 and 1 that is applied to speech recognition output. A CM gives an indication of how confident we are that the unit to which it has been applied (e.g. a phrase, word, phone) is correct. Confidence measures are extremely useful in any speech application that involves a dialogue, because they can guide the system towards a more intelligent dialogue that is faster and less frustrating for the user. In fact, a low degree of confidence should be assigned to the outputs of a recognizer presented with an out-of-vocabulary(OOV) word or some unclear acoustics, caused by noise or a high level of background music. Both OOV words and unclear acoustics are a major source of recognizer error and may be detected by employing a confidence measure as a test statistic in a statistical hypothesis test. Nowadays, to a certain degree, the capability to evaluate reliability of speech recognition results has been regarded as a crucial technique to increase the usefulness and *intelligence* of an Automatic Speech Recognition(ASR) system in many practical applications.

Generally speaking, all methods proposed for computing confidence measures in speech recognition can be roughly classified into three major categories[13]. First, a large portion of works aim to compute confidence measures based on a combination of the so-called predictor features, which are collected during the decoding procedure and may include acoustic as well as language information about recognition decisions. Then all predictor features are combined in a certain way to generate a single score to indicate the correctness of the recognition decision. Second, it is well known that the posterior probability in the standard maximum a posteriori (MAP) decision rule is a good candidate for CM in speech recognition since it is an absolute measure of how good the decision is. However, it is very hard to estimate the posterior probability in a precise manner due to its normalization term in the denominator. In practice, many different approaches have been proposed to approximate it, ranging from simple filler-based methods to complex word-graph-based approaches. Third, under the name of utterance verification (UV), lot of research have been conducted to verify the claimed content of a spoken utterance. The content can be hypothesized by a speech recognizer or keyword detector or human transcriber. Under the framework of utterance verification (UV), the CM problem can be formulated as a statistical hypothesis testing[13].

2.2.1 Combination Of Predictor Factors

A predictor feature is a feature that is informative enough to distinguish correctly recognized words from other recognition errors. So for a given feature, if the probability distribution function (pdf) for correct words is distinct from the wrongly recognized ones, the feature can be used as a predictor. Usually, the predictor features have to be collected within the recognition process at levels of acoustics, language model, syntax, and semantics. Some of the commonly cited features in literature include:

- *Pure normalized likelihood score related*: acoustic score per frame.
- *Duration related*: HMM state duration, phoneme duration, word duration.
- *Language model (LM) related*: LM score, LM back-off behavior, etc.
- *N-best related*: count in the N-best list, N-best homogeneity score (the weighted ratio of all paths passing through the hypothesized word in N-best list), top N recognition scores, top $N-1$ difference in adjacently ranked recognition scores, etc.
- *Acoustic stability*: a number of alternative hypotheses are generated based on different language model weights in decoding and the acoustic stability of any given word is defined as the number of times the word occurs in the list divided by the number of alternatives in the list.

More features can be found in [13, 14, 15]. According to the literature, it is very hard to find a single ideal feature. That's why combinations of several predictor features have been attempted to improve performance. Many combinational models have been reported in the literature such as: single or mixture Gaussian classifier[16], neural networks[14], support vector machine[17] and many others.

2.2.2 Posterior Probability

An automatic speech recognition algorithm is usually formulated as a pattern classification problem using the (maximum a posteriori) (MAP) decision rule to find the most likely sequence of $\widehat{W}$ which achieves the maximum posterior probability $p(W|X)$ given any acoustic observation X.

$$\widehat{W} = \arg\max_{W\in\Sigma} p(W|X) = \arg\max_{W\in\Sigma} \frac{p(X|W).p(W)}{p(X)} = \arg\max_{W\in\Sigma} p(X|W).p(W) \quad (2.1)$$

where Σ is the set of all possible sentences, $p(W)$ is the probability of W evaluated from the language model and $p(X|W)$ is the probability of observing X given that W is the underlying word sequence for X. Theoretically, the posterior probability $p(W|X)$ is a good confidence measure. But for most recognizers $p(X)$ is discarded as is the case in (2.1) because it is a constant for all the words. This is why the raw ASR output scores are not reliable to judge the recognition performance. But if these scores get normalized by $p(X)$ they can serve as a good confidence measure. Unless some assumptions or approximations are adopted, the computation of the $p(X)$ is very hard. Most of the work in this approach focuses on the computation of the normalizing factor.

2.2.3 Utterance Verification

Using the confidence measure as utterance verification was mainly motivated by the *speaker verification problem*. Under this framework, the confidence measure problem is formulated as a statistical hypothesis testing problem. For an acoustic sequence X, the ASR recognizes the word W which is represented by the Hidden Markov Model(HMM) λ_W. The utterance verification examines the reliability of the hypothesized recognition result. Let:

H_0: X is correctly recognized and comes from λ_W.
H_1: X is wrongly classified and is not from λ_W.

Then, testing H_0 against H_1 determines whether the recognition result should be accepted or rejected. The *Likelihood Ratio Testing* expressed as (LRT) $\frac{p(X|H_0)}{p(X|H_1)}$ is compared to a decision threshold. The LRT-based utterance verification provides a good theoretical formulation to address confidence measure problems in ASR. Further details can be found in [13, 15].

It is well known that good confidence measures will largely benefit a variety of ASR applications, to intelligently reject non-speech noises, detect/reject out-of-vocabulary words, and to assist high-level speech understanding and dialogue management. However, confidence measures for ASR are an extremely difficult problem. In fact, even today's best available measures are not good enough to support most of the cited applications.

2.3 Keyword Spotting Techniques

Most of the techniques are filler model-based. In fact, the idea is to try to find a way to model as good as possible the out of vocabulary words so that whenever the user says something outside the grammar scope, it is directly captured by the filler model and discarded.

The work in [2] presents an approach for modeling and recognizing out-of-vocabulary (OOV) words in a one-stage recognizer. The recognizer is word-based and is augmented with an extra out-of-vocabulary word model which enables OOV words to be recognized. The OOV model used is phoneme-based and therefore any OOV word can be realized as an arbitrary sequence of phones.

There are basically three different problems related to OOV words:

- The detection of the OOV word in an utterance;
- The accurate detection of the units constituting the OOV word;
- The probable conversion of these units to actual words.

Most of the keyword spotting systems address the first problem. The most common approach is to use a garbage/filler model in order to absorb all the OOV words. Usually, the OOV words absorbed by the filler are of poor interest. The work presented in [2] differs from others in keyword spotting systems. In fact, these absorbed OOV words are of great importance because they will be used for recovering the actual spoken words. Details on how the recovery of the words is done are not provided in the paper since the research is still going on. The paper only presents the ground on which they are building such a system. The system consists of coupling a baseline word recognizer with an OOV recognizer (a phone recognizer in fact since the OOV model is phoneme based). Also, the transition to the filler model word can be controlled via a penalty (which is related to the probability of observing an OOV word). The performance of this hybrid recognizer was evaluated on the weather information domain with one test set containing only In-vocabulary data, and another containing OOV words. On the in vocabulary test set, the recognizer had an OOV insertion rate of only 1.3% and degraded the baseline word error rate from 10.4% to 10.7%. On the OOV test set, the recognizer was able to detect nearly half the OOV words (47% detection rate). So with a very simple generic word model, the system was able to detect half of the OOV words with a very small degradation in the word error rate.

The work in [3] presents a novel keyword spotting method: a combination of a Finite State Grammar (FSG) and a filler model, the two conventional technologies of keyword spotting. This mixed grammar model incorporates both a priori knowledge (from the FSG) and the capability of covering all possible forms in real speech. This combination has a better performance than a filler model based keyword spotting system and is more robust than a simple FSG-based keyword spotting system. In fact, when the size of the keyword set increases to several hundred, FSG-based systems still work efficiently for the already defined sentences in the grammar, but the performance deteriorates drastically for the undefined structures. This is beside the fact that the recognition speed decreases whenever we increase the scale of the non-keyword set in the grammar.

Filler model-based keyword spotting systems behave well for a small keyword table, but severely degrades as the keyword list gets bigger. Moreover, increasing the number of filler model is not a practical solution to solve this problem. A priori information must be introduced to help distinguish between keywords and non keywords. This explains the power of the combination of these two concepts, since their advantages and disadvantages are complementary. With the proposed system, only the most frequently used forms need to be described in the grammar. The size of the non keyword-set significantly decreases and filler models can absorb the non keywords that are not involved in the non-keyword set. As such the search is easier than simpler for FSG systems: only a few non-keywords are needed and the filler model will catch all the outbursts in the sentences. The proposed system has been tested for English and Chinese. A filler model of 6 Hidden Markov Model states with 6 Gaussian mixtures in each state has been used. The system has been compared to a filler model based keyword spotter, and results showed that for complex sentences the proposed system outperforms the filler model-based spotter by 25%.

Following this brief overview on the three pertinent areas of research, namely confidence measure, phone classification and keyword spotting systems, we present our approach in the next chapter.

Chapter 3

System Architecture And Proposed Techniques

In this chapter, we present the designed system architecture as well as the different techniques that have been used to implement the different components of the proposed system. The first section describes the system components. Then the phoneme classifier part is detailed in section 2 and finally the confidence measure technique is presented.

3.1 System Architecture

3.1.1 Specification

One of the targeted applications of the system presented in this thesis is keyword spotting. At the end of the day, we would like our system to be able to distinguish clearly between a keyword and a garbage word (out of vocabulary). Robustness and reliability are key features here. In fact, our module should output the right decision as often as possible and it has to produce the same type of output for the same inputs. In order to be able to judge whether a given ASR output is correct or not, we need to rely on the underlying mechanism and techniques of the speech recognition by itself. The most widely used parameter for this purpose is the ASR confidence. The desired output of our system is a high confidence value for correct words and a lower confidence for incorrect output of the ASR. In this context, the term correct implies that the word has been definitely spoken by the user, whereas

an incorrect token corresponds to a word that has been falsely mapped to an in-vocabulary item. False mapping occurs in one of the following two cases:

- A keyword is falsely mapped to another in vocabulary token;
- A garbage word is falsely mapped to a keyword.

We do not take into account the fact that a keyword gets falsely mapped to a garbage word, because the end system that we are targeting is a one which is a filler model free keyword spotting. That is, we do not provide the ASR with a filler model to capture the OOV words from the user utterance. When compared to the ASR confidence measures, our module should generate for correct words an equal or higher confidence measure, and a low value for the falsely mapped keywords. In this case, we can affirm that our system can distinguish more efficiently the right from the wrong ASR output.

3.1.2 System Description

The system we are proposing is mainly composed of two components:

- Phoneme classification module;
- Confidence measure module.

Feature vectors are extracted from the user speech. These features are then passed to the ASR system for recognition. Among the ASR outputs, we extract word level confidences, recognized words and the aligned frame wise phones as described in Figure 3.1. The aligned phones will allow us to pass the corresponding frames to the next module, the phoneme classifier. We have chosen to build a Gaussian mixture model based classifier for the English phonemes. Given a set of frames, this GMM-based classifier outputs the phoneme that best matches the input data. Details of the algorithm will be provided later in this chapter. Classifier outputted phones of a given word are then passed to our confidence measure module in order to deduce a confidence value. This newly computed confidence is the outcome of our system and will be ultimately used to check if a word is either correct or incorrect.

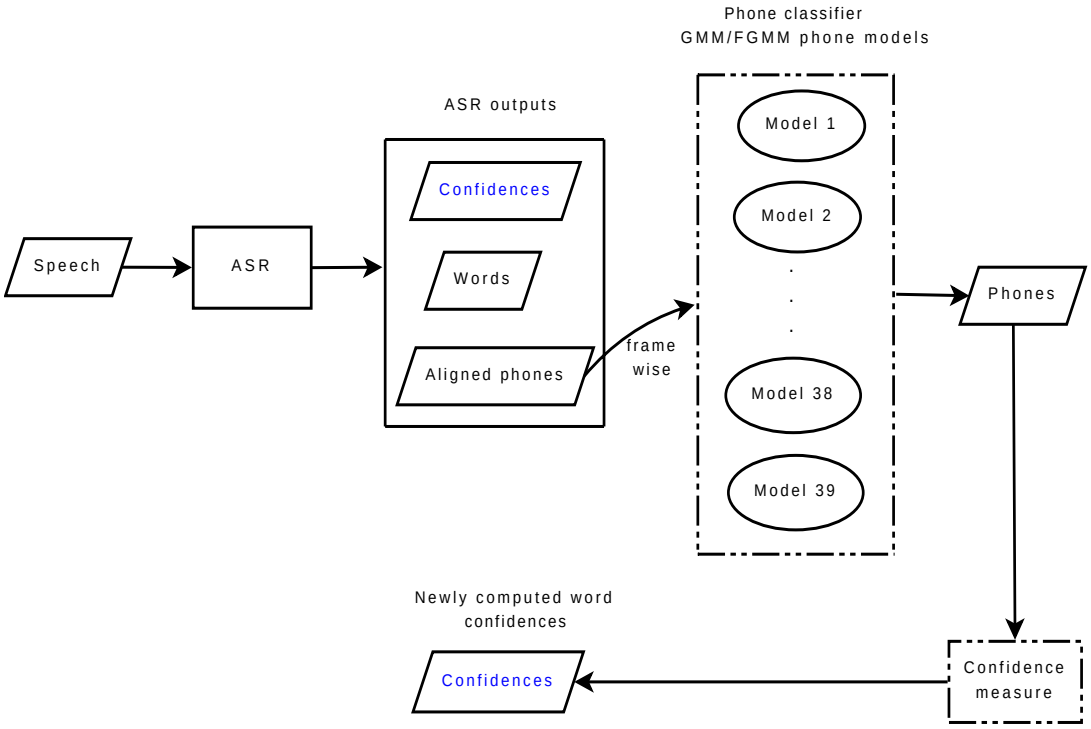

Figure 3.1: System architecture

3.2 Phoneme Classification

Given a sequence of acoustic observations and the full phoneme set, the phoneme classifier module is intended to classify these observations to one of the target phonemes. In this section, we provide the technical details of the phone classifier. A background and the detailed algorithms will be presented.

A Gaussian mixture modeling of the English phonemes has been attempted. We have chosen this particular type of technique because its performance has been proven in several areas such as speaker identification, speaker recognition, emotion recognition, etc. This technique is known to be reliable and robust enough where there is a high degree of overlapping between the different classes. And that is exactly the case here. In fact, several phones present a lot of similarity with each other, which make their classification a complex problem. To emphasize the difficulty of the problem, Figure 3.2 shows for the two first dimensions of frames, the high overlapping between two target phones. Also, we have adopted this technique

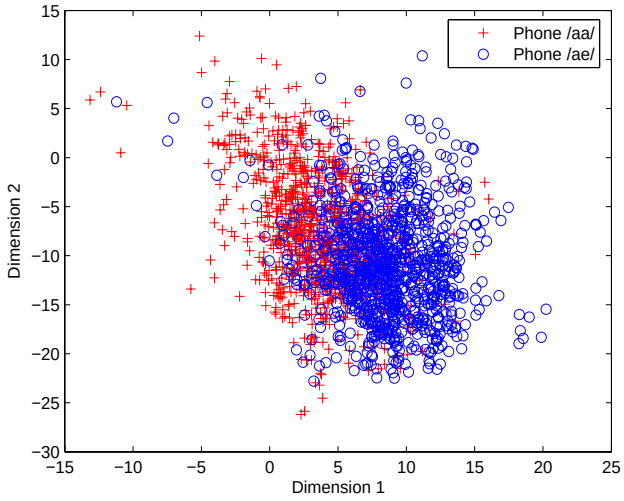

Figure 3.2: Two first dimensions plot for both phone /aa/ and /ae/

since it is faster and easier compared to other techniques such as neural networks. In fact, neural networks training tasks take considerable time compared to GMM training. GMMs model the probability density function of observed variables using a multivariate Gaussian mixture density. Given a series of inputs, it refines the weights of each distribution through expectation-maximization algorithms[18]. We outline next the GMM and the fuzzy GMM algorithms in details.

3.2.1 Gaussian Mixture Models Algorithm

This algorithm has been used in [19]to deal with the speaker identification problem. We kept the same notations as in [19]. Let $X = x_1, x_2, ..., x_T$ be a set of T feature vectors from the voice data of a person, each of which is a d-dimensional feature vector extracted by digital speech signal processing. The likelihood of the GMM is:

$$p(X|\lambda) = \prod_{t=1}^{T} p(x_t|\lambda) \tag{3.1}$$

We can approximately model the distribution of theses vectors by a mixture of Gaussian densities:

$$p(x_t|\lambda) = \sum_{i=1}^{c} w_i N(x_t, \mu_i, \Sigma_i) \tag{3.2}$$

where $\lambda = \{w_i, \mu_i, \Sigma_i\}$ denotes a set of model parameters, w_i and $N(x_t, \mu_i, \Sigma_i)$, $i = 1, ..., c$, are the mixture weights and the d-variate Gaussian component densities with mean vectors μ_i and covariance matrices Σ_i, respectively:

$$N(x_t, \mu_i, \Sigma_i) = \frac{\exp\{-\frac{1}{2}(x_t - \mu_i)'\Sigma_i^{-1}(x_t - \mu_i)\}}{(2\pi)^{d/2}|\Sigma_i|^{1/2}} \tag{3.3}$$

where $'$ stands for transpose and $|\ |$ stands for the discriminant of a matrix.
In training the GMM, these parameters are estimated such that they best match the distribution of the training vectors. The most widely-used training method is the *maximum likelihood*(ML) estimation, where a new parameter model $\overline{\lambda}$ is found such that $p(X|\overline{\lambda}) \geq p(X|\lambda)$. An auxiliary function Q is used for this task[20]:

$$Q(\lambda, \overline{\lambda}) = \sum_{i=1}^{c}\sum_{t=1}^{T} p(i|x_t, \lambda)\log[\overline{w}_i N(x_t, \overline{\mu}_i, \overline{\Sigma}_i)] \tag{3.4}$$

where $p(i|x_t, \lambda)$ is the a posteriori probability for acoustic class i,$i = 1, ..., c$ and satisfies:

$$\sum_{i=1}^{c} p(i|x_t, \lambda) = 1 \tag{3.5}$$

Setting derivatives of the Q function with respect to $\overline{\lambda}$ to zero, the following re-estimation formulas are found:

$$p(i|x_t, \lambda) = \frac{w_i N(x_t, \mu_i, \Sigma_i)}{\sum_{k=1}^{c} w_k N(x_t, \mu_k, \Sigma_k)} \tag{3.6}$$

$$\overline{\mu}_i = \frac{\sum_{t=1}^{T} p(i|x_t, \lambda)x_t}{\sum_{t=1}^{T} p(i|x_t, \lambda)} \tag{3.7}$$

$$\overline{\Sigma}_i = \frac{\sum_{t=1}^{T} p(i|x_t, \lambda)(x_t - \overline{\mu}_i)(x_t - \overline{\mu}_i)'}{\sum_{t=1}^{T} p(i|x_t, \lambda)} \tag{3.8}$$

$$\overline{w}_i = \frac{1}{T}\sum_{t=1}^{T} p(i|x_t, \lambda) \tag{3.9}$$

The training procedure of the Gaussian mixture modeling is summarized in algorithm **1**. Details of the k-means algorithms used for initialization can be found in

Algorithm 1 GMM training procedure

1: Given X as the training set of all the phones and M is the number of phones.
2: X^i contains the data for the phone i, where $1 \leq i \leq M$.
3: For each of the M subsets, train a GMM phone model as follows:
4: Generate $p(i|x_t, \lambda)$ at random stratifying (3.5) by running k-means for a few iterations.
5: Calculate phone model $\lambda = \{w_i, \mu_i, \Sigma_i\}$ using equations (3.7), (3.8) and (3.9).
6: Update $p(i|x_t, \lambda)$ using (3.6).
7: Compute Q
8: If the change of Q compared with that in the previous iteration is less than a preset value then stop else go to step 5.

section 3.2.3. For phoneme classification, let $\lambda_k,\ k = 1, ..., N$, denote N phoneme models, then a classifier is designed to classify X into N phoneme models by computing the log-likelihood of the unknown X given each phoneme model λ_k and select phoneme k^* if

$$k^* = \arg\underset{1\leq k\leq N}{max} \sum_{t=1}^{T} \log p(x_t|\lambda_k) \tag{3.10}$$

3.2.2 Fuzzy Gaussian Mixture Models Algorithm

Fuzzy GMM is the fuzzy c-means-based modification of the GMM. This fuzzy version of the GMM has been taken from [21]. The FGMM has been used in [21] in order to identify and classify cancer cells in different stages of the disease. Fuzzy c-means algorithm is presented in section 3.2.4. Given X as a set of T feature vectors, we define $U = \{u_{it}\}$ to be a fuzzy C partition of X, each u_{it} represents the degree of vector x_t to the ith mixture and is called the fuzzy membership function.

For $1 \leq i \leq C$ and $1 \leq t \leq T$, we have:

$$\begin{aligned} 1 \leq & \quad u_{it} \quad \leq 1 \\ \sum_{i=1}^{C} & \quad u_{it} \quad = 1 \\ 0 < \sum_{t=1}^{T} & \quad u_{it} \quad < T \end{aligned} \tag{3.11}$$

where C is the number of mixtures, $m > 1$ is a weighting exponent on each fuzzy membership u_{it} and is called *degree of fuzziness*.
The fuzzy objective function was proposed in [22] as follows:

$$J_m(U, \lambda) = \sum_{i=1}^{C} \sum_{t=1}^{T} u_{it}^m \; d_{it}^2 \tag{3.12}$$

The generalization of the objective function is done through the use of a fuzzy mean vector, a fuzzy covariance matrix and fuzzy mixture weight. To obtain these, since the density of the data in cluster i is proportional to the joint mixture density function $P(x_t, i|\lambda)$, the dissimilarity can be defined by the distance in (3.12) as follows:

$$d_{it}^2 = -\log P(x_t, i|\overline{\lambda}) = -\log\left[\overline{w}_i N(x_t, \overline{\mu}_i, \overline{\Sigma}_i)\right] \tag{3.13}$$

From (3.13) and (3.3), we have:

$$d_{it}^2 = -\log \overline{w}_i + \frac{1}{2}\log(2\pi)^d |\overline{\Sigma}_i| + \frac{1}{2}(x_t - \overline{\mu}_i)' \overline{\Sigma}_i^{-1} (x_t - \overline{\mu}_i) \tag{3.14}$$

Substituting (3.13) into (3.12) gives:

$$J_m(U, \mu, \Sigma, w; X) = -\sum_{i=1}^{C} \sum_{t=1}^{T} u_{it}^m \log \overline{w}_i - \sum_{i=1}^{C} \sum_{t=1}^{T} u_{it}^m \log N(x_t, \overline{\mu}_i, \overline{\Sigma}_i) \tag{3.15}$$

In order to minimize J_m, we need to minimize each term on the right hand side of (3.15).
For the first term, after using a Lagrange multiplier and having:

$$\sum_{i=1}^{C} \overline{w}_i = 1 \tag{3.16}$$

we obtain the fuzzy mixture weight as follows:

$$\overline{w}_i = \frac{\sum_{t=1}^{T} u_{it}^m}{\sum_{i=1}^{C}\sum_{t=1}^{T} u_{it}^m} \tag{3.17}$$

Minimizing the second term of (3.15) is obtained by using (3.3) and (3.14) and setting the derivative with respect to μ_i and Σ_i to zero for every $i = 1, ..., C$:

$$\sum_{t=1}^{T} u_{it}^m \overline{\Sigma}_i^{-1}(x_t - \overline{\mu}_i) = 0 \tag{3.18}$$

$$\sum_{t=1}^{T} u_{it}^m [\overline{\Sigma}_i - (x_t - \overline{\mu}_i)(x_t - \overline{\mu}_i)'] = 0 \tag{3.19}$$

From (3.18) and (3.19) we obtain the fuzzy mean vector as well as the fuzzy covariance matrix:

$$\overline{\mu}_i = \frac{\sum_{t=1}^{T} u_{it}^m x_t}{\sum_{t=1}^{T} u_{it}^m} \tag{3.20}$$

$$\overline{\Sigma}_i = \frac{\sum_{t=1}^{T} u_{it}^m (x_t - \mu_i)(x_t - \mu_i)'}{\sum_{t=1}^{T} u_{it}^m} \tag{3.21}$$

where the u_{it} is computed using (3.12) since it is derived by minimizing J_m with $\{u_{it}\}$ as variables:

$$u_{it} = \left[\sum_{k=1}^{C} \left(\frac{d_{it}}{d_{kt}}\right)^{\frac{2}{m-1}}\right]^{-1} \tag{3.22}$$

The training procedure of the fuzzy Gaussian mixture modeling is summarized in algorithm **2**. For phoneme classification, let $\lambda_k,\ k = 1, ..., N$,denote N phoneme models, then a classifier is designed to classify X into N phoneme models by computing the fuzzy objective function of the unknown X given each phoneme model

Algorithm 2 FGMM training procedure

1: Given X as the training set of all the phones and M is the number of phones.
2: X^i contains the data for the phone i, where $1 \leq i \leq M$.
3: For each of the M subsets, train a FGMM phone model as follows:
4: Generate u_{it} at random stratifying (3.11) by running fuzzy c-means for a few iterations.
5: Calculate phone model $\lambda = \{w_i, \mu_i, \Sigma_i\}$ using equations (3.17),(3.20) and (3.21).
6: Update u_{it} using (3.22).
7: Compute $J_m(U, \lambda)$
8: If the change of $J_m(U, \lambda)$ compared with that in the previous iteration is less than a preset value then stop else go to step 5.

λ_k and select phoneme k^* if

$$k^* = \arg\underset{1 \leq k \leq N}{max} \sum_{t=1}^{T} \left[\sum_{i=1}^{C} [-\log p(x_t, i|\lambda_k)]^{\frac{1}{1-m}} \right]^{1-m} \tag{3.23}$$

3.2.3 K-means Clustering Algorithm

K-means was first introduced by Mac Queen[23]. It is one of the simplest unsupervised clustering algorithms. The procedure follows a simple way to classify a given data set through a certain number of clusters (assume k clusters) fixed a priori. The main idea is to define k centroids, one for each cluster. The next step is to take each point belonging to a given data set and associate it to the nearest centroid. When no point is pending, the first step is completed and an early grouping is done. At this point we need to re-calculate k new centroids as barycenters of the clusters resulting from the previous step. After we have these k new centroids, a new binding has to be done between the same data set points and the nearest new centroid. A loop has been generated. As a result of this loop we may notice that the k centroids change their location step by step until no more changes are done. In other words the centroids do not move any more[24]. This algorithm aims at minimizing the objective function defined in equation 3.24.

$$J = \sum_{j=1}^{k} \sum_{i=1}^{n} \|x_i^{(j)} - c_j\|^2 \tag{3.24}$$

3.2.4 Fuzzy C-means Clustering Algorithm

First introduced by Dunn[25] and then modified by Bezdek[26], fuzzy c-means is a clustering technique that allows one piece of data to belong to more than one cluster at the same time. It aims at minimizing the objective function defined by equation 3.25.

$$J = \sum_{i=1}^{N}\sum_{j=1}^{C} u_{ij}^{m}\|x_i^{(j)} - c_j\|^2 \qquad , \qquad 1 < m < \infty \tag{3.25}$$

where u_{ij} is the degree of membership of x_i in the cluster j. Fuzzy partitioning is carried out through an iterative optimization of the objective function shown above[27], with the update of membership u_{ij} and the cluster centers c_j by equation 3.26.

$$u_{ij} = \frac{1}{\sum_{k=1}^{C}\left(\frac{\|x_i^{(j)} - c_j\|}{\|x_i^{(j)} - c_k\|}\right)^{\frac{2}{m-1}}} \qquad , \qquad c_j = \frac{\sum_{i=1}^{N} u_{ij}^{m}.x_i}{\sum_{i=1}^{N} u_{ij}^{m}} \tag{3.26}$$

This iteration will stop when $max_{ij}\left\{\left|u_{ij}^{k+1} - u_{ij}^{k}\right|\right\} < \epsilon$, where ϵ is the termination criterion.

3.3 Confidence Measure

This module aims at computing a new confidence value for each recognized word by the ASR. Figure 3.3 shows that the predicted phones from the phone classifier as well as the original ASR phones are the major inputs to this module. The computation of the confidence score is straightforward. We have basically adopted two types of measures:

- Hard measure: For each recognized word, we compute the number of perfect phone matches between the ASR and the GMM-based classifier as in Figure 3.3. The obtained number is normalized by the total number of phones of the given word.

$$confidence = \frac{\#perfect\ match}{\#phones} \tag{3.27}$$

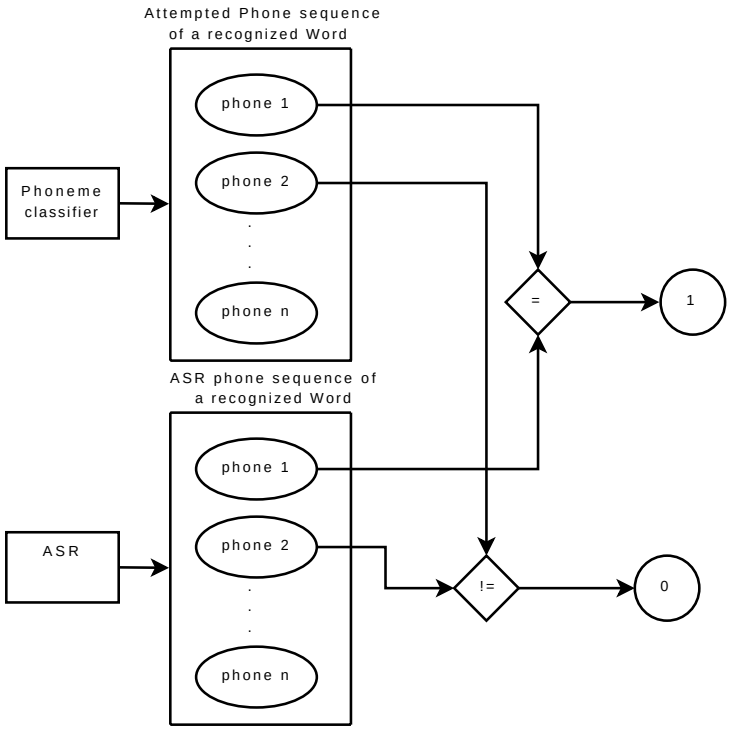

Figure 3.3: Confidence measure module

- Weighted measure: Whenever we have a perfect match between the ASR and the GMM classifier, we sum up the length in terms of number of frames of the corresponding matched phone. The obtained value is normalized by the total length of the word.

$$confidence = \frac{length\ of\ perfect\ match}{word\ length} \tag{3.28}$$

3.3.1 Integration Of The Classifier With Confidence Measure

Now that both modules have been presented, we outline their integration. Figure 3.4 describes the integration flow. The input to our system are the aligned phone/frames from the ASR output, which will be fed to the FGMM/GMM based classifier, and the actual ASR outputted phones of the corresponding recognized word which will be used by the confidence measure module. The classifier outputs

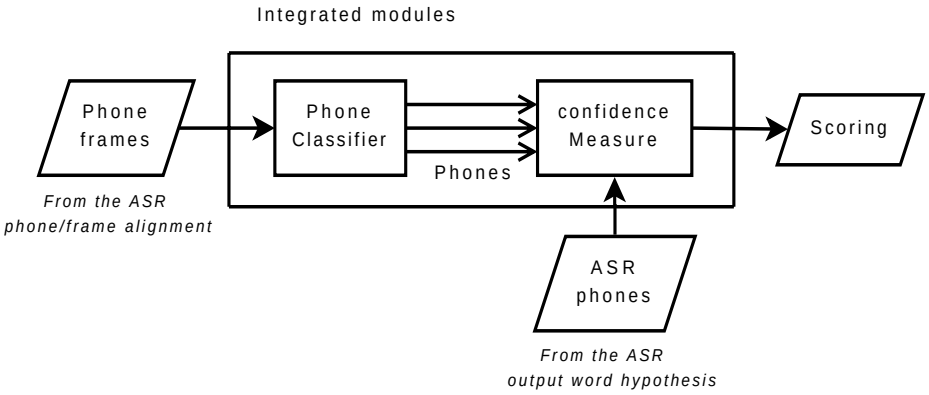

Figure 3.4: Classifier and confidence measure integration

the best phone hypothesis corresponding the the input frames, then these phones are fed along with the ASR phone hypothesis to the confidence module to compute the new scoring of a specific word.

3.4 Conclusion

We presented in this chapter the architecture of our system. Underlined used techniques and algorithms have been detailed as well. The next chapter will present the implementation aspect of the proposed approach and the experimental results will be then interpreted.

Chapter 4

Experimental Results And Interpretations

In this chapter, we describe the experimental framework used to implement the proposed approach.

4.1 Experimental Framework

4.1.1 The JULIUS ASR System

Julius[28] is a high performance continuous speech recognition software based on word N-grams. It is able to perform recognition at the sentence level with a vocabulary in the tens of thousands. Julius realizes high-speed speech recognition on a typical desktop PC. It performs at near real time and has a recognition rate of above 90% for a $20,000$-word vocabulary dictation task. The best feature of the Julius system is that it is multipurpose. By recombining the pronunciation dictionary, language and acoustic models one is able to build various task specific systems. The Julius code also is open source so one should be able to recompile the system for other platforms or to alter the code for one's specific needs. In this thesis, we have used the Julian system[29], a continuous speech recognition parser based on finite state grammar. It is exactly the same engine as Julius expect that Julian uses a grammar instead. High precision recognition is achieved using a two pass hierarchical search. Julian can perform recognition on microphone input, audio files, and feature parameter files. Also, as standard format acoustic models and grammar based language models can be used, these models can be changed to

perform recognition under various conditions. The maximum vocabulary is 65,535 words.

4.1.2 ATT Text To Speeh

In order to generate audio data from the natural language sentences, we have used ATT Natural voices text-to-speech engine (TTS)[30]. The TTS engine provide synthesis services in multiple languages for application builders creating desktop applications. High-quality male and female voices are included in 8 KHz μlaw and CCITT G.711 *alaw* for telephony applications. Additional voices and higher quality 16 KHz voices are available for non-telephony applications. Further details on this TTS engine can be found in [30].

4.1.3 Hidden Markov Model Toolkit (HTK)

HTK is a toolkit for building Hidden Markov Models (HMMs). HMMs can be used to model any time series and the core of HTK is similarly general-purpose [31]. However, HTK is primarily designed for building HMM-based speech processing tools, in particular recognizers. The tool kit is open source and is available in [32]. We mainly used this toolkit for:

- ASR performance evaluation: for each given sentence, the number of deleted, inserted, substituted and correct words is computed and an overall accuracy and correctness is outputted;
- Transcript force alignment: phoneme/frame alignment is needed for the transcript testing data. We use the HTK to do recognition of wave files corresponding to the testing data and get forced aligned recognition results that will be used later for performance evaluation

4.1.4 TIMIT Speech Corpus

The data set used for training and testing the FGMM/GMM models is the TIMIT database[33]. It is a corpus of high quality continuous speech from North American speakers, with the entire corpus reliably transcribed at the word and surface phonetic levels. The TIMIT corpus of read speech has been designed to provide speech data for the acquisition of acoustic-phonetic knowledge and for the development and

evaluation of automatic speech recognition systems. TIMIT has resulted from the joint efforts of several sites under sponsorship from the Defense Advanced Research Projects Agency - Information Science and Technology Office (DARPA-ISTO). The text corpus design was a joint effort among the Massachusetts Institute of Technology (MIT), Stanford Research Institute (SRI), and Texas Instruments (TI). The speech was recorded at TI, transcribed at MIT, and has been maintained, verified, and prepared for CD-ROM production by the National Institute of Standards and Technology (NIST).

The speech is parameterized as 12 Mel-frequency coefficients (MFCC) plus energy for $25ms$ frames, with a $10ms$ frame shift. Note that delta and acceleration coefficients can be derived from these 13 coefficients, resulting in a total of 39 coefficients. The target labels consists of 40 different classes, representing 40 phonemes from the English language. This is a configuration commonly used in speech recognition[34]. The corpus is divided into 3648 training utterances and 1344 test utterances. No speakers from the training set appear in the test set, making it 100% speaker-independent. TIMIT contains sentences spoken by speakers from 8 major dialect regions of the United States (New England, Northern, North Midland, South Midland, Southern, New York city, Western, Army brat where the speakers moved around a lot during their childhood).

4.2 Phone Classification

In this section, we tackle the training and the evaluation of the Fuzzy GMM and the GMM models that will serve as the phone classifiers. We start by presenting the experimental setup, then results will be provided as well as the interpretations and finally we conclude the section by a summary.

4.2.1 Experimental Setup

For the training part as well as for the classification part of both algorithms GMM and FGMM, we have used the TIMIT database, the de facto speech database for evaluating speech recognition systems. We have used as well the commonly used features vectors in speech systems, the MFCC features.

Feature extraction As for most of the pattern recognition problems, we need to extract a set of features from the audio raw data that represents all the

dynamics and variations of the input. These features will serve as the input set for training the Gaussian models that will model the phonemes. So the choice of these features will affect enormously the overall performance of the system. There are plenty of features we can use[1], but we cannot use all of them since the training data is limited. Adding new features doesn't mean that the error will decrease systematically. This is not due to the fact that the feature we added is poor, but rather that our data is insufficient to reliably model all the features. The first feature we use is speech waveform. In general, time-domain features are much less accurate than frequency-domain features such as the mel-frequency cepstral coefficients (MFCC). This because many features such as formants, which are useful in classifying vowels, are better characterized in the frequency domain with a low dimension feature vector. Temporal changes in the spectra play an important role in the human perception. In order to capture this information, we use the delta coefficients that measure the change in coefficients over time. For this thesis, we used the typical feature vector for speech recognition, the 39 MFCC coefficients :

- 13th order MFCC c_k
- 13th order first order delta MFCC computed from $\Delta c_k = c_{k+1} - c_k$
- 13th order second order delta MFCC computed from $\Delta\Delta c_k = \Delta c_{k+1} - \Delta c_k$

Further details on these features and how to compute them can be found in details in [1, 35]. A procedure to extract MFCC coefficients of a given speech is given in appendix *D*.

Data collection Now that we have extracted the features from the wave files of the TIMIT database, we need to collect all the frames (feature vectors) of each phone in the whole database. This data will serve for training and testing the FGMM/GMM models. In order to do this data collection, we will use the alignment provided in the database, alignment in terms of frames (i.e. for every phone, we have the corresponding the first and the last frame). We used all the 8 dialects in the database to collect the data for each of the 39 phones listed in appendix *C*.

4.2.2 Results And Interpretations

Once we have the data for each phoneme, we can launch the training algorithm to optimize the models representing each of the phones. According to the theoretical

considerations presented in chapter 3, we present now the results of phoneme classification using Gaussian Mixture Modeling and Fuzzy Gaussian Mixture Modeling.

For the GMMs, the initialization of the parameters is done using 5 iterations of the k-means algorithm, and the fuzzy c-mean algorithm run for 5 iterations has also been used to initialize the model parameters of the fuzzy GMMs. We have chosen the degree of fuzziness $m = 1.03$. In order to evaluate the performance of our FGMM/GMM based classifier, we built a confusion matrix. A confusion matrix[36], contains information about actual and predicted classifications done by a classification system. Performance of such systems is commonly evaluated using the data in the matrix. Each column of the matrix represents the instances in a predicted class, while each row represents the instances in an actual class. One benefit of a confusion matrix is that it is easy to see if the system is confusing two classes (i.e. commonly mislabeling one as another). We have trained models with different components. We have chosen to use diagonal covariances. Table 4.1 shows the classification rate for both GMMs and FGMMs. The best configuration

Components	**GMM (%)**	**FGMM (%)**
8	42	42.51
16	47	58.51
32	56.39	57.87
64	64.21	66
128	65.70	67.29

Table 4.1: Classification rate for fuzzy GM phone models and GM phone models

is with 64 components. Although with 128 components we have obtained slightly better results, we decided to go with the 64-component GMMs and FGMM. In fact, training time as well as the testing time are multiplied by almost 2.5 times when using 128 components compared to the 64 components configuration. The fuzzy GMMs yield better performance in all model sizes which are $8, 16, 32$ and 64 compared to the GMMs.In table 4.2, the precision measure for all the phonemes is presented. The precision measure shows the proportion of correct prediction for every phone. We notice here that some of the phonemes have low precision. This is due in part to the size of the training corpus; for example the phone /zh/ is very rare and so the training data is small, thus the precision rate is low. Compared to other techniques, a 66% classification rate is acceptable for the task they are intended to do.

Phoneme	Precision %
/aa/	68.01
/ae/	78.7
/ah/	70.93
/ao/	65.65
/aw/	27.55
/ay/	61.96
/b/	69.91
/ch/	45.94
/d/	52.72
/dh/	70.29
/eh/	39.86
/er/	52.36
/ey/	50.88
/f/	78.09
/g/	65.81
/hh/	84.13
/ih/	54.19
/iy/	82.34
/jh/	34.09
/k/	83.03
/l/	84.65
/m/	67.13
/n/	74.80
/ng/	40.27
/ow/	40.48
/oy/	45.49
/p/	72.50
/r/	84.33
/s/	83.35
/sh/	82.14
/t/	58.21
/th/	32.37
/uh/	52.08
/uw/	46.66
/v/	56.63
/w/	77.81
/y/	76.59
/z/	63.92
/zh/	26.67

Table 4.2: Phoneme precisions using the 64 fuzzy mixture models

4.2.3 Preliminary Remarks

Using the TIMIT database, we have trained GMM and FGMM models for each of the 39 phone target classes. We have proved that this FGMM technique outperforms the conventional GMM by approximately 2%. The classification rate, however, is still low compared to the other techniques, especially the HMM based recognizers that perform a 78% phoneme recognition rate. In order to improve the classification results, we might need to try more features. In fact, formants are critical in the recognition of vowels, and we didn't use them for training our models, so it might be worth trying to improve the vowel classification rate. We can as well try other feature extraction techniques, like the RASTA (Relative Spectra), LPC (Linear Predictive Coding), etc.

4.3 Confidence Measure

In this section, we outline experiments and results preliminary to our second module, which does the confidence evaluation of the ASR output. We start the section by a presentation of the experimental framework, then the obtained results are discussed and we conclude with a summary.

4.3.1 Experimental Setup

We have used a call routing example in order to evaluate our system. The routing tree is presented in 4.1. The user chooses between one of the 6 products (nodes in the first level of the tree). Then, for each product, the user might ask for one of the 9 services (nodes in the second level of the routing tree). For esthetic purposes, only services for one product have been represented. All the products have the same 9 services. This call routing tree can be considered as fairly realistic and complex enough to base our analysis on this application. In fact, most of the routing trees in current use are 1-level trees and the number of nodes is around 10. This number decreases drastically when the application deals with a multilevel tree.

Data generation As we already mentioned before, the system will be tested with natural language sentences; the user will not be constrained. In order to express a request, the speaker can use a natural language sentence that contains one or more keywords. In order to expand the number of possible natural sentences, we allowed the speaker to use 4 synonyms to point to a specific second

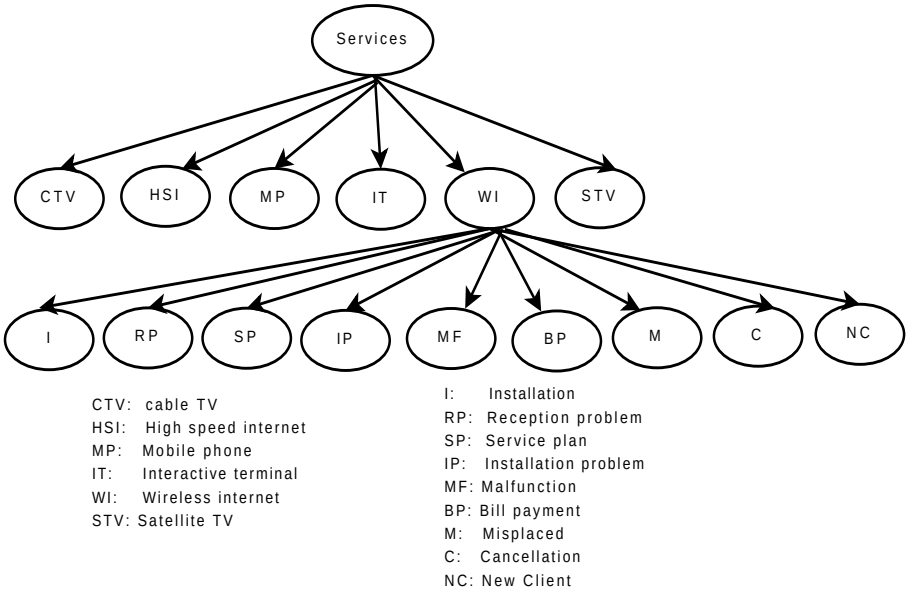

Figure 4.1: Call routing tree

level node. The list of synonyms for each node can be found in appendix *A*. Then we prepared some templates of natural language sentences to generate automatically natural sentences. In total, we generated 3601 possible natural sentences that a user might utter when calling on the call center. Each sentences contains two keywords, one from each level. Here are some natural sentences where the keywords have been emphasized:

> There is a *reception problem* with my *cable tv*.
> I want to speak to somebody regarding *installation problem* of my *high speed internet*.
> I have been waiting for 15 minutes to talk about my *satellite tv poor reception*.
> Is there a way to solve my *bill payment* problem of my *wireless internet*.

In total we have created 6 different templates because of the different context of the keywords. All the templates created to generate these sentences can be found in appendix *B*.

Tagging and alignment The grammar of the ASR only contains the keywords, no filler model is used here. The grammar is composed of the 6 product names

as well as the list of services along with their synonyms. After the ASR does the recognition of the 3601 audio files, we should parse each ASR output and tag each word. The tags can be:

- Correct (**C**) for words that are correctly spotted;
- Incorrect (**I**) otherwise i.e. false mapping.

In order for us to carry this tagging task, we need to force align the transcripts: frame/word alignment. We have used the *HVite* tool of the HTK toolkit to carry out this force alignment. Here is an example of transcript force alignment:

> ***Reference transcript:*** There is a *reception problem* in my *cable tv*.
> ***Force alignment:*** Table 4.3 shows a sample output after aligning the words from the reference transcript along with the corresponding frames in the audio file.

Begin	0	14	35	53	56	111	152	170	192	239	292
End	13	34	52	55	110	151	169	191	238	291	294
Word	sil	there	is	a	reception	problem	with	my	cable	tv	sil

Table 4.3: Transcript force alignment (word/frame)

Once we have the reference transcripts aligned, tagging the ASR output is now straightforward. We have chosen to tolerate a margin of 20 frames to the left or/and right. Knowing that each second contains 100 frames, the tolerance margin is only 0.2 seconds. So if the recognized word (ASR output) is aligned with the reference word with a margin of $+/-$ 20 frames, the word is tagged as *correct*, otherwise it is an *incorrect* word. Here is an example of the tagging procedure:

> Let us suppose the reference sentence is the one defined in table 4.3.
> ***Reference transcript:*** There is a *reception problem* in my *cable tv*.
> ***ASR output:*** not working *reception problem* to cut lost *cable tv*.
> ***Tagging:*** Table 4.4 shows the result of the tagging. Note the frame shifting is always below the 20 frames threshold.

Confidence	Begin	End	Word	Tag
1.000	0	11	sil	C
0.670	12	29	not	I
0.000	30	50	working	I
0.981	51	106	reception	C
1.000	107	149	problem	C
0.928	150	159	to	I
0.486	160	172	cut	I
0.985	173	192	lost	I
1.000	193	227	cable	C
1.000	228	291	tv	C
1.000	292	293	sil	C

Table 4.4: Tagging the ASR output

Now that the data is ready, we can use the trained model and our confidence measure approach to evaluate the Julius confidence measure.

4.3.2 Results And Interpretations

The FGMM/GMM models trained previously are now used to evaluate the ASR output. We feed the frames corresponding to each phone of each word of the ASR output, to the 39 models in order to pick the best phone that matches the input frame data. Table 4.5 shows the result for the word *reception* of the same reference sentence discussed in the previous section. In chapter 3, we have defined two types

Begin	End	ASR	GMM
51	59	/r/	/r/
60	65	/ah/	/aa/
66	75	/s/	/s/
76	83	/eh/	/ae/
84	89	/p/	/p/
90	95	/sh/	/sh/
96	99	/ah/	/sh/
100	106	/n/	/d/

Table 4.5: ASR vs GMM phone output

of confidence measures: the hard and the weighted measures. For each of these two measures, we computed (for both the correct and incorrect words), the number of times where our confidence is below or above the Julius confidences. Table 4.6

Output	Hard		Weighted	
	C	I	C	I
GMM	4.94%	89.31%	4.025%	89.42%
ASR	95.06%	10.69%	95.975%	10.58%
FGMM	5.21%	89.67%	4.2%	90.13%
ASR	94.79%	10.33%	95.8%	9.87%

Table 4.6: ASR vs GMM/FGMM confidence comparisons

shows the results obtained for both type of measures. For the incorrect columns labeled as (**I**), we computed the number of times where our confidence measure is ***lower*** than the ASR confidence. And for the correct columns labeled as (**C**), we computed the number of time where our confidence measure is ***higher*** than the ASR one. With this type of statistics, we can detect if our confidence measure approach differentiates better between correct and incorrect words.

For the incorrect words spotted by the ASR and when using the GMMs for the phoneme classification, 89.31% of the time the hard measure of the confidence is lower than the actual ASR confidence value. This percentage increases slightly when using the fuzzy GMM models. It is clear that our system ranks better the falsely mapped words. But this is not the case for the correctly spotted words. In fact only 4.94% of the times our confidence is larger than the ASR confidence. When using the FGMMs for the phoneme classification, this percentage slightly increases to reach 5.21%. There is not much difference for the second type of measure, especially with regards to the correctly spotted words. That is, using our technique we are only 4.2% of the times above the ASR confidence values. These results are inconclusive at this early stage of the analysis. Noting that ASR confidences are usually high whether it is for correctly spotted words or falsely mapped ones, we need to provide a fairer way for carrying out this analysis. This will be done by computing the deviations from the average confidences for correct and incorrect words. We computed the average confidence for correct words and incorrect words. Results are shown in table 4.7. The column **C** corresponds to the mean confidence for all the correctly accepted words, whereas the column **I** corresponds to the mean confidence of the falsely mapped keywords. Then the overall confidence is the mean of the two columns **C** and **I**. The mean values in table 4.7 show clearly the difference

Output	Hard			Weighted		
	Correct	Incorrect	Overall Mean	Correct	Incorrect	Overall Mean
GMM	0.3492	0.1197	0.2344	0.1809	0.1121	0.1465
FGMM	0.3607	0.11	0.2353	0.186	0.098	0.142
ASR	0.9146	0.703	0.8079	0.9146	0.703	0.8079

Table 4.7: ASR vs GMM/FGMM average confidences

in the difference in the confidence values range between the ASR and our approach, which explains the low percentage for the correctly accepted word in table 4.6.

We can notice here as well that the two measurements have different distributions. So in terms of the deviations from the overall mean: Julius is 13% and our approach is 50% (either with GMM or with FGMM), showing a much larger differentiation. This is a very positive indicator for our system. That means that we distinguish easily and better between the falsely mapped words and the correctly spotted ones. Note that the weighted approach doesn't seem to perform better than the hard approach. To emphasize this aspect of different distributions, we have plotted the correctly accepted and the falsely mapped word confidence distribution for both Julius and our system. The histogram distribution graphs are shown in figure 4.2. We notice in figure 4.2 that the ASR's and our system's confidence distributions are completely different. For incorrect words, our system has asserted lot of zero confidences, which makes perfect sense since the words are completely wrong. Julius on the other hand wrongly ranks incorrect words with high confidence values. In order for us to present the distributions more clearly, we have removed in figure 4.3 the words with confidence values equal to 1 by the ASR and all the words with confidences equal to 0 by our system. Features of the distributions are now more visible in figure 4.3. Obviously, the type of distribution for the ASR is not preferred. In fact, there is lot of overlapping between the correct word confidence distribution and the incorrect one, which shows again the inability of Julius to decide and to rank appropriately the correct and the wrongly recognized words. However for our system, the distinction is clear and the distributions for correct and incorrect words are quite dissociated. But it is not optimal: in fact some overlap exists around the 0.2 confidence value. Ideally, the two distributions (for correct and incorrect) have to avoid any overlapping so that we don't have correctly accepted word confidences smaller than wrongly spotted word confidences and falsely mapped word confidences bigger than correctly accepted word confidences.

The distribution of the confidences for the correct words with respect to that of the

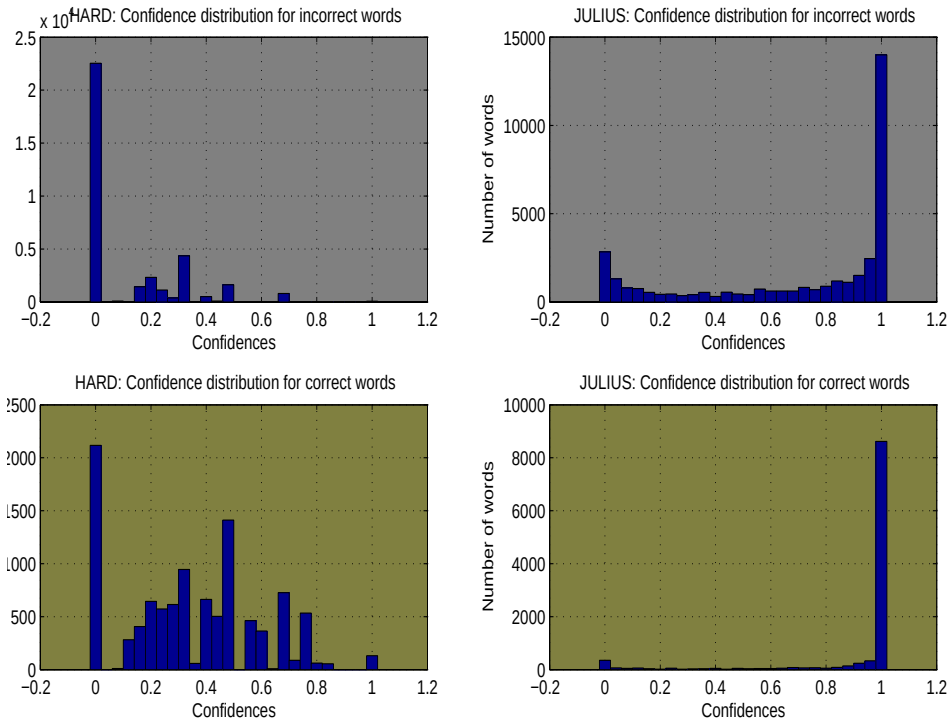

Figure 4.2: Confidence distributions for JULIUS and with our approach

incorrect ones, relative to the overall mean, is also important. Figure 4.4 shows the words that have confidences less (respectively higher) than the overall confidence value for both Julius and our system. For all the plots, the middle red point is the overall mean confidence value for a specific system and specific status (correct or incorrect). For correct words, ideally most of the words have to be located right of the overall mean. This is the case for both Julius and our approach. However, Julius behaves quite better than our approach. For the incorrect words, ideally most of the words have to be located left of the overall mean. For our system, this is so, but Julius behaves in complete opposition to this. In fact, for Julius, most of the incorrect words have their confidences above the overall confidence value. This again explains the fact that Julius cannot distinguish clearly and efficiently between the correct and falsely mapped words.

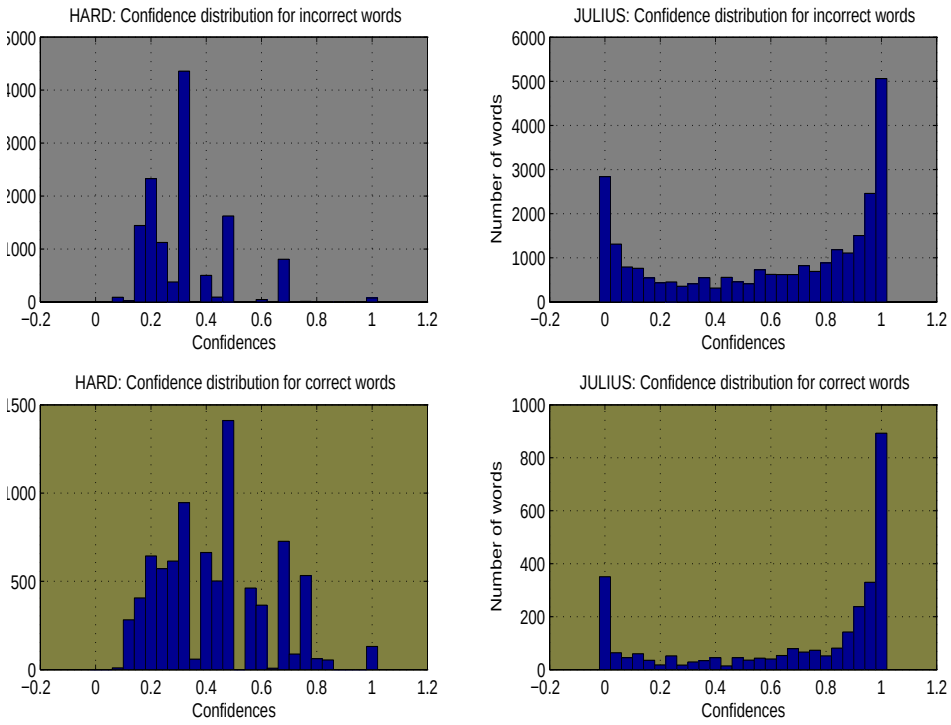

Figure 4.3: Confidence distributions for JULIUS and with our approach

4.3.3 Summary

The confidence measure we proposed in this thesis outperforms the Julius confidence metric especially in the ranking of incorrect words. In fact, we have proved that with our technique, we can distinguish easily between the falsely mapped words and the correctly spotted ones. However, our technique still needs improvement in terms of ranking of the correctly accepted keywords. But given the fact that our model has been trained with only 10 hours of data versus Julius which is trained with more than 100 hours of speech data, we can see that our results could only improve once we train our system with more speech data. Moreover and despite the fact that the Julius model is context-dependent phones (thousands of tri-phones), while our models are context-independent phones, which again gives Julius much advantage, we still achieved better results than Julius's way of confidence scoring.

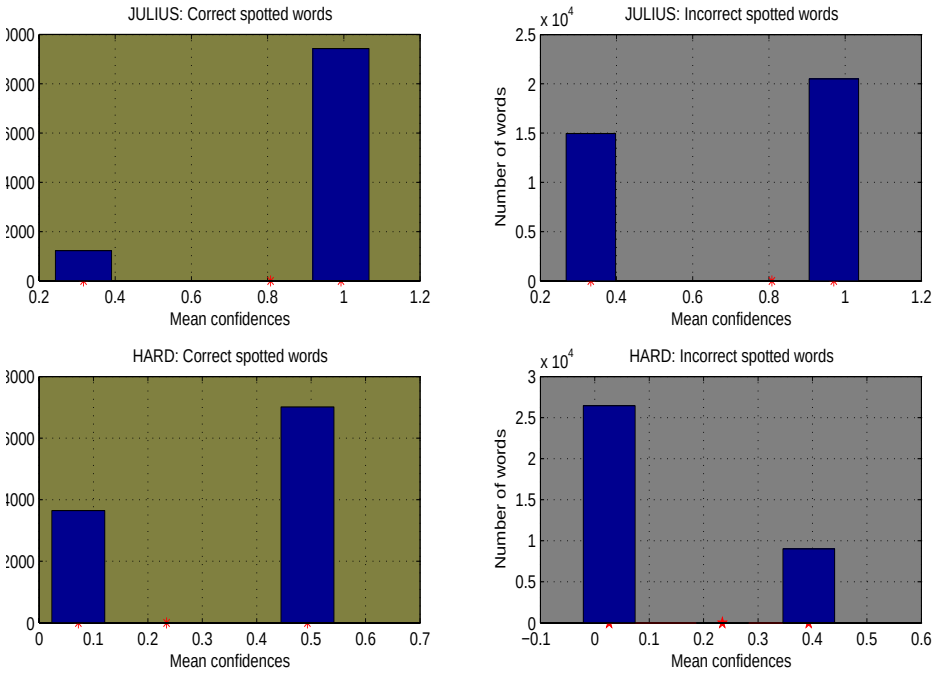

Figure 4.4: Correct and incorrect word confidences compared to the overall mean

Chapter 5

Conclusion And Future Work

The main goal of this thesis was to investigate a new approach to filler model-free keyword spotting system. The approach consisted of using a phone classifier and some confidence metrics, different from the existing speech recognizer ones, to get a new ranking for the ASR output. Ultimately, this scoring will be used to discard and filter out all the falsely mapped words returned by the ASR.

We presented in chapter 3 the adopted techniques and methods to implement our approach. A fuzzy c-mean based modification of the Gaussian mixture modeling was attempted. Results showed an improvement of approximately 2% when using FGMM for phoneme classification that led to a classification rate of 66%.

The analysis in chapter 4 of the scoring mechanism that we introduced proved that in terms of deviations from total mean, the ASR gives a 13% result compared to 50% with our proposed approach, showing a much larger differentiation between correct and incorrect spotted words. This shows that our approach is promising, and with further investigation and development it could lead to an efficient way to build a filler model-free keyword spotter. The obtained results are encouraging and we can see lot of potential in this approach toward a robust and reliable scoring of the ASR output.

We have already targeted a few areas worthy of further investigation to improve the performance of the proposed approach in this thesis. Areas of improvements in phoneme classification module include:

- We have to use much more data for training the FGMM/GMM models. In fact, in this thesis, we have only used the TIMIT database to create the mod-

els, whereas the Julius acoustic model was trained with at least 100 hours of speech, and with data coming from a Wall Street Journal database, Resource Management and other corpus. Using more training data will certainly improve the classification rate of our models.

- The phone models used in this thesis are context-independent whereas Julius is using a context-dependent models, with thousands of tri-phones. So using context-dependent models is another direction to investigate. In fact, this can boost the performance of the phone classifier. We could try using bi-phone or tri-phone context models. Then instead of having to train only models for the set of phones, we will train bi-phone models.

- In [19], a robust clustering approach to fuzzy Gaussian mixture modeling is proposed. GMMs and FGMMs both have a common disadvantage in the problem of sensitivity to outliers. This approach of discarding outliers is quite successful in improving the robustness of a variety of fuzzy clustering algorithms. We can use this technique to improve the performance of our FGMMs.

- Finally, the feature extraction techniques used for converting the acoustic events (speech) to feature vectors. We might need to use other features that will help in the classification of similar phones or boost the detection rate for certain classes of phones, like the formants for the vowels.

Areas of improvements for the confidence metric we have proposed can be summarized as follows:

- We need to improve the scoring mechanism. For example we might need to keep the N-Best hypothesis in the phoneme classification, and then check which one of the N-best matches the ASR output.

- We can cluster the phonemes in different classes based on the phone similarity aspect. And when deciding whether a FGMM/GMM phone hypothesis does or does not match the ASR-output phone, we should accept phones that belongs to the same family (class) instead of choosing only the perfect match.

- Another aspect we might need to consider is the output score of the FGMM/GMM models. A metric using both the likelihood of the models and the number of matches in one word might boost the performance. But this type of approach implies that our FGMM/GMM have high classification accuracy. So it may be that other techniques for phone modeling might be worth investigating to increase the phone classification rate.

We strongly believe that further research work in this direction could lead to substantial improvements on the results we have so far obtained during this investigation and could lead to potential commercial adoption.

Bibliography

[1] X. Huang, A. Acero, and H. Hon. *Spoken language processing: A guide to theory, algorithm and system development*. Prentice Hall, 2001.

[2] I. Bazzi and J. Glass. Modeling out-of-vocabulary words for robust speech recognition. In *Proceedings ICSLP*, Beijing,China, 2000.

[3] C. Yining, L. Jing, Z. Lin, L. Jia, and L. Runsheng. Keyword spotting based on mixed grammar model. In *International Symposium on Intelligent Multimedia, Video and Speech Processing*, pages 425–428, Hong Kong, 2001.

[4] M. Sigmund. Searching for phoneme boundaries in speech signal. Technical report, Institute of radio electronics,University of Technology, Purkynova, Czech Republic, 1995.

[5] A. Ahmadi. *Modular-based classifer for phoneme recogntion*. PhD thesis, University of Waterloo, 2005.

[6] J. Deller, J. Hansen, and J. Proakis. *Discrete- Time Processing of Speech Signals*. IEEE Press, 2000.

[7] J. Coleman. *Introducing speech and language processing*. Cambridge university press, 2005.

[8] T. Robinson and F. Fallside. A recurrent error propagation network speech recognition system. *Computer Speech and Language*, 5:259–274, July 1991.

[9] T. Robinson. An application of recurrent nets to phone probability estimation. *IEEE Transactions on Neural Networks*, 5(2):298–305, 1994.

[10] R. Chen and L. Jamieson. Experiments on the implementation of recurrent neural networks for speech phone recognition, purdue university, west lafayette, usa, 1998.

[11] P. Clarkson and P. Moreno. On the use of support vector machines for phonetic classification. *Acoustics, Speech and Signal Processing*, 2:585–588, 2000.

[12] K. K. Chin. *Support vector machines applied to speech pattern classification.* Master's thesis, Engineering Department, Cambridge University, 1999.

[13] H. Jiang. Confidence measures for speech recognition: A survey. *Speech Communication*, 45(4):455–470, April 2005.

[14] R. San-Segundo, B. Pellom, K. Hacioglu, and W. Ward. Confidence measures for spoken dialogue systems. *Proc of International Conference on Acoustics, Speech and Signal Processing*, 2001.

[15] S. Cox and R. Rose. Confidence measures for the switchboard database. *Proc. of International Conference on Acoustics, Speech and Signal Processing*, pages 511–514, 1996.

[16] B. Chigier. Rejection and keyword spotting algorithms for a directory assistance city name recognition application. *Proc. of International Conference on Acoustics, Speech and Signal Processing*, pages 93–96, 1992.

[17] R. Zhang and A.I. Rudnicky. Word level confidence annotation using combinations of features. *Proc. of European Conference on Speech Communication Technology*, 2001.

[18] A. Dempster, N. Laird, and D Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of Royal Statistical Society*, 39(1):1–38, 1977.

[19] D. Tran and M. Wagner. A robust clustering approach to fuzzy gaussian mixture models for speaker identification. In *1999 Third international conference on Knowledge-Based Intelligent Information Engineering Systems*, pages 337–340, Adelaide, Australia, August 1999.

[20] X. Huang, Y. Ariki, and M. Jack. *Hidden markov models for speech recognition.* Edinburgh university press, 1990.

[21] D. Tran, T. Pham, and X. Zhou. Cell phase identification using fuzzy gaussian mixture models. In *the 2005 International Symposium on Intelligent Signal Processing and Communications Systems*, pages 465–468, Hong Kong, December 2005.

[22] A. Bleau and L. Leon. Watershed-based segmentation and region merging. *Computer Vision and Image Understanding*, 77:317–370, 2000.

[23] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. Berkeley, University of California Press, 1967.

[24] M. Eucci. K-means clustering. http://www.elet.polimi.it/upload/matteucc/Clustering/tutorialhtml/kmeans.html.

[25] J. C. Dunn. A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. *Journal of Cybernetics*, 3:32–57, 1973.

[26] J. C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algoritms*. Plenum Press, New York, 1981.

[27] M. Eucci. C-means clustering. http://www.elet.polimi.it/upload/matteucc/Clustering/tutorialhtml/cmeans.html.

[28] T. Kawahara and A. Lee. Free software toolkit for japanese large vocabulary continuous speech recognition. *International Conference on Spoken Language Processing (ICSLP)*, 4:476–479, 2000.

[29] I. Lane. Multipurpose large vocabulary continuous speech recognition engine julius. Technical report, Kyoto University, Kyoto, Japan, March 2001.

[30] ATT Natural Voices. Text-to-speech engines, system developers guide: Server, server-lite, and desktop editions, 2004.

[31] S. Young and al. *The HTK Book*. Cambridge University Engineering Department, December 2006.

[32] Cambridge University Engineering Department. HTK speech recognition toolkit. http://htk.eng.cam.ac.uk/.

[33] J. Garafolo. Getting started with the DARPA TIMIT CD-ROM: An acoustic phonetic speech database, 1988.

[34] L. R. Rabiner and B. H. Juang. *Fundamentals of Speech Recognition*. Prentice Hall, New Jersey, USA, 1993.

[35] Y. Li. Singal processing for speech applications. Language Technology Institute, School of Computer Science,Carnegie Mellon University Pittsburgh, 2004.

[36] R. Kohavi and F. Provost. Confusion matrix. *Machine learning*, 30(2-3):271–274, February 1998.

[37] F. Karray and C. de Silva. *Soft computing and intelligent systems design: Theory, Tools and Applications*. Addison Wesley Publishing, 2004.

[38] M. Seltzer. Sphinx III signal processing front end specification. Technical report, CMU Speech Group, 1999.

Appendix A

Node Synonyms Of the Call Routing Tree

Each second level node in the call routing tree in figure 4.1 can be referenced by 4 synonyms. Table A shows the list of synonyms:

Node name	Synonyms
Installation	installation to install to start commencement
Reception problem	reception problem poor reception no reception weak signal
Service plan	service plan plan of service warranty repair plan
Installation problem	installation problem starting problem won't start set up problem
Malfunction	malfunction not working working incorrectly operating problem
Bill payment	bill payment bill payment pay bill money payment
Misplaced	misplaced lost stolen dropped
Cancelation	cancellation to cancel to cut to end
New client	new client subscriber customer member

Table A.1: 2-level node synonyms for the routing tree in figure 4.1

Appendix B

Templates For Natural Language Sentence Generation

The 6 distinct templates prepared to generate natural sentences with the nodes defined in figure 4.1, are presented in this appendix.

1. The sentences below can be used with the following terminal (ie. final routing) nodes of the routing tree **reception problem**, **installation problem**, and **malfunction**. Note that [cable tv] will be substituted by all the 6 different products in the tree.

    - There is a [reception problem] with my [cable TV]
    - Can you help me with [cable tv]'s [reception problem] now
    - I want to speak to someone regarding [reception problem] of my [cable tv]
    - Is there a way to solve the [reception problem] of my [cable tv]
    - I am sick and tired of my [cable tv]'s [reception problem]
    - My [cable tv]'s [reception problem] is absolutely terrible
    - Why don't you understand that I need help with [cable tv]'s [reception problem]
    - Please help fix my [cable tv]'s [reception problem] right now
    - Do you have anyone to discuss [reception problem] of [cable tv]
    - I can't wait until Friday for someone to fix my [cable tv]'s [reception problem]

- Do you understand that I have a serious [reception problem] with my [cable tv]
- Stop arguing with me and have someone fix [cable tv]'s [reception problem]
- I can't believe that you can't fix [reception problem] of my [cable tv]
- I have been waiting for twenty minutes to talk about my [cable tv]'s [reception problem]
- So, nobody is available to help me [cable tv]'s [reception problem], huh

2. The sentences below can be used with the following terminal (ie. final routing) nodes of the routing tree **misplaced**. Note that [mobile phone] will be substituted by all the 6 different products in the tree.

   - I have [misplaced] my [mobile phone] somewhere
   - Do you understand that I have [misplaced] my [mobile phone] or not
   - Didn't I just tell you that I [misplaced] my [mobile phone]
   - I have been trying to explain for ten minutes that I have [misplaced] my [mobile phone]
   - I can't believe that you don't understand that my [mobile phone] has been misplaced
   - Please help me as I have [misplaced] my [mobile phone]
   - I have been waiting to report my [misplaced] [mobile phone]
   - Sorry, I forgot to call yesterday to tell you that I [misplaced] my [mobile phone]
   - Is there someone who can help regarding my [misplaced] [mobile phone]
   - So, you are telling me that I can't report a [misplaced] [mobile phone]
   - All I want to report is that I [misplaced] my [mobile phone]
   - Just listen to me and find someone regarding [misplaced] [mobile phone]

3. The sentences below can be used with the following terminal (ie. final routing) nodes of the routing tree **bill payment**. Note that [cable tv] will be substituted by all the 6 different products in the tree.

   - Can you help me with [cable tv]'s [bill payment] now
   - I want to speak to someone regarding [bill payment] for my [cable tv]

- Is there a way to solve the [bill payment] problem of my [cable tv]
- I am sick and tired of your losing my [bill payment] for [cable tv] account
- Why don't you understand that I need help with [cable tv]'s [bill payment]
- Please report [cable tv]'s [bill payment] right now
- Do you have anyone to discuss [reception problem] of [cable tv] with me
- I can't wait until Friday for someone to process my [cable tv]'s [bill payment]
- Do you understand that I have a serious [bill payment] issue with my [cable tv]
- Stop arguing with me and just process [cable tv]'s [bill payment] for heaven's sake
- I can't believe that you can't fix my [cable tv] [bill payment] problem until next week
- I have been waiting for twenty minutes to talk about last month's [cable tv] [bill payment]
- So, nobody is available to help me [cable tv]'s [bill payment], huh
- I am telling you you forgot to report last month's [cable tv]'s [bill payment]
- How can I make a [bill payment] for my [cable tv]

4. The sentences below can be used with the following terminal (ie. final routing) nodes of the routing tree **cancelation** and **installation**. Note that [cable tv] will be substituted by all the 6 different products in the tree.

    - Can you help me with [cable tv]'s [cancelation] now
    - I want to speak to someone regarding [cancelation] of my [cable tv] service
    - Is there a way to solve the [cancelation] problem of my [cable tv]
    - I am sick and tired of your losing my [cancelation] notice for [cable tv] service
    - Why don't you understand that I need help with [cable tv]'s [cancelation]
    - Please report [cable tv]'s [cancelation] right now
    - Do you have anyone to discuss [cancelation] of [cable tv] service with me

- I can't wait until Friday for someone to process my [cable tv]'s [cancelation] notice
- Do you understand that I have requested [cancelation] of my [cable tv] service three times
- Stop arguing with me and just process [cable tv]'s [cancelation] order for heaven's sake
- I can't believe that you won't put in [cable tv] [cancelation] order until next month
- I have been waiting for twenty minutes to talk about last month's [cable tv] [cancelation] order
- So, nobody is available to help me [cable tv]'s [cancelation], huh
- I am telling you you forgot to report last month's [cable tv]'s [cancelation]
- How can I order a [cancelation] for my [cable tv] service

5. The sentences below can be used with the following terminal (ie. final routing) nodes of the routing tree **new client**. Note that [cable tv] will be substituted by all the 6 different products in the tree.

    - Can you help me become [new client] of [cable tv]
    - I want to speak to someone regarding becoming [new client] of [cable tv]
    - Don't you think I deserve as discount as a [new client] of [cable tv]
    - I am sick and tired of your losing my [new client] order for [cable tv]
    - Why don't you understand that I need help with become [cable tv]'s [new client]
    - Please report my status as [cable tv]'s [new client] right now
    - Do you have anyone to explain how I can be [new client] of [cable tv]
    - I can't wait until Friday for someone to process my [cable tv]'s [new client] request
    - Do you understand that I want to become [new client] of your [cable tv] service
    - Stop arguing with me and just process [cable tv]'s [new client] request for heaven's sake
    - I can't believe that you won't process my [cable tv] [new client] request until next week

- I have been waiting for twenty minutes to talk about last month's [cable tv] [new client] order
- So, nobody is available to help me become a [new client] of [cable tv], huh
- I am telling you you forgot to report that I am a [new client] of [cable tv]
- You cannot charge me extra fees as a [new client] of [cable tv]

6. The sentences below can be used with the following terminal (ie. final routing) nodes of the routing tree **service plan**. Note that [satellite tv] will be substituted by all the 6 different products in the tree.

    - Can you help me with [satellite tv]'s [service plan] now
    - I want to speak to someone regarding [service plan] for my [satellite tv]
    - Is there a way to change the [service plan] offered for [satellite tv] service
    - I am sick and tired of your third-rate [service plan] for [satellite tv] account
    - Why don't you understand that I need help with [satellite tv]'s [service plan]
    - Please explain [satellite tv]'s [service plan] right now
    - Do you have anyone to discuss [service plan] of [satellite tv] with me
    - I can't wait until Friday for someone to process my [satellite tv]'s [service plan] request
    - Do you understand that I have a serious [service plan] issue with my [satellite tv]
    - Stop arguing with me and just process [satellite tv]'s [service plan] changes for heaven's sake
    - I can't believe that you can't modify my [satellite tv] [service plan] until next week
    - I have been waiting for twenty minutes to talk about last month's [satellite tv] [service plan]
    - So, no body is available to help me [satellite tv]'s [service plan], huh
    - I am telling you you forgot to send me [satellite tv]'s [service plan] last month
    - How can I inquire about the [service plan] for my [satellite tv]

Appendix C

List Of Phonemes Based On TIMIT Database

Table C.1 shows the list of target phones. Note that we removed the silence and the short pause, only the phones that we have modeled using FGMM/GMM are in the table.

Phonemes	Word examples
aa	C**a**r
ae	**A**t
ah	**U**p
ao	D**o**g
aw	H**ow**
ay	**I**ce
b	**Bi**g
ch	Mar**ch**
d	**Di**g
dh	Fa**th**er
eh	P**e**t
er	T**ur**n
ey	D**ay**
f	**F**ork
g	An**g**le
hh	**H**elp
ih	H**i**t
iy	F**ee**l
jh	**J**oy
k	**C**ut
l	Sai**l**
m	M**a**d
n	**En**d
ng	Si**ng**
ow	G**o**
oy	T**oy**
p	**P**ut
r	**R**ed
s	**Si**t
sh	**Sh**e
t	**T**alk
th	**Th**in
uh	G**oo**d
uw	T**oo**l
v	O**v**er
w	**Wi**th
y	**Y**ard
z	La**z**y
zh	A**z**ure

Table C.1: Phone list based on TIMIT database

Appendix D

MFCC Features

The steps to construct MFCC features are as follows [38]:

1. Pre-Emphasis:
   The following FIR pre-emphasis filter is applied to the input waveform:

   $$y[n] = x[n] - \alpha x[n-1] \tag{D.1}$$

   α is provided by the user or set to the default value. If $\alpha = 0$, then this step is skipped. In addition, the appropriate sample of the input is stored as a history value for use during the next round of processing.

2. Windowing: The frame is multiplied by the following Hamming window:

   $$w[n] = 0.54 - 0.46\cos(\frac{2\pi n}{N-1}) \tag{D.2}$$

   N is the length of the frame.

3. Power Spectrum
   The power spectrum of the frame is computed by performing a DFT of length specified by the user, and then computing its magnitude squared.

   $$S[k] = (real(X[k]))^2 + (imag(X[k]))^2 \tag{D.3}$$

4. Mel Spectrum
   The mel spectrum of the power spectrum is computed by multiplying the

power spectrum by each of the of the triangular mel weighting filters and integrating the result.

$$\widetilde{S}[l] = \sum_{k=0}^{N/2} S[k] M_l[k] \quad l = 0, 1, ..., L-1; \tag{D.4}$$

N is the length of the DFT, and L is total number of triangular mel weighting filters.

5. Mel Cepstrum

   A DCT is applied to the natural logarithm of the mel spectrum to obtain the mel cepstrum:

$$c[n] = \sum_{i=0}^{L-1} ln(\widetilde{S}[i]) \cos(\frac{\pi n}{2L}(2i+1)) \quad n = 0, 1, ..., C-1; \tag{D.5}$$

   C is the number of cepstral coefficients.

Delta MFCC
Also, delta MFCC coefficients can be calculated using the following equation [35]:

$$\Delta c[n] = c[n+1] - c[n] \tag{D.6}$$

Delta delta MFCC
Moreover, to obtain delta-delta coefficients, following equation will be applied [35]:

$$\Delta\Delta c[n] = \Delta c[n+1] - \Delta c[n] \tag{D.7}$$