Abstract

With the enormous involvement of computer in everyday life, the human-computer interaction becomes more and more important. In particular the voice interaction with a computer which may likely replace standard keyboards in a near future. Such complete systems should recognize human speech, however, some applications do not need full speech recognition but only identification of the spoken language. This identification may require to be done relatively quickly, so, training of the system done offline to get small and efficient models for each language to insure fast and reliable recognition for the test utterance.

This project tackles the automatic spoken language recognition by a computer. From this perspective an implementation of this kind of system has been done with the objective to recognize spoken language in telephone speech. The process of Language recognition consists of 2 modules mainly: - feature extraction and feature matching. Feature extraction is the process in which we extract a small amount of data from the voice signal that can later be used to represent each Language. Feature matching involves identification of the unknown language speech by comparing the extracted features from speech input with the ones from a set of known languages model, Cepstral Coefficient Calculation and Mel frequency Cepstral Coefficients (MFCC)[see 3.1] are applied for feature extraction purpose. K-Mean, UBM-GMM (Universal Background Model based Gaussian Mixture Modeling), SVM (Support Victor Machine), Push-Back SVM to GMM, intersession variability compensation, Phonotactics and N-Gram, Gaussian Tokenization and Fused system algorithms are used for generating template and feature modeling, reduction, matching purpose.

An implementation was done and gave acceptable results (above 96% of accuracy) on the 1149 different utterance from the twelve languages of call-friend carpus data set [see 1.4]. The results of the implementation for this project and the different systems are shown and discussed in the last chapter.

المستخلص

وخلاصة البحث أنه بفضل وجود العولمة والتي كان القصد منها تعميم الفكر والثقافة على العالم، لينصهر كل شيء في بوتقة واحدة ,قد بدأ مصطلح "العالم قرية صغيرة" واقعا يتجسد أكثر فأكثر.

فمع التطور الهائل لتكنولوجيا المعلومات, أصبح التواصل بين الانسان والآلة أمرا يحتاج منا الوقوف عليه. ولينجح هذا التواصل , على الآلة أن تفهم خطاب الانسان وربما قد لا نحتاج الى تحديد الخطاب بل تحديد اللغة المحكية, وعلى هذا التحديد أن يتم بسرعة نسبيا بحيث تتمكن التطبيقات الأخرى أن تواكب سيرها بسرعة أيضا وعلى الرغم من أنك قد ترى هذه المهمة سلسة على إنسان ملم بتلك اللغات ويستطيع تحديد أي لغة وبسهولة, الا أنها تعتبر أمرا صعبا جدا لتطبيقه على آلة, حيث يمكن لتطبيق كهذا أن يستخدم في المطارات, المستشفيات,وحتى الفنادق.

وهذا ماقد حققناه في هذا المشروع بتطبيق عدد من الخوارزميات والأنظمة التي هدفت سويا لإنجاز هذا النظام والتحسين عليه ليصل إلى هذه النتيجة , علما أننا خضنا في أساليب وتجارب حديثة والتي تعتبر من خوارزميات الوقت المعاصر إلى أن حصلنا على نتائج مرضية يمكن القول أنها تصل الى ٩٦٪.

Table of Contents

Abstract I
List of Figures III
List of Tables IV
List of Flowchart V
List of Abbreviation VI
Declaration VII
Acknowledgment VIII

Chapter 1: Introduction 1
1.1. Introduction: 1
1.2. Problem statement and contribution 2
1.3. Possible applications 2
1.4. Data Set Description 4
1.5. Literature View 5
1.6. Outline of report 6

Chapter 2: Language Identification Systems 7
2.1 Overview 7
2.1.1 Acoustic features Based System 7
One acoustic feature dominates the field of LID: Mel-frequency cepstral coefficients (MFCCs) [see 3.1.2]. They have been implemented in many systems and have given satisfying results 7
2.1.2 Phonotactics-based systems 8
2.1.3 Fused systems 9
2.2. Decision Making 10
2.2.1. Likelihood ratio 11
2.2.2. Universal Background Model 12
2.2.3. Detection Error Tradeoff Curve 12

Chapter 3: Acoustic Front-End Processing 14
3.1.1 Overview 14
3.1.2 Mel Frequency Cepstral Coefficient (MFCC) 14
3.1.3 MFCC Steps at a Glance 14
3.1.4 MFCC Steps in Depth 15
3.1.5 Deltas and Delta-Deltas 19
3.2 Speech Enhancement Technique 21
3.2.1. Mean and Variance Normalization 21
3.2.2. RASTA filtering 21
3.2.3. Cepstral Mean Subtraction 22
3.2.4. Silence removal 23
3.3. Chapter Summary 25

Chapter 4: Acoustic Based System 26
4.1. Overview 26
4.2. Gaussian Mixture Models (GMM) 26
4.2.1. Theory Overview 26
4.2.2. Choosing Covariance Mixture Models 27
4.2.3. Estimation of GMM parameters 28
4.2.4 GMM-UBM 30
4.3 Support Vector Machine (SVM) 32
4.3.1 GMM-SVM 33
4.4 Push-Back From SVM to GMM 34
4.5 Intersession variability compensation 35
Chapter 5: GMM tokenization and Phonotactic Based system 37
5.1 Phone Tokenization 37
5.2 GMM TOKENIZATION 38
6.2 N-gram Scoring 40
Chapter 6: Experiments and Results 41
6.1. Experiments 41
6.1.1.GMM-UBM Experiment's and Result 42
6.1.2.GMM-SVM Experiment's and result 45
6.1.3.Push-Back Model Experiment's and result 46
6.1.4.Inter-Session Variability Compensation Experiment's and result 47
6.1.5.GMM Tokenization Model Experiment's and result 49
6.1.6.Fused System Experiment's and result 51
Chapter 7: Conclusion 52
Bibliography 53

List of Figures

Figure 1.1: World Language distribution……2
Figure 1.2: LID system application……3
Figure 1.3: Automatic speech recognition…...……3
Figure 2.1: Five levels of LID features……..……..7
Figure 2.2: Block diagram of the training part of a LID system…….……8
Figure 2.3: Phoneme Recognition followed by Language Modeling (PRLM)……...…8
Figure 2.4: Parallel Phoneme Recognition followed by Language Modeling (PPRLM)…...…9
Figure 2.5: Practical examples of Identification and Verification Systems……..……10
Figure 2.6: Example on DET curve and the threshold setting…….……..13
Figure 2.7: DET curve and the system efficiency……..13
Figure 3.1: The MFCC System…….……15
Figure 3.2: human auditory system……15
Figure 3.3: Frame output of truncated signal……16
Figure 3.4: Hamming Window……..……17
Figure 3.5: Example of a Mel-spaced frequency bank…….....18
Figure 3.6: Delta and Delta-Delta computation……19
Figure 3.7: Delta and Delta-Delta computation……20
Figure 3.8: Block diagram of the ARMA filter that realizes the RASTA filtering…….…..22
Figure 3.9: the discrimination between speech and silence……..……23
Figure 3.10: The plot distinguish between voiced and unvoiced by short- time energy…...…24
Figure 4.1: GMM model showing a feature space and corresponding Gaussian model...……26
Figure 4.2: (a) Spherical Model (b) Diagonal Model (C) Full Model……..……27
Figure 4.3: K-means clustering……..28
Figure 4.4: Block diagram representation of GMM-UBM training's techniques. (a) Data from populations are pooled together then the GMM model is trained. (b) A GMM is trained for each population then models are combined……...……30
Figure 4.5: Illustration for SVM……...……31
Figure 4.6: process behind the GMM-SVM Super-vectors……...……33
Figure 4.7: Pushing Model construction……...……34
Figure 5.1: Diagram for a single-language PRLM……37
Figure 5.2: Block diagram for GMM tokenization……37
Figure 5.3: Diagram for LID System based on GMM tokenization and language modeling...38
Figure 6.1: the recognition rate from the 256 component……..……42
Figure6.2: the recognition rate from the 512 component……..……42
Figure 6.3: Plots of miss and false alarm probabilities for GMM-UBM system…..……43
Figure 6.4: Plots of miss and false alarm probabilities for GMM-SVM system…..……46
Figure 6.5: DET curve of U-Matrix and Size Effect on fixed GMM……..….. …..……48
Figure 6.6: Plot defined the relation between U-Matrix size and efficiency……...……48

List of Tables

Table 1.1 - Target Language…….4

Table 6.1 - Feature selection experiment: error rates and correct percentage of twelve languages by four different features…….40

Table 6.2 - Best Enhancement method selection experiment: error rates and correct percentage of twelve languages by six different techniques……..….40

Table 6.3- K-mean Vs. Random..……..……...……41

Table 6.4: Result of a parameter selection experiment: Error rates of twelve languages with seven different numbers of components…….....….41

Table 6.5 - Result of SVM evaluation on different number of components: Error rates of twelve languages with seven different numbers of components….....44

Table 6.6 - Result of Push-Back evaluation on different number of components: Error rates of twelve languages with seven different numbers of components….....45

Table 6.7 - Result of Intersession Variability Compensation on different size of U-Matrix and Fixed GMM ….....46

Table 6.8 - Result of Intersession Variability Compensation evaluation on different number of components. ……...….....49

Table 6.9 - Result of GMM-Tokenization evaluation on different number of component.....49

Table 6.10 - Result of Fused System evaluation on different number of components ….....51

List of Flowchart

Flowchart 1: GMM-UBM…..……30

Flowchart 2: GMM-SVM……..33

Flowchart 3: Push-Back From SVM to GMM……..34

Flowchart 4: Intersession variability compensation…..…….35

List of Abbreviation

LID	Language Identification
LPCC	Linear Predictive Cepstrum Coding
MFCC	Mel-Frequency Cepstrum Coefficients
PCA	Principal Component Analysis
PLP	Perceptual Linear Prediction
LPCC	Linear Prediction Cepstrum Coefficients
DFT	Discrete Fourier Transform
GMM	Gaussian Mixture Model
SVM	Support Vector Machine
NIST	National Institute of Standards and Technology
EM	Exception Maximization
DET	Detection Error Tradeoff
DFT	Discrete Fourier Transform
FFT	Fast Fourier Transform
DCT	Discrete Cosine Transform
IVC	Intersession Variability compensations
MVN	mean-variance normalization
CMS	Cepstral Mean Subtraction
SDC	the shifted delta cepstral
PPRLM	parallel Phoneme Recognition followed by Language Modeling
UBM	Universal background model

Declaration

We, Mohammad Nasser Kabajah, Hussam Abu Mazen and Renad Amleh, do here by declare this project report as our original work and has never been submitted for any award of a degree in any institution of higher learning.

Signed: ..... Date: 5/6/2014

Signed: ..... Date: 5/6/2014

Signed: ..... Date: 5/6/2014

Acknowledgment

First of all we would like to express our deep and honest gratitude towards our advisor and guider Dr. Abualseoud Hanani who has always been a guiding force behind this project work. His highly influential personality has provided us constant encouragement to tackle any difficult task assigned. We are indebted to him for his invaluable advice and for propelling us further in every aspect of our academic life. His depths of knowledge, crystal clear concepts have made our academic journey a cakewalk. We consider it our good fortune to have got an opportunity to work with such a wonderful personality.

Next we are grateful to all faculty members and staff of the Department of Computer System Engineering. For their generous help in various ways for the completion of this project.

We would like to thank all our friends and especially our classmates for all the thought provoking discussions we had, which inspired us to think beyond the obvious.

We are especially indebted to our parents for their love, sacrifice, and support. They are our first teachers after we came to this world and have always been milestones to lead us a disciplined life.

Chapter 1: Introduction

1.1. Introduction:

Thanks to the globalization, the world becomes more and more a small village, and with the rapid growth of information technology, human-to-machine communication becomes more and more usual, in particular voice interaction with a machine. For a complete interaction the computer should recognize human speech, however, some applications do not need full speech recognition but only identification of the spoken language. Moreover, this identification may require to be done relatively quickly, so other applications can operate quickly.

Automatic spoken Language Identification (LID) is a example on human to machine interface, and even if it may be clear for a human ear, language identification is difficult for a computer. LID refers to the process of having a spoken language recognized by a computer [1]. LID systems could be used, for instance, in airport or hospitals to help international visitors by electronic guide or in call centers to find automatically an interlocutor who matches the language of the caller or for automatically classifying some speech data from a huge database. Moreover, in some applications, an automatic LID system should be very quick, for example in case of an emergency call, where each second could be vital.

The LID system is not fresh. Research has been done since the 1970s and LID systems have become more and more complex and complicated. The simplest systems use only acoustic features which branch to many types; then to increase the accuracy, phonotactics features like phoneme recognition are added. The most complicated systems use word level approaches and analyze the syntax of the language. The top performance of these kinds of systems is high [2]. However this top performance is obtained under some strict lab conditions and existing LID techniques need to be significantly improved for real applications. This project attempts to investigate several key issues related to applying LID in real applications, and to go deeper in main obstacles that prevent this system to work in real-time life.

In general, there are three main approaches to building LID systems: The first approach generally deals with only acoustic features and unlabeled data or labeled data. The language identification features are extracted then the model is trained. The typical model used is a Gaussian Mixture Models (GMM) [3] for each language to be identified because of GMM's high performance, easy computation and considered as comprehensive modeling, The Other model used is Support Victor Machine (SVM) for each language where SVM considered as discriminative modeling. The second approach is to use a phoneme recognizer of one language or several recognizers in parallel. Then, models for different languages are built and used to identify the phoneme sequences of a test sample of speech. The resulting sequences can then be used in decision-making. The Third approach is Gaussian Mixture Model Tokenization which considered as the link between the previous two systems so we use the Acoustic feature to get a sequence of numbers related to the highest Gaussian for each feature which give a indication about the grammar of each language from the acoustic extraction not phoneme extraction as the second approach.

1.2. Problem statement and contribution

In real-life its fact that humans can recognize a language if they know it accurately. However, this is still an open problem for a machine. Each language is just a set of rules and yet it could sound very different depending on, set of factors such that, the speaker or accent or the environmental noises such channel noise, white noise, variability in hardware used such microphones, cellphone, etc. and the variability in communication media such VOIP, radio, telephone-line etc., Not to forget the huge size of data set which take too much time in training process, which limited the number of experiments during the semester. Moreover, The Data itself wasn't easy reachable.

All of these problem motivate us to build a system which has the ability to deal with the limitation due to these factor, not to forget that the algorithm achieved in this project can be used not in speech recognition only but can extend to another field such image processing and any field which have a feature to make processing on it.

Many of algorithm was introduced, over than 100 experiments was achieved in 7 different systems, after all, we mentioned in the introduction that our goal is to reach the state of art in this field and we were too close to it but the limitation on the hardware and the time prevent us to continue, we hope that our works can help other people in future, The result of this project discussed in the last chapters.

1.3. Possible applications

The possible applications related to LID system are enormous, there are 7000 million people around the world and 64% of them speak 14 languages as shown in Figure 1.1, whereas , the world's known language it's around 3000 languages [4]. So, to communicate with the others, it is necessary to know which language we are dealing with firstly.

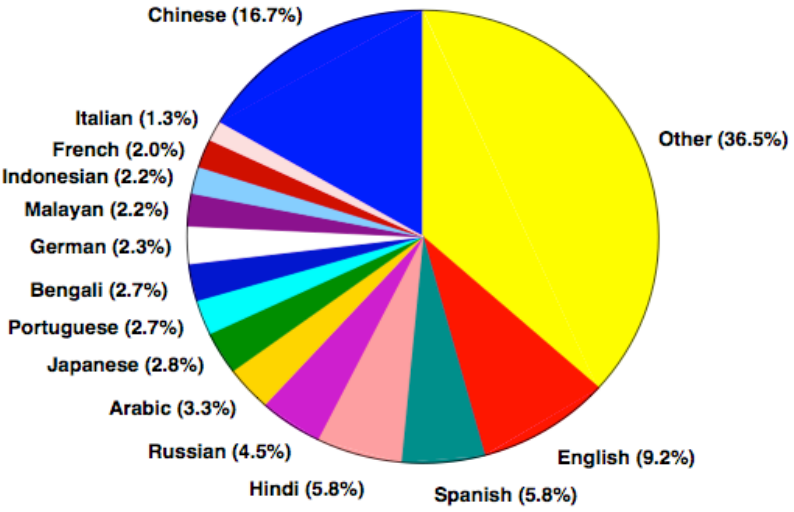


Figure 1.1: World Language distribution**.**

There are multiple field of applications of LID system nowadays. Probably the most prevalent field is security services for monitoring communications. Which help other systems such that keyword spotting (KWS) [4] see Figure 1.2.

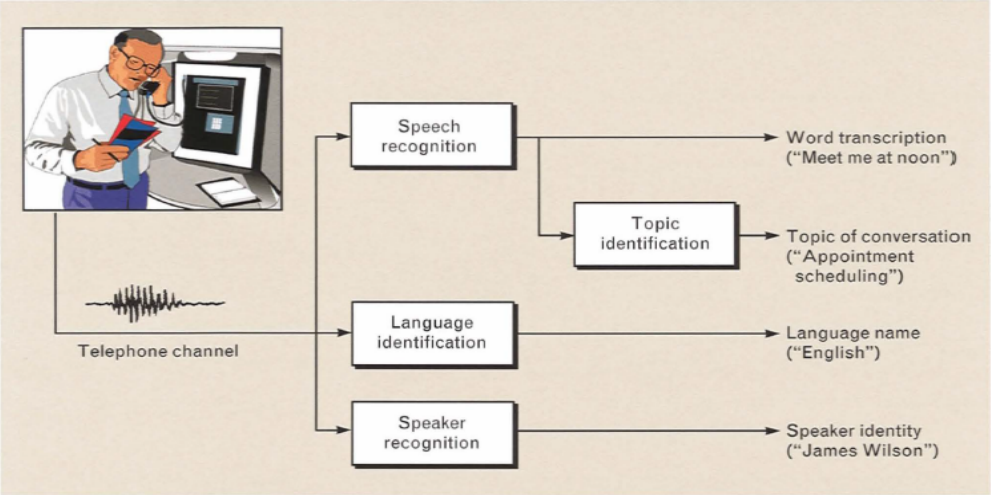


Figure 1.2: LID system application [4].

Alternatively, LID might be used in "call-centers" to route an incoming telephone call to a human switchboard operator fluent in the corresponding language see Figure 1.3. When a caller of a language line does not speak any English, a human operator must attempt to route the call to an appropriate interpreter. Much of the process is trial and error and requires connections to several human interpreters before the appropriate interpreter is found. The delay in finding appropriate human interpreter can up to minutes. Such delay can be destructive in emergency situation. An automatic language recognition system that can quickly determine the most likely languages of the incoming call might cut the delay by one or two orders of magnitude and ultimately save human lives.

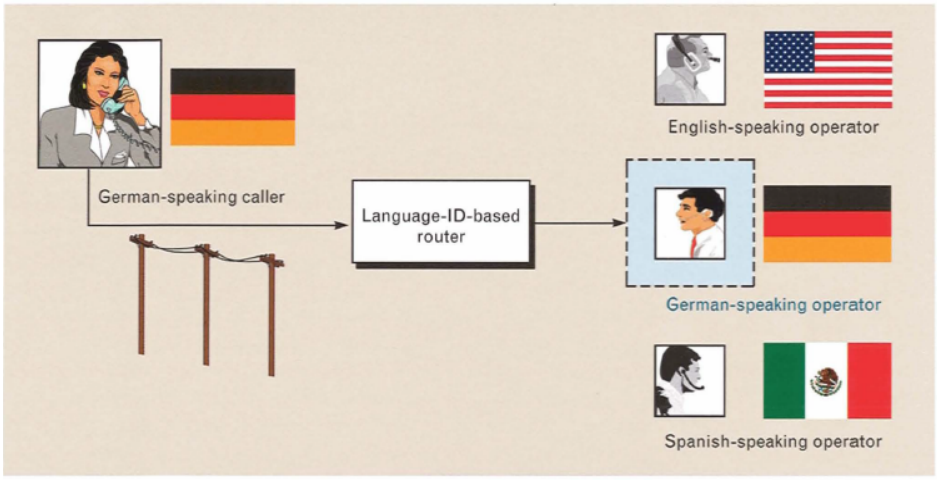


Figure 1.3: Automatic speech recognition [4].

There is lot of audiovisual data accessible through Internet and browsers similar to Google or Yahoo will need information about language of audio in which they are going to look for information. This system might use LID as a pre-processing of its own search.

In recent years there is increasing number of published papers on international conferences which means that the demand and requirements for automatic language recognition systems are steadily increasing.

1.4. Data Set Description

The data set and how to collect it are an important part of our project. We were had two choices to collect data. The first choice was to collect data by ourselves and this choice was very difficult since we can't collect different language from different countries. The second choice was to use data set already exist since it is easier and take less time than collect data. So in our project we used available data set.

The National Institute of Standards and Technology (NIST) [5] Language Recognition evaluation worked on telephone speech conversations between two friends from different language around the world in 1993-1994, 1996 and 2003 [5]. These conversations formalize the database where the database first is divided into two datasets. The first one is for training the system where is used to train language models, the second is the evaluation set where is used for test full system.

The CallFriend corpus dataset from NIST for year 1996 was used for training process which containing 12 different languages as shown in the table 1.1 below:

Table 1.1: Target Language [5].

English (General American)	**English** (Southern American)	**Arabic** (Conversational Egyptian)	**Farsi**	**French** (Canadian French)
Mandarin (from Mainland China)	**Mandarin** (from Taiwan)	**German**	**Hindi**	**Japanese**
Spanish (Caribbean)	**Spanish** (Highland)	**Korean**	**Tamil**	**Vietnamese**

Note that this set of languages includes two dialects for each language.

The speech signal to be processed will be represented as standard 8-bit 8 kHz mu-law digital telephone data. The conversations will be drawn primarily from LDC's CallFriend corpus. A subset of the Linguistic Data Consortium “CallFriend” corpus was used to evaluate the system. The CallFriend corpus consists of unscripted conversations between two people recorded at a sample frequency of 8 kHz in various languages captured over domestic

telephone lines. The particular subset of the CallFriend corpus used in these experiments was the same as that used for the 1996 NIST language recognition evaluation.

The test segments have three different durations: 3s, 10s and 30s. The data are taken from each of 20 conversations of the development data portion of CALLFRIEND database for each of the 12 target languages and dialects. Two test segments of each of the three test durations will be supplied for each side of each conversation. Thus there will be a total of 3,600 development test segments (for a total of about 15 hours of speech where 1147 for 30s, 1172 for 10s and 1174 for 3s).

The data set was used in our project consists of 12 languages (we regardless the dialects) and each of them approximately 30 minutes. Which mean 10 Hours for each language, and the evaluation set of the corpus consists of 1492 30-second utterances, distributed among the various languages of interest.

1.5. Literature View

There are many work related to the topic of Language Identification (LID) of spoken, after studying multiple researches done in the topic of Language Identification, we can notice that there is no significant change in the speech features. Most of the ideas came from how to classify the data. More details of the related work are discussed below.

Here some the state of art project shown to get a glance about used techniques and result obtained.

- In [6], researches used hybrid robust feature extraction technique for (LID) system. The Identification system is based on analyzing speech signal. The features extracted from speech by using Mel-frequency cepstral confidents (MFCC), Perceptual linear prediction coefficients (PLP) along with two hybrid features are used for language Identification. Two hybrid features, Bark Frequency Cepstral Coefficients (BFCC) and Revised Perceptual Linear Prediction Coefficients (RPLP) were obtained from combination of MFCC and PLP. The classification is done by using Vector Quantization (VQ) with Dynamic Time Warping (DTW) and Gaussian Mixture Model (GMM). A test has been carried out to decide the best features combinations. They found out that using hybrid feature extraction techniques compared to conventional feature extraction methods will give better identification rate. They got a recognition rate of 80.5% by applying MFCC.

- In [7], researchers used a set of six phone language-dependent recognizers based on HMM followed by language modeling of phone sequence for each language, they achieved accuracy of 91% on nine languages.
- In [8], researchers used big vocabulary speech recognition system based on phone level and word level; first they used bigram, but in the second stage trigram give better results. On four languages, the system achieved 84% accuracy with trigram.
- In [9], researchers used multigame for phonotactic modeling, they used English and German were used for testing, they achieved 73% accuracy on 10 second and 84% accuracy on 30 seconds, and using 3-gram they got 84% and 91% respectively.
- In [10], researchers used fused three systems – phone recognition Gaussian Mixture Modeling and SVM classification, they achieved more improvement in GMM Modeling, since they used gender dependent GMM and feature mapping technique.

1.6. Outline of report

The report is organized as follows: The first chapter presents the introduction to the LID system, most application and challenges of LID system. Then each chapter describes a step of an LID system: including speech enhancement techniques to reduce background noise or silence removal, feature extraction, language modeling methods and different decision making techniques according to the purpose of the system. Finally, the last chapter presents the implementation done for the purpose of this project and showed its results in real application.

Chapter 2: Language Identification Systems

In this chapters , we will introduced the main component used in LID system, the different type of it, starting from the Acoustic Based system to the phonotactic to the Fused system.

2.1 Overview

A LID system takes advantage of the characteristics of Human Formation [6], which distinguish one language from another. The level of difficulty of these characteristics is shown in figure 2.1. To keep it simple and to stay in systems with reasonable computation time, the LID systems presented below uses either acoustic or phonotactic features or systems between them or fused system, which contain all systems mentioned.

The next sections present the acoustic features, the phonotactics-based system and finally how integrated systems can be fused to improve their performance

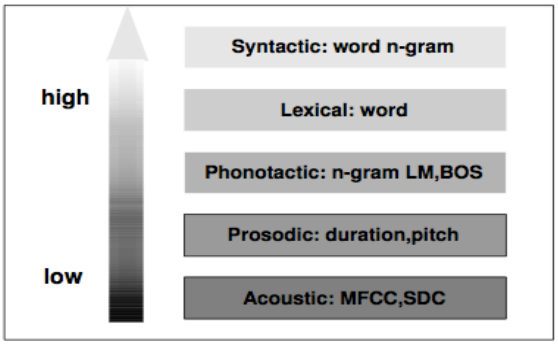


Figure 2.1: Five levels of LID features.

2.1.1 Acoustic features Based System

One acoustic feature dominates the field of LID: Mel-frequency cepstral coefficients (MFCCs) [see 3.1.2]. They have been implemented in many systems and have given satisfying results.
The training of each LID system generally follows the same steps, represented by figure 2.2. During pre-treatment, thanks to Hamming windows, the input signal is cut into several overlapping frames of 20 or 30ms. These frames are overlapped to overcome the edge's attenuation by the window. Next, features are computed from the frames and then used to train a model for each language. This results in one or more models that will be used during the recognition phase in order to determine the language of the test data.

Figure 2.2: Block diagram of the training part of a LID system.

However, a better set of features derived from the MFCCs has been discovered; they are called the shifted delta cepstral coefficients (SDC) [see 3.1.6]. To prove their efficiency, these new coefficients have been used with a GMM Gaussian Mixture Modeling (GMM), support vector machine (SVM) and Pushback technique as a classifier. These three classification methods are described in chapter 4 in detailed.

The definition and the computation algorithm of the MFCCs, SDCs is explained with details in chapter 3.

2.1.2 Phonotactics-based systems

Phonotactics-based systems produce a better in result than acoustic-based systems. However these kinds of systems need labeled data, which is very difficult to find or produce. Then such systems are less flexible due to adding a language to the system means that we need labeled data from this language.

Using phonotactics features necessitate that the phoneme recognizer must be trained before. Once again, the MFCCs are commonly used to detect the phoneme in a frame. Then, models for the desired languages have to be built by a generative model of phonetic n-gram sequence, for example. The phoneme recognizer does not need to be in the same language as the model; the same phoneme recognizer can be used for all models. Figure 2.3 shows an example of these LID systems.

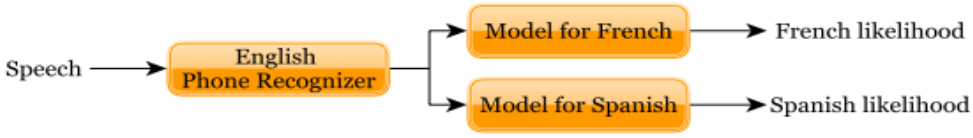


Figure 2.3: Phoneme Recognition followed by Language Modeling (PRLM).

However phoneme recognizers in different languages can be used in parallel, and each phoneme recognizer is followed by all the language models. A parallel Phoneme Recognition followed by Language Modeling (PPRLM) system provides state-of-the-art language recognition performance [2]. Figure 2.4 shows an example of PPRLM system. In this case the scores from the different models should be fused before making a decision, because it produces several scores for the same language. The fusion techniques said that If we used two or more language recognizers are complementary, then combining them could increase the language recognition rate. That is why researchers have developed language recognizer fusion. Fusion is also useful to combine the scores of a PPRLM system [7].

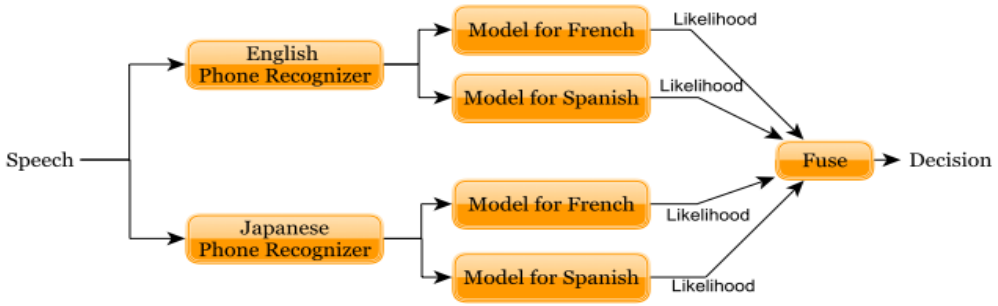


Figure 2.4: Parallel Phoneme Recognition followed by Language Modeling (PPRLM).

2.1.3 Fused systems

If two or more language recognizers are complementary, then combining them could increase the language recognition rate. That is why researchers have developed language recognizer fusion. Fusion is also useful to combine the scores of a PPRLM system [7]. This section presents two fused systems.

2.1.3.1 Product-rule fusion

Given a speech utterance, a score vector is computed for each recognizer, this vector contains the scores of all the target languages. To get the final score for each language, one needs to fuse the scores of each recognizer. For a system with K recognizers and L languages, the score for an input utterance X to a language l is computed from the likelihoods $s_{k,l}$ as follows:

$$Y(x \mid l) = \prod_{K=1}^{K} \frac{KS_{k,l}}{\sum_{i=1}^{L} S_{k,i}} \ldots\ldots\ldots (2.1)$$

Normalization guarantees that the output from the different phoneme recognizers is in a common range, and the product produces the final score among all the recognizers. However it assumes that the individual PRLM systems are independent. Moreover, a quasi-null score from a recognizer k may have a big impact on the final score. Despite that, it appears that it is an effective way to fuse scores [8].

2.1.3.2. Gaussian-based fusion

A Gaussian-based fusion has been proposed recently [8]. The principle is as follows: each recognizer produces a score (or it could be a vector of scores) for each language to be identified. Then Linear Discriminant Analysis is applied to the score vector to project it into a smaller dimensional space where the languages are well discriminated. Finally, the projected score vector is processed by a Gaussian classifier for each language which produces a final score for this language.

2.2. Decision Making

Its important before to introduce the decision maker technique to understand the Language Verification and the language identification, Figure 2.5 below illustrate the basic differences between Languages identification and verification systems.

1. **Language identification:** It is the process of recognize which language nearest to the test utterance by assign score from comparing with each language models then the highest score taken.

2. **Language verification:** It is the process a threshold is estimated for each language, for each file if the value exceed that threshold we can consider it from this model else its rejected so here we really didn't recognize we just accept or reject.

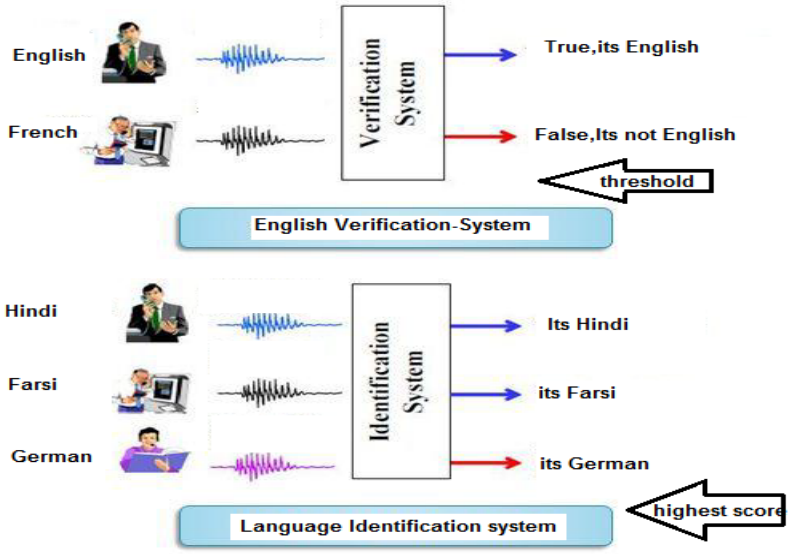


Figure 2.5: Practical examples of Identification and Verification Systems.

Once the different scores of a speech utterance for all the target languages have been computed, the system needs to make a decision as indeed the aim is to Recognize in which language a speech utterance is spoken. In the following, some of decision maker techniques used in LID are presented.

2.2.1. Likelihood ratio

One could simply take the maximum scores among all languages as the guessed Language. This may be good enough for a closed set identification, which means we know that the language of the test utterance belongs to one of the languages used for training. However, it is not enough for a detection task where a test utterance may not belong to the target languages, one could use a threshold to take a decision. But this threshold has to be compared with meaningful scores: the score or likelihood for each language $y(X_{s1})$ could, for example, is normalized according to the scores of the other languages. This is called a likelihood ratio $r(X|l)$ and it is defined as:

$$r(X|l) = \frac{y(X|l)}{\frac{1}{L-1}\sum_{i\neq l}^{L} y(X|i)} \quad \ldots\ldots\ldots\ldots (2.2)$$

The likelihood ratio is consistent and can be compared with a fixed threshold regardless the testing utterance.

2.2.2. Universal Background Model

The language detection task is to determine the language of a speech utterance X. The assumption that there is only one spoken language in the speech utterance is generally made, and it is made in this report. The previous detection method simply computes the likelihood of an utterance belonging to a particular language. But it could be more interesting and accurate to build a test between two hypotheses:

$H_{0,l}$: Utterance belongs to the language l.

$H_{1,l}$, : Utterance does not belong to the language l.

Then the test is organized as follows, given a decision threshold n for accepting $H0, l$:

$$\frac{P(\mathbf{X}|H_{0,l})}{P(\mathbf{X}|H_{1,l})} \begin{cases} \geq \eta & \text{accept } H_{0,l} \text{ , reject } H_{1,l} \\ < \eta & \text{reject } H_{0,l} \text{ , accept } H_{1,l} \end{cases} \quad \ldots\ldots\ldots (2.3)$$

Thus, the computation of the log likelihoods (or scores) y(X|l) for each language modellized by a model C_l is simply:

$$y(X|l) = log\, p(X|C_l) - log\, p(X|\bar{C}_l) \ldots\ldots\ldots\ldots (2.4)$$

$p(X|C_l)$ is easily defined because one just needs a method to model the language l and represent $H_{0,l.}$ This can be done by classification methods described in Chapter 5. However $p(X|\bar{C}_l)$ is less well-defined, because $\overline{C_l}$ should theoretically represent the complement of the set defined by the class C_l. It is obviously impossible to completely cover all this set.

2.2.3. Detection Error Tradeoff Curve

In the previous sections, the decision making was to the Identification system purpose so we select the candidate model with largest probability measure, whereas, In Language verification systems however, the decision as to whether to accept the claim of identity is more complicated since this is an open set problem.

The solution to the verification problem is to set a threshold distance and accept the claim if the distance to the claimed model is below the threshold. If this is done, changing the threshold can vary the security of the system. In a secure system, the threshold is set very low which should result in fewer false acceptance (FA) errors and more false rejections (FR). In a less secure but more convenient system, the threshold is set higher resulting in more FA errors and fewer FR errors. This is illustrated in the DET curve figure 2.6 which plots the false accept rate vs. the false reject rate for different values of the threshold. A point on the error curve where the two error rates are equal is called the Equal Error Rate (EER) and is often quoted as an overall measure of a system's performance.

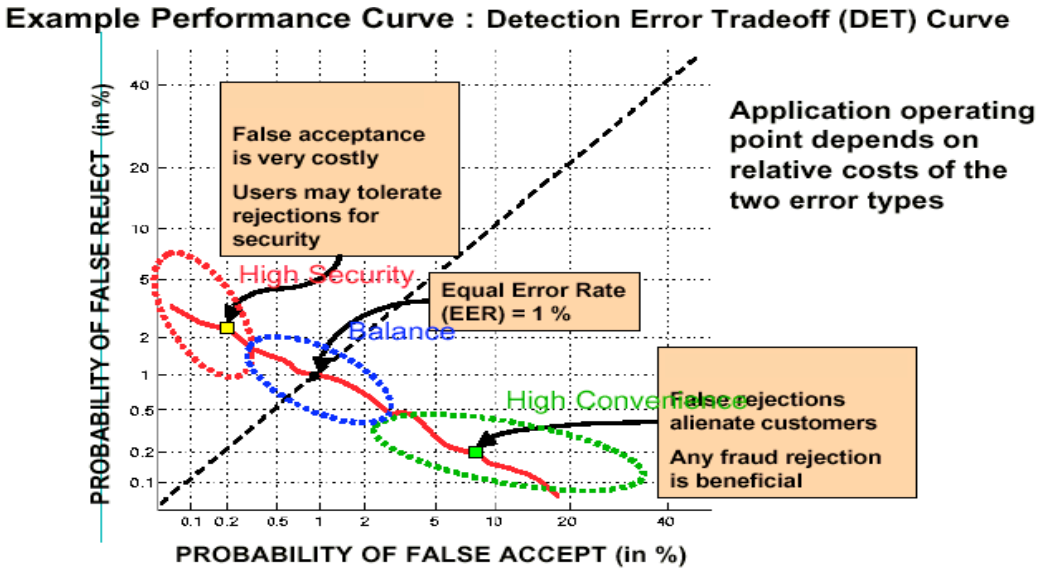


Figure 2.6: Example on DET curve and the threshold setting [9].

There is relation between DET curve and the robustness of the system which says in each time the DET curve become closer to the origin that means that the system becomes more accurate see figure 2.7.

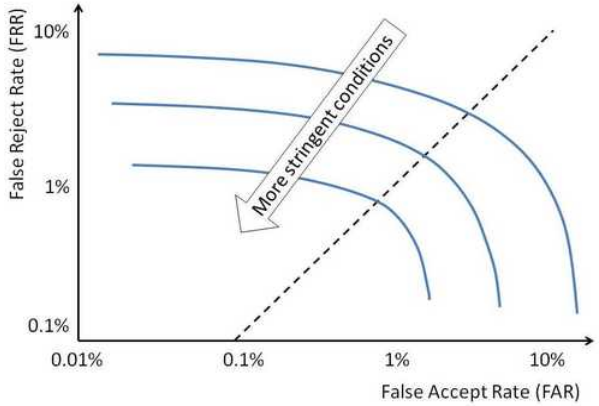


Figure 2.7: DET curve and the system efficiency [9].

Chapter 3: Acoustic Front-End Processing

3.1. Feature Extraction

3.1.1 Overview

Feature Extraction (or Front End processing) is the first stage of any speech processing system. The goal of it is to convert the speech signal to a suitable representation for application such as speech recognition, speaker identification, language and accent recognition, etc. It is also called feature extraction from the point of view of general pattern [10].

The acoustic feature extraction is a critical part of a speech recognizer, especially when the application is used in a noisy environment. To develop a good feature set, it is necessary to understand the different feature extraction techniques developed so far.

In this section we describe feature extraction methods and techniques that we are intending to use in our project and show each step of this process.

MFCC technique was chosen to setup a baseline of our project because of its popularity and success in similar systems, the steps of the MFCC shown in figure 3.1.

3.1.2 Mel Frequency Cepstral Coefficient (MFCC)

MFCC it is a feature extraction method that dominantly used in speech recognition. It was suggested for the first time for speech recognition, and its Mel-warped frequency scale in order to mimic how human ears process sound. Using Mel frequency scaling approximates the frequency response of human-auditory system, since the recent studies have shown that human auditory system does not follow a linear scale [11].

3.1.3 MFCC Steps at a Glance

We will give here a scintillation to the implementation steps, then travil to the depth why we do the things we do. Towards the end we will go into a more description of how to calculate

MFCCs.

1. Frame the signal to get short frames.
2. For each frame calculate the Fast Forier Transform(FFT)[12] to estimate the power spectrum.
3. Apply the mel filterbank to the power spectra, then sum the energy in each filter.
4. Take the logarithm of all filterbank energies.
5. Take the DCT of the log filterbank energies.
6. Keep DCT coefficients 1-19, discard the rest , which mean 19 coefficients.
7. There are a few more things commonly done, sometimes the frame energy is appended to each feature vector. Delta [see 3.1.5] and Delta-Delta[see 3.1.5] and Shifted-Delta Capestral(SDC) [3.1.6] features are usually also appended. Liftering is also commonly applied to the final features.

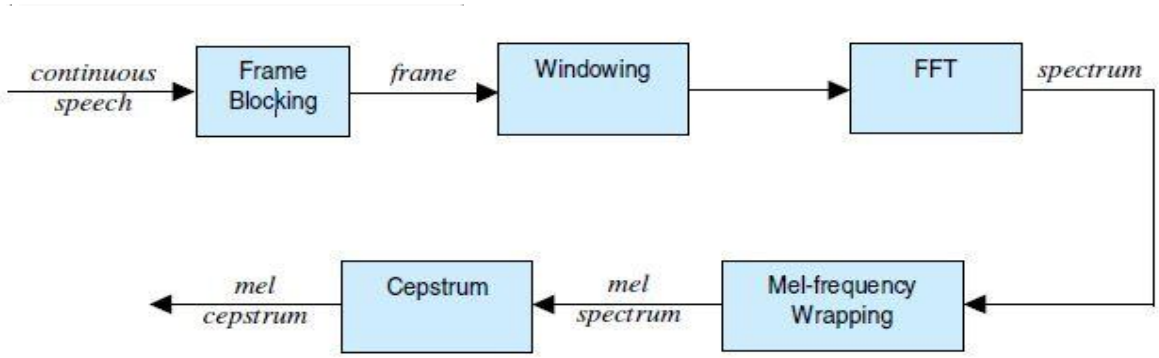


Figure 3.1: The MFCC System [12].

3.1.4 MFCC Steps in Depth

We represent the system of speech analysis like human peripheral auditory system as shown in the figure 3.2. In human auditory processing, sound enters the ear canal and then converted into mechanical motion by the eardrum. This sets the cochlea into motion, including the basilar membrane, which has mechanical resonant frequencies that vary systematically along its length: high-frequency components induce motion of the basilar membrane near its input end, and low-frequency components induce motion at the opposite end [12].

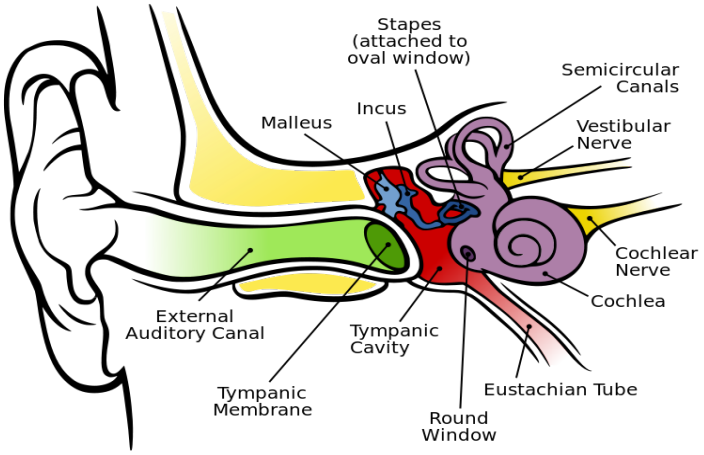


Figure 3.2: human auditory system [12].

MFCC speech analysis passing through many steps, frame blocking, windowing, Fast Fourier Transform, and discrete cosine transform (DCT).

3.1.4.1 FRAME BLOCKING

In this step the continuous speech signal is divided into frames of N samples, with adjacent frames being separated by M samples with the value M less than that of N. The first frame consists of the first N samples. The second frame begins from M samples after the first frame, and overlaps it by N - M samples and so on. This process continues until all the speech is accounted for using one or more frames [12]. We have chosen the values of M and N to be N = 256 which related to 20ms and M = 128 respectively. Figure 3.3 below gives the frame output of the truncated signal.

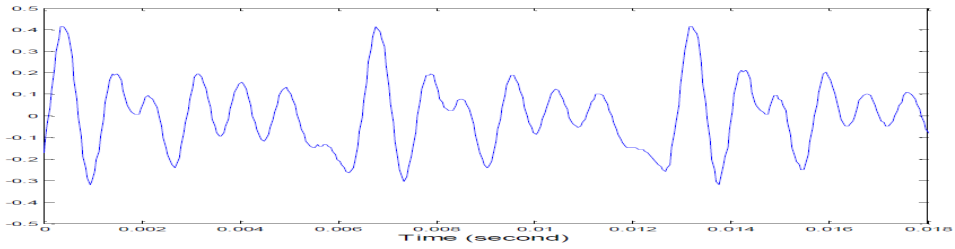


Figure 3.3: Frame output of truncated signal**.**

The value of N is chosen to be 256 because the speech signal is assumed to be periodic over the period. Also the frame of length 256 being a power of 2 can be used for using a fast implementation of Discrete Fourier Transform (DFT) called the FFT (Fast Fourier Transform).

3.1.4.2 WINDOWING

The next step is to window each individual frame to minimize the signal discontinuities at the beginning and end of each frame. The concept applied here is to minimize the spectral distortion by using the window to taper the signal to zero at the beginning and end of each frame. If we define the window as, where *N* is the frame length, then the result of windowing is the signal

$$\mathrm{Y(n)} = \mathrm{X(n)} * \mathrm{W(n)}, 0 \leq n \leq N-1 \ldots\ldots\ldots\ldots \text{(3.1)}$$

We have used the Hamming window in our project, which has the form:

$$\boldsymbol{W(n) = 0.54 - 0.46cos\left(\frac{2\pi n}{N-1}\right), 0 \leq n \leq N-1} \ldots\ldots\ldots \text{(3.2)}$$

Figure 3.4 below is the figure of a Hamming window.

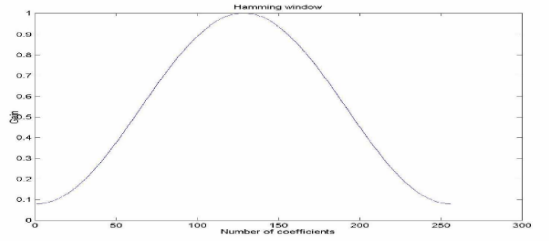


Figure 3.4: Hamming Window.

3.1.4.3 SHORT TERM FAST FOURIER TRANSFORM

The next step is the application of Fast Fourier Transform (FFT), which converts each frame of N samples from the time domain into the frequency domain. The FFT is a fast algorithm to implement the Discrete Fourier Transform (DFT) is defined on the set of N samples x_n, as follows: -

$$X_k = \sum_{n=0}^{N-1} x_n e^{-j2\pi kn/N}, \qquad k = 0,1,2,\ldots,N-1 \qquad \ldots\ldots\ldots \text{(3.3)}$$

In general X_k's are complex numbers and we consider only their absolute values. The result after this step is often referred to spectrum [13].

3.1.4.4 CEPSTRAL COEFFICIENTS USING DCT

Pre-processing the signal reduces the computational complexity while operating on the speech signal. We reduce the number of samples of operation. Instead of working on a large set of samples we restrict our operation to a frame of sufficiently reduced length. After conditioning the speech signal i.e. after pre-procession the next step is to extract the features of the training signal. We have made use of 2 methods for the same. The first method is calculating the Cepstral Coefficients of the signal using DCT (Discrete Cosine Transform). The Cepstral Coefficients are calculated using the following formula: -

$$Cep = DCT(\log|FFT(y)_{windowed}|) \ldots\ldots\ldots\ldots (3.4)$$

Then the 1st 19-cepstral coefficients are taken.

3.1.4.5 MEL-FREQUENCY WRAPPING

The Mel-frequency scale provides linear frequency spacing below 1 KHz and a logarithmic spacing above 1 KHz to deal with Psychophysical studies that revealed that human perception of frequency content of sounds for speech signals doesn't follow a linear scale .

The Mel Frequency Scale is given by: -

$$F_{mel} = \frac{1000}{\log 2} * \log 1 + \frac{f}{1000} \ldots\ldots\ldots (3.5)$$

One approach towards simulating the subjective spectrum is to use a filter-bank, which is spaced uniformly on the Mel-scale. The filter bank has a triangular band pass frequency response. The spacing and the bandwidth are determined by a constant Mel frequency interval. We choose K, the number of Mel spectrum coefficients to be 24. This filter bank being applied in the frequency domain simply amounts to applying the triangle-shape windows to the spectrum. A useful way to think about this filter bank is to view each filter as a histogram bin (where bins have overlap) in the frequency domain. Figure 3.5 below gives an example of a Mel-spaced frequency bank.

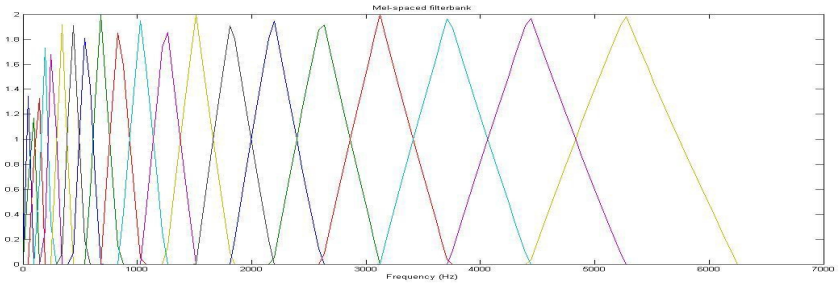


Figure 3.5: Example of a Mel-spaced frequency bank.

3.1.4.6 CEPSTRUM

In this final step, we convert the log Mel spectrum to time domain. The result is called the MFCC's. This representation of the speech spectrum provides a good approximation of the spectral properties of the signal for the given frame analysis. The Mel spectrum coefficients being real numbers are then converted to time domain using Discrete Cosine Transform (DCT). If we denote the Mel power spectrum coefficients that are the result of the last step as S_k , k=1,2,...,K, we can calculate the MFCC's C_n as

$$c(n) = \sqrt{\frac{2}{M}} \sum_{m=1}^{M} g(m) \cos\left(\frac{\pi n}{M}(m - 0.5)\right) \quad \text{..... (3.6)}$$

We exclude the first component from the DCT since it represents the mean value of the input signal, which carries little speaker specific information [13].

By applying the procedure described above, a set of mel-frequency cepstrum coefficients (MFCC) is computed for each speech frame. The set of coefficients is called an acoustic vector. Thus each input speech utterance is transformed into a sequence of acoustic vectors.

3.1.5 Deltas and Delta-Deltas

It's also known as differential and acceleration coefficients. The MFCC feature vector describes only the power spectral envelope of a single frame, but it seems like speech would also have information in the dynamics i.e. what are the trajectories of the MFCC coefficients over time. It turns out that calculating the MFCC trajectories and appending them to the original feature vector increases LID performance by quite a bit (if we have 19 MFCC coefficients, we would also get 19 delta coefficients, which would combine to give a feature vector of length 38) and to calculate the Delta to the delta , we would also get 19 delta-delta coefficients, then to append to the 38 to get vector of length 57.

To calculate the delta coefficients, the following formula is used:

$$d_t = \frac{\sum_{n=1}^{N} n(C_{t+n} - C_{t-n})}{2\sum_{n=1}^{N} n^2} \text{..... (3.7)}$$

Where d_t is a delta coefficient, from frame t computed in terms of the static coefficients C_{t+n} to C_{t-n}. A typical value for N is 2. Delta-Delta (Acceleration) coefficients are calculated in the same way, but they are calculated from the deltas, not the static coefficients. See figure 3.6 which show the Delta and Delta-Delta computation.

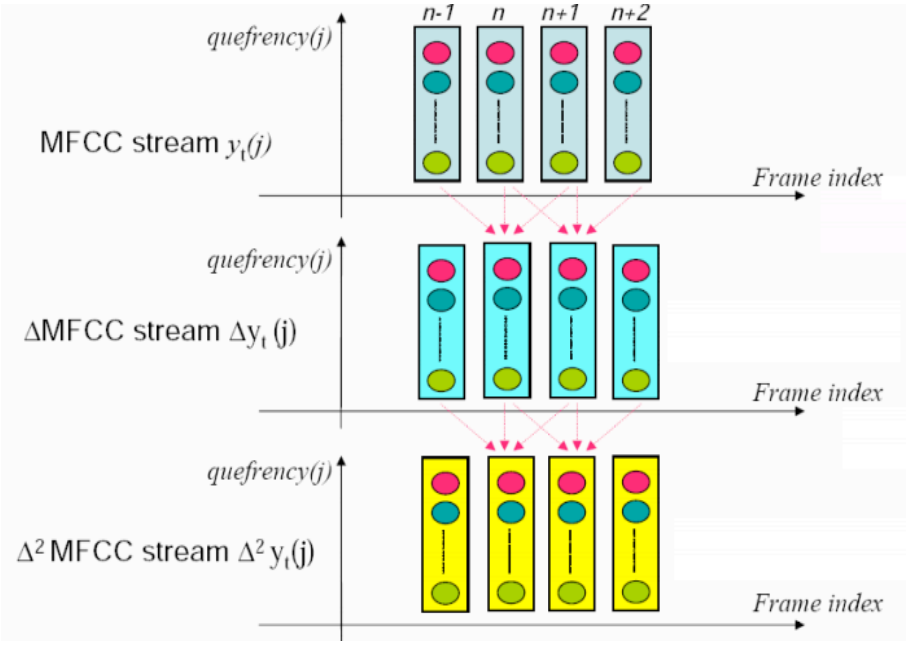


Figure 3.6: Delta and Delta-Delta computation.

3.1.6 The SDCs

MFCCs are usually augmented with their Delta (velocity) and double Delta (acceleration). A recently developed technique called Shifted Delta Cepstra (SDC) was successfully used in the language recognition area. SDC coefficients are obtained by doing concatenation between all delta cepstra computed across multiple frames of speech [14] as shown in figure 3.7.

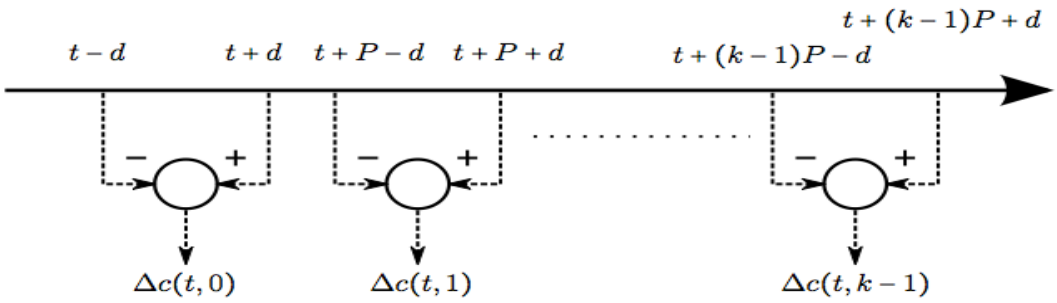


Figure 3.7: Delta and Delta-Delta computation.

SDC features are specified by a set of four parameters (N, d, P, k), where N is the number of cepstral coefficient computed at each frame, d is the time advance and delay for the delta computation, k is the number of blocks whose delta coefficient are concatenated to form the

final feature vector, and P represent the time shift between consecutive blocks [14]. So SDC coefficient for a cepstral frame i at time t, are computed as equation (eq) below:

$$\Delta^2 C_t(n) = \Delta C_{t-D}(n) -= \Delta C_{t+D}(n) \ldots\ldots\ldots\ldots \text{(3.8)}$$

3.2 Speech Enhancement Technique

Presently, with the massively increasing use of mobile phones with embedded microphones, an LID system must be prepared to deal with all kinds of noises, which reduce its performance. Indeed, white noise from a telephone channel, street noise or music may merge with speech and disturb the information contained in the speech sample. Noise might have to be removed by pre-processing the signal in order to gain good performance for a recognition task. This chapter describes some speech enhancement techniques.

3.2.1. Mean and Variance Normalization

The mean-variance normalization technique (MVN) is commonly used to increase the robustness of speech recognition features and to remove the channel effect. In MVN the mean and the variance of the cepstral coefficients of clean speech are assumed to be invariant. Therefore, removing mean and variance is assumed to reduce only irrelevant information, no matter what that information may be [15].

Equation (eq) describes how to calculate mean and equation (eq) describes how to calculate variance.

$$\mu = \frac{1}{n}\sum_{i=1}^{n} p_i x_i \ldots\ldots\ldots\ldots \text{(3.9)}$$

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n} P_i\,(x_{i-\mu})^2 \ldots\ldots\ldots\ldots \text{(3.10)}$$

After calculating these two values for each feature vector we subtract the mean and divide by the variance to bring all the data to the normal distribution. In this way we removed the channel effect from the entire speech.

3.2.2. RASTA filtering

Relative Spectra (RASTA) filtering is a speech enhancement technique used to reduce the channel noise from a speech sample [16]. RASTA filtering comes from the fact that the human vocal tract can't move too fast or too slow. The idea of RASTA filtering is to suppress slow variation thanks to a band filter in the frequency domain. So, it leads to removing the constant

magnitudes, during short-term spectrum, in each spectral component.

Research showed that the transfer function of this kind of filter is:

$$H(z) = 0.1z^4.\frac{2+z^{-1}-z^{-3}-z^{-4}}{1-0.98z^{-1}} \text{..... (3.11)}$$

The RASTA technique is performed by applying the autoregressive moving-average (ARMA) filter [refrence] defined by this equation to each component in feature vectors. Figure 3.8 shows the block diagram of ARMA filter used to perform the RASTA technique.

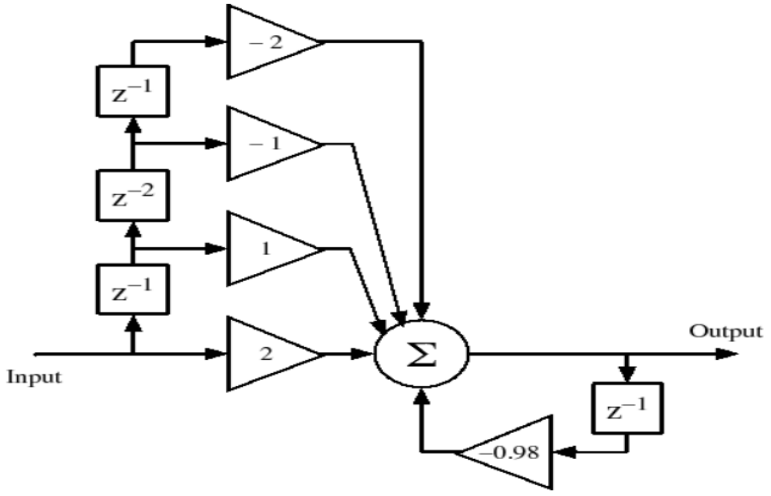


Figure 3.8: Block diagram of the ARMA filter that realizes the RASTA filtering.

3.2.3. Cepstral Mean Subtraction

Suppose we extract feature vectors for T frames where t represents a discrete time index of a frame so that t = 0 . . . T–1. Let c_t represent the t-th frame's MFCC feature vector,

$$\boldsymbol{c_t} = \begin{bmatrix} \boldsymbol{c_{t1}} \\ \vdots \\ \boldsymbol{c_{ti}} \end{bmatrix} \text{..... (3.12)}$$

Once a collection of MFCC feature vectors throughout all frames is calculated, it can have a mean vector μ from the following equation:

$$\boldsymbol{\mu} = \frac{\mathbf{1}}{\boldsymbol{n}} \sum_{\boldsymbol{t}=\mathbf{1}}^{\boldsymbol{t}-\mathbf{1}} \boldsymbol{c_t} \ldots\ldots\ldots \text{(3.13)}$$

Cepstral mean subtraction (CMS) is the operation of subtracting the mean vector from each MFCC feature vector, i.e.

$$\widetilde{\boldsymbol{c_t}} = \boldsymbol{c_t} - \boldsymbol{\mu}, \boldsymbol{t} = \mathbf{0} \ldots \boldsymbol{T} - \mathbf{1} \ldots\ldots\ldots \text{(3.14)}$$

CMS is often used in speaker verification systems for the removal of slowly varying convolutive noise due to communication channel [16]. Hence, CMS is effective when the communication channel is approximately time invariant.

3.2.4. Silence removal

Speech signals usually contain many areas of silence. Therefore, in speech analysis it is needed to first apply a silence removal method, in order to detect 'clean' speech segments. The method shown here is a very simple example of how the detection of speech segments can be achieved [17].

3.2.4.1. Zero crossing

The concept of zero-crossings is defined to be: "the number of times in a sound sample that the amplitude of the sound wave changes sign" For a 10ms sample of clean speech, the zero-crossing rate is approximately 12 for voiced speech and 50 for unvoiced speech [17]. For clean speech the zero crossing rate should also be useful for detecting regions of silence, as the zero-crossing rate should be zero.

Unfortunately, very few sound samples are recordings of perfectly clean speech. This means that often there is some level of background noise, that interferes with the speech, meaning that silent regions actually have quite a high zero-crossings rate as the signal changes from just one side of zero amplitude to the other and back again. For this reason a tolerance threshold is included in the function that calculates zero crossings to try and alleviate this problem. The thresholds work by removing any zero-crossings, which do not both start and end a certain amount from the zero value. Figure 3.9 shows the differences between voice and unvoiced segment.

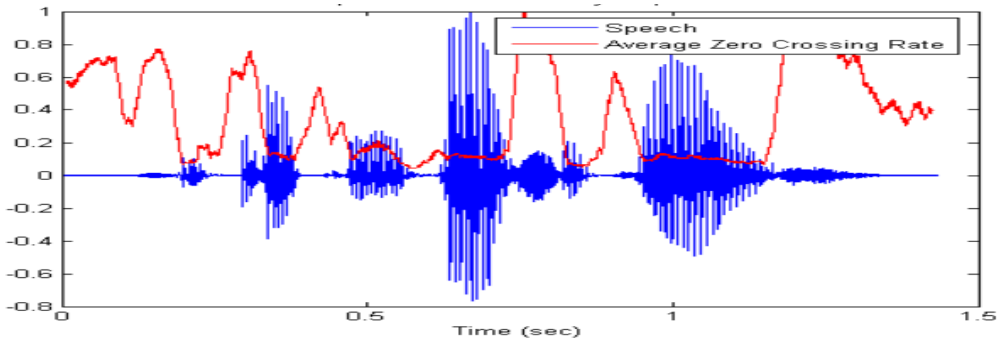


Figure 3.9: the discrimination between voiced and unvoiced regions of speech, or between speech and silence. Unvoiced speech has in general, higher zero-crossing rate.

3.2.4.2. Short-Time energy

In order to extract the feature sequences, the signal is first broken into non-overlapping short-term-windows (frames) of 20-millisecond length. Then for each frame, the Signal Energy is computed. If the energy exceeds certain threshold it considered voice signal where its not considered if the energy below the certain threshold assigned.

Let x_i (n); n = 1…N the audio samples of the i^{th} frame, of length N. Then, for each frame i the energy is calculated according to the equation:

$$E(i) = \frac{1}{N}\sum_{n}^{N}|X_i(n)|^2 \ldots\ldots\ldots (3.15)$$

This simple feature can be used for detecting silent periods in audio signals, but also for discriminating between audio classes [17].

The reason that this particular feature was selected (apart from their simplicity in implementation) is for simple cases, (where the level of background noise is not very high) the energy of the voiced segments is larger than the energy of the silent segments see figure 3.10.

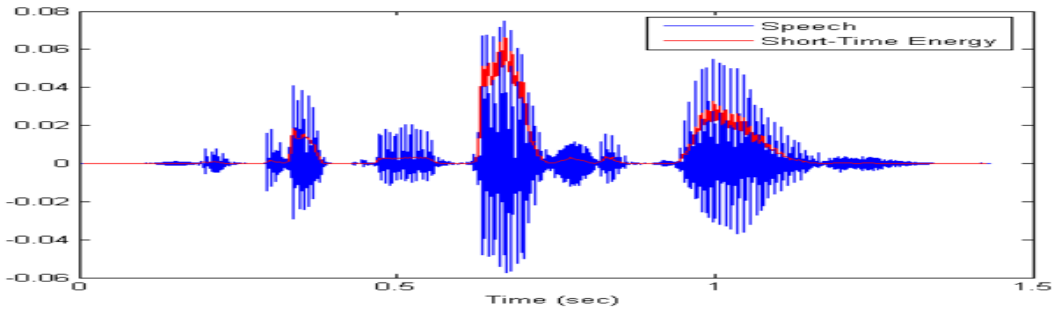


Figure 3.10: The plot distinguish between voiced and unvoiced speech segments, since unvoiced speech has significantly smaller short- time energy.

3.3. Chapter Summary

By applying the procedure described above, a set of Mel-frequency cepstrum coefficients (MFCC) is computed for each speech frame. The set of coefficients is called an acoustic vector. Thus each input speech utterance is transformed into a sequence of acoustic vectors.

Our data set contain a lot of different type of noise and a long period of silence due to the telephone speech natural .the pre- processing stage is needed and mightily, to gain good performance for a recognition.

Some of this technique was used in our project such that zero crossing and short time energy for the noise removal, MVN and RASTA filters for the noise reduction.

Chapter 4: Acoustic Based System

As described in the previous chapters, LID systems need classification models to model the languages that have to be identified. This chapter describes classification methods commonly used in this field and the recent state of art in dimension reduction and channel variability compensation.

4.1. Overview

Two very famous classifiers are widely used in LID because of their high performance and easy computation: Support Vector Machines and the Gaussian Mixture Model.

Support Vector Machines (SVM) is a data classification technique. An SVM produces a model from training data, which will predict the class of testing examples. The principle is "find a linear separating hyper plane with the maximal margin in this space" [18]. Then this linear hyper plane will determine the class of a testing sample according to: if it is on one side of the boundary or the other (for a binary classification). However, input data may not be linearly separable, SVM allows us to map the training data into a higher-dimensional space, then the optimal separating hyper plane can be computed in this space. The calculus of the SVM equations is explained in detail in this book [19].

The Gaussian Mixture Model (GMM) is an unsupervised classification technique which, applied to LID, tries to model the phonetic sound of the language by approximating their probability distribution. It is the modeling method retained for this project because it offers a large number of degrees of freedom which are easy to train. Section 4.2.4 describes how that can be approached by a set of background models called a Background Model.

4.2. Gaussian Mixture Models (GMM)

4.2.1. Theory Overview

In mixture models, we can represent the probability density function (pdf) as linear combination of basic functions multiplied by weighted quantities Figure 4.1. A mixture model of M components (or classes) has the following general form:

$$p(x) = \sum_{j=1}^{M} P(j)\, p(x|j) \quad \ldots\ldots\ldots\ldots (4.1)$$

In the above equation, $P(j)$ represents the weighted coefficients of mixture with $\sum_{j=1}^{M} P(j)$ =1, and $p(x|j)$represents the pdf of mixture model. Mixture model is good and efficient; since we can generate samples from its pdf by using Bayes' Theorem we can compute the probability of class j with a given value of x as:

$$P(j|x) = \frac{P(j)P(x|j)}{p(x)} \ldots\ldots\ldots (4.2)$$

GMM provides a good approximation for this principle. It represents the mixture model as a mixture of weighted Gaussian pdfs. Where the weighted coefficients are computed using Exception Maximization Algorithm (EM) and decision is made on the basis of the model of likelihood. There are two general steps involved in GMM process steps described in subsections below.

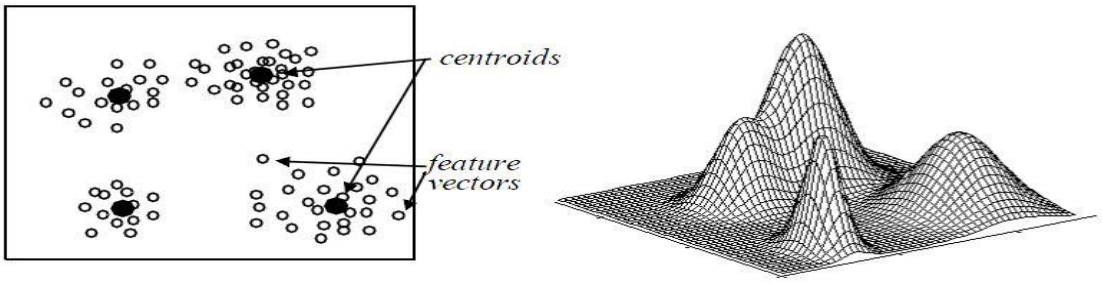


Figure 4.1: GMM model showing a feature space and corresponding Gaussian model.

4.2.2. Choosing Covariance Mixture Models

The first step in Gaussian mixture modeling is choosing the Covariance mixture model. Every mixture has its covariance matrix and pdf. There are three types of Covariance Mixture models; each one has different covariance and pdf:

1. Spherical model: the covariance matrix is an identity matrix multiplied by a scalar with pdf as: (model is shown in Figure **4.2-a**)

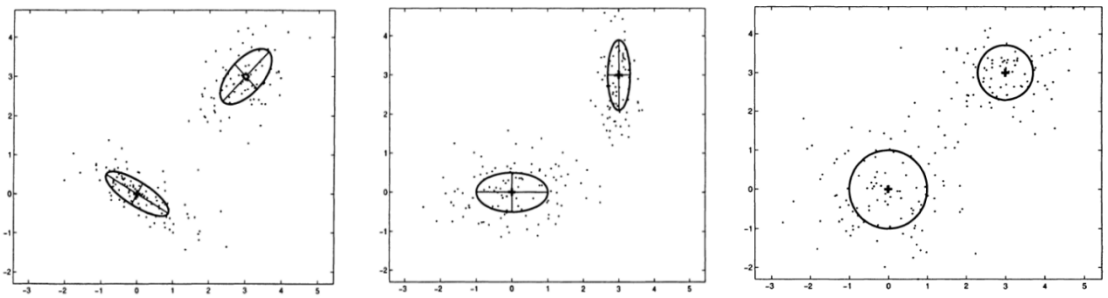
$$p(x|j) = \frac{1}{(2\pi\sigma j^2)^{d/2}} . \exp(-\frac{(x-\mu j)^2}{2\sigma j^2}) \ldots\ldots\ldots\ldots (4.3)$$

2. Diagonal model: the covariance matrix is diagonal, with pdf as: (model is shown in Figure **4.2-b**) $p(x|j) = \frac{1}{(2\pi)}$

$$p(\boldsymbol{x}|j) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}_j|^{1/2}} \exp\left\{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_j)^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_j)\right\} \quad \ldots\ldots\ldots (4.4)$$

3. Full model: the covariance matrix any d x d matrix, with pdf as: (model is shown in Figure **4.2-c**)

$$p(\boldsymbol{x}|j) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}_j|^{1/2}} \exp\left\{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_j)^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_j)\right\} \quad \ldots\ldots\ldots (4.5)$$



(a) (b) (c)

Figure 4.2: (a) Spherical Model (b) Diagonal Model (C) Full Model.

4.2.3. Estimation of GMM parameters

The second step in GMM process is the estimation of parameters of mean and covariance. We will introduce two methods: K-means clustering and expectation maximization algorithm.

4.2.3.1. K-means clustering

K-means is method for clustering items into k groups by minimizing the root square error between points and centroids see figure 4.3.

We first assumes a given number of clusters K, and we choose initial partitioning for data set which can determined randomly, or dynamically. Then we begin to map each point to nearest centroid, this operation is iterative and it includes estimation new mean for each cluster in that iteration and compute sum of square error.

We can stop the iteration process after the sum of square error is below some specific threshold. The main advantage of K-means algorithm that it becomes faster in computations for large value of k.

K-means clustering can be very useful to compute the GMM parameters, since EM is relatively is slow iterative process, so we can use K-mean clustering as initial estimation for GMM parameters, then applying EM algorithm.

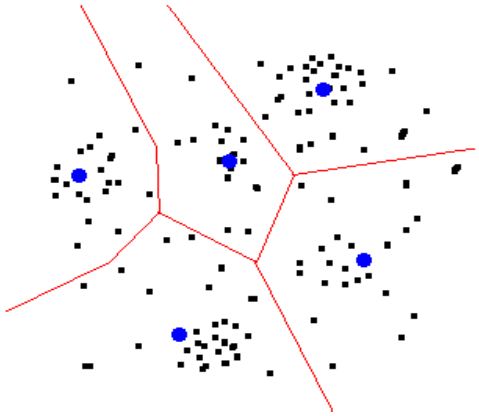


Figure 4.3: K-means clustering.

4.2.3.2. Exception Maximization (EM)

EM-algorithm is used to estimate the GMM parameters (means, covariance and weights), which then can be used to estimate the parameters of GMM, including the weight of each Gaussian, mean, and covariance matrix of each component (T.Nabney). EM process is based on minimizing the negative log of likelihood, with following error function:

$$E = -\sum_{n=1}^{N} \log p(x^n) \ldots\ldots\ldots (4.6)$$

EM is iterative process that modifies the parameters of GMM in each iteration, until to reach the minimum error value or the. Maximum permitted iteration .If we have total points of N and n_j is the number of points belong to class j then:

$$\mathbf{P(j)} = \frac{\boldsymbol{n_j}}{\boldsymbol{N!}} \ldots\ldots\ldots\ldots (4.7)$$

And mean of:

$$\mu_j = \frac{1}{n_j} \sum_{xi \in j} \boldsymbol{xi} \quad \text{..... (4.8)}$$

And it was proven in (T.Nabney) that for p(j), μ_j and the covariance are the followings:

$$P^{(m+1)}(j) = \frac{1}{N} \sum_{n=1}^{N} P^{(m)}(j|\boldsymbol{x}^n)$$

$$\boldsymbol{\mu}_j^{(m+1)} = \frac{\sum_{n=1}^{N} P^{(m)}(j|\boldsymbol{x}^n)\boldsymbol{x}^n}{\sum_{n=1}^{N} P^{(m)}(j|\boldsymbol{x}^n)} \quad \text{..... (4.9)}$$

Covariance can be calculated as the following:

$$\boldsymbol{\Sigma}_j = \frac{\sum_{n=1}^{N} P^{(m)}(j|\boldsymbol{x}^n)(\boldsymbol{x}^n - \boldsymbol{\mu}_j^{(m+1)})(\boldsymbol{x}^n - \boldsymbol{\mu}_j^{(m+1)})^T}{\sum_{n=1}^{N} P^{(m)}(j|\boldsymbol{x}^n)} \quad \text{..... (4.10)}$$

4.2.4 GMM-UBM

After introducing the world model in section 2.2.2, A Background Model could be any combination of models of different languages. Some research has been done about the composition of the training data for this background model [20]. It seems that the best performances are achieved for background models, which are specifically designed for each class. However, the huge drawback is that one needs to find the specific data to build a specific background model for each class. Thus, it may be preferable (because it is simpler to achieve) to train only one background model for every class. This is called an Universal Background Model (UBM) [21].

Thus, an UBM is a model common to all the training classes. One still needs to find the set of training data to train this model. This dataset should be large enough to represent all the expected data during the recognition task. However some research's showed that this dataset can belong to the same data as the data used for training the classes and still gives a promising result.

A GMM-UBM is an UBM where the model is a GMM. There are some advantages to this kind of UBM. This is a relatively easy unsupervised classification method and in recent

project showed how it decreased the training and testing computation time when the classes have been derived from the GMM-UBM.

This background model can be trained in two ways. One way is to pool the data from all populations together, and then a GMM with the number of desired components may be trained on all these data. The second way may be quicker, because one trains smaller GMMs, and it leads to the same result. For each population a GMM with the same number of components may be trained, then, models are combined by simply pooling all components and normalizing the mixture weights. Figure 4.4 represents these two ways of training. However, one needs to be careful with the training population of an UBM.

Indeed, all characteristics should be exactly equally balanced. For example, this means that each language should be equally represented; also that the population should contains as many male speakers as female speakers if it is a mixed system; and also contains as many data from one channel as another channel is several channels are represented. Otherwise the UBM could be biased.

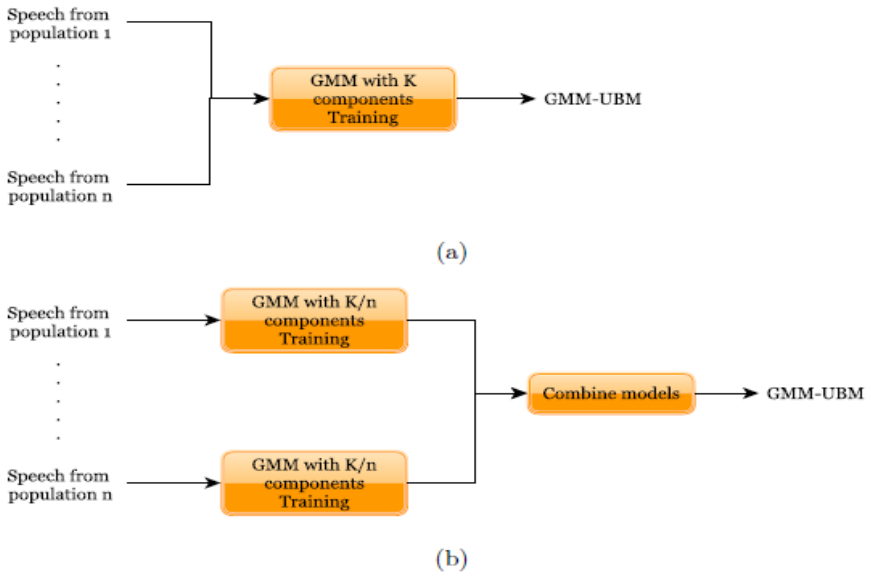


Figure 4.4: Block diagram representation of GMM-UBM training's techniques. (a) Data from populations are pooled together then the GMM model is trained. (b) A GMM is trained for each population then models are combined.

After explained system's component, and introduced the first technique in the Modeling the Design of the system GMM-UBM including each steps, algorithm used, enhancement technique, which implemented in our system is shown in Flowchart 1.

4.3 Support Vector Machine (SVM)

SVM is used for binary classification for data of two classes. The basic idea of SVM is to build N-dimensional hyper plane that separates data into two classes. SVM use hyper plane with a linear separation for between the data in a high dimension, but since most of data doesn't have linear separation; SVM provides Kernel function to support non-linear separation.

To represents the mathematical model of SVM, we have different mapping between inputs and their class as $\{X_i, Y_i\}$, where the input X_i has M-dimensions ($X_i \in R^M$), and each class Y_i belong to {-1,1}. Each hyper plane represented by values (**w**,**b**) where (**w**) is a vector and (**b**) is constant in a form of the following equation :

$$w.x + b = 0 \ldots\ldots\ldots (4.11)$$

Given a fact that the vector w is orthogonal to the hyper plane, so we can have the following function:

$$F(x) = sign(w.x + b) \ldots\ldots\ldots (4.12)$$

The above function separates data into $Y_{i\ in}$ {-1, 1}, with functional distance is greater or equal to 1:

$$Yi(w\ .xi + b) \geq 1 \ldots\ldots\ldots (4.13)$$

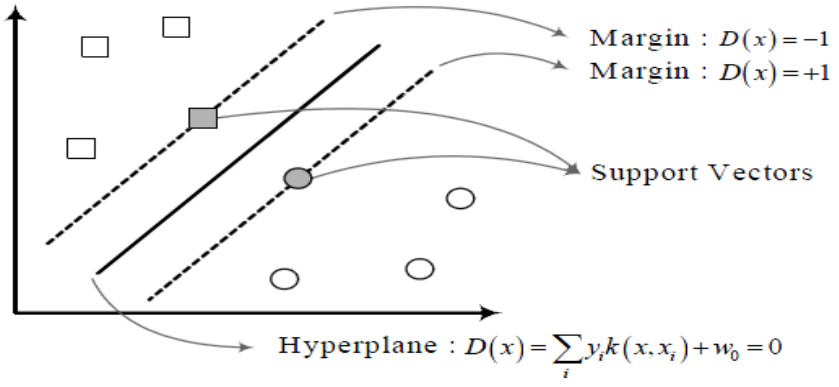


Figure 4.5: Illustration for SVM.

The hyper plane should maximize the margin of two classes, which contains some samples with each margin and called Support Vector (see simple illustration of SVM in Figure 4.5).

The general form of SVM hyper plane can be described as equation:

$$D(x) = \sum_{j=1}^{n} yi\, K(x\,.xi\,) + W0 \ldots\ldots\ldots (4.14)$$

Where K (x. xi) represents the Kernel function. No suppose we have text point of x', we want to measure the probability of x' belong to a class c; we can use the following equation:

$$P(class\ c \mid x') = \frac{Pc}{1+\exp\,(1-D(x')/D(xc))} \ldots\ldots\ldots (4.15)$$

In our Language recognition system, the class of Language with highest probability can be considered as the system output.

Kernel function used to transfer vectors from the original space into higher dimension space, examples of kernel functions:

- **Simple dot product** $k(x, y) = x.y \ldots\ldots (4.16)$
- **Vovk's polynomial** $k(x, y) = (x.y + 1)^p \ldots\ldots (4.17)$
- **Radial basis function** $k(x, y) = \exp\left(\frac{(x-y)^2}{2\delta}\right) \ldots\ldots (4.18)$
- **Sigmoid kernel** $k(x, y) = \tanh(k(x, y) - \theta) \ldots\ldots (4.19)$

4.3.1 GMM-SVM

The concept of GMM-SVM Super-vectors for Language identification is to combine the generative power of GMM with the discriminative properties of SVM.

The SVM vectors for all of the systems described are derived from the UBM model. Figure 4.6 illustrates the process behind the GMM-SVM Super-vectors, which in enrolment phase proceeds as follows [20]:

1. Acoustic feature vectors X are extracted from all available training utterances of the enrolling speaker.
2. A GMM model G with M Gaussians is obtained via MAP adaptation (means only) from a UBM model.
3. A super-vector V_x is constructed for Language L by concatenating the N-dimensional means for the S number of Speakers in each language, typically normalized by the corresponding standard deviation of each of the Gaussian mixtures in the adapted GMM model. For a GMM model composed of M Gaussian mixtures, it results is a S of $M * N$ dimensional vector.

4. An SVM classifier is trained using the V_x vectors computed for the target Language as positive examples (class t = +1), and a set of impostor language vectors V_i (common to all enrolment language) as negative examples (class t = −1).
5. The UBM model and the SVM parameters are stored as the language's fingerprint.

On verification stage, given an input speech utterance converted to a feature vector sequence, and a language model to be verified, the steps 2 − 3 of the procedure above are executed. The result is a Super vector, generated from the input sequence using the stored UBM model. The Super vector is evaluated against the Language's SVM and a decision is made whether the enrolled language and the input speech came from the same language (positive) or not (negative).

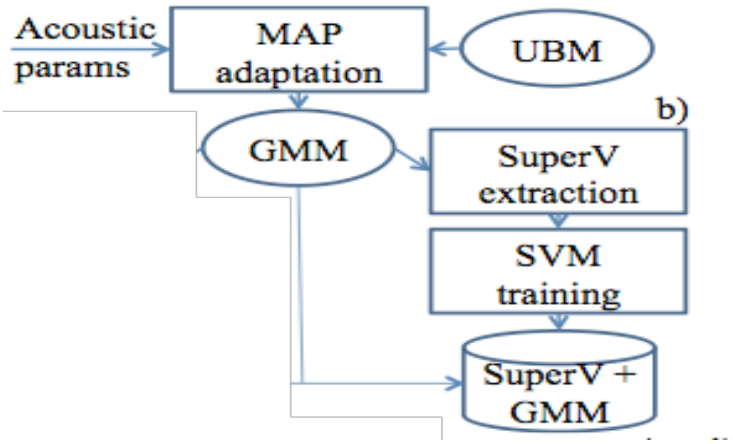


Figure 4.6: process behind the GMM-SVM Super-vectors.

After explained system's component, and introduced the second technique in the Modeling The Design of the system GMM-SVM including each steps, which implemented in our system is shown in Flowchart 2.

4.4 Push-Back From SVM to GMM

Instead of using traditional SVM scoring modern researches introduced the possibility of transferring the SVM parameters back to a GMM model and using standard log likelihood ratio scoring.

This technique mixed between the generative in the GMM modeling and the discriminative technique in the SVM modeling, it's obvious that GMM include all the vectors in the

calculation, which mean if there is a noise vector the GMM mean and variance will be affected and that what generative means, whereas, SVM focus on the support victors located on the marginal area around the separation hyper-plane, so if there a noise vectors the SVM will not affected as GMM, here and after this both idea a new technique was emanated to get the benefit from each modeling technique by push back the SVM back to get the GMM by taking the support vectors and to consider there point as new means in the new GMM this initial Construction completed by the UBM model to get the whole number of mixture desired. This new GMM contain the best between the GMM and SVM and give better result than each of this technique as shown in the new projects related to LID system [21]. The Figure 4.7 shows the basics in the pushing model.

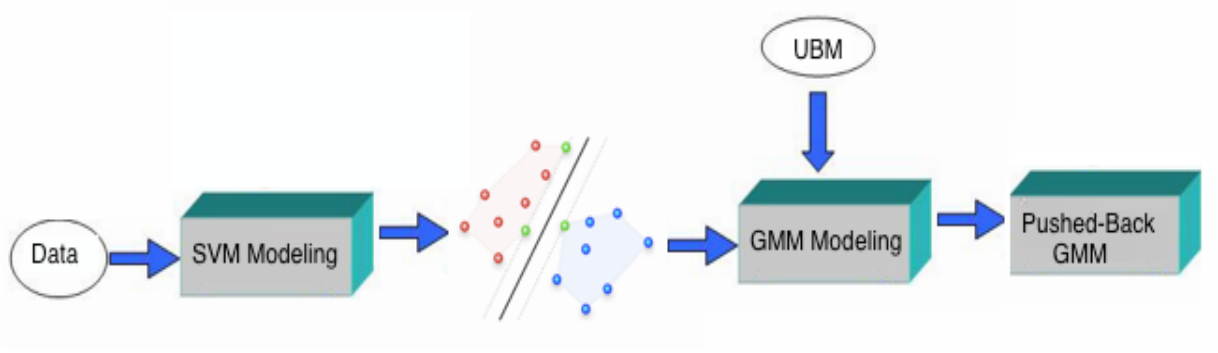


Figure 4.7: Pushing Model construction.

After introducing this new modeling technique, the implementation of this system was achieved, the system design which contain the all component shown in the Flowchart 3.

4.5 Intersession variability compensation

While research in the field of Language recognition and verification has been ongoing for many years, the greatest cause of errors still remains the same. In particular, in Language recognition, errors are due not only to the similarity among voice-prints of different Language, but also to the intrinsic variability of different utterances of the same Language. Moreover, performance is heavily affected when a model, trained in certain conditions, is used to recognize a Language of speaker voice collected via different microphones, channels, and environments. All these mismatching conditions are generally referred to as intersession variability.

More recently a method for directly modeling the inter-session variability has been proposed and has provided impressive reductions in verification error rates [21]. The motivation behind the proposed technique is to attempt to directly model session variability in the model space

without discrete categories and with less restrictive data labeling requirements. The proposed technique incorporates session differences into the Language modeling process.

- **MODELLING SESSIONVARIABILITY**

The approach to modeling the session variability in telephony-based Language verification adopted in this Project is to introduce a constrained offset of the Language's Gaussian mixture model mean vectors to represent the effects introduce by the session conditions. In other words, the Gaussian mixture model that best represents the acoustic observations of a particular recording is the combination of a session-independent Language model with an additional session- dependent offset of the model means. This can be represented in terms of the $R_Y \times 1$ concatenated GMM component means super-vectors as

$$\boldsymbol{m_h(s) = m + Dy(s) + Uz_h(s)} \ldots\ldots\ldots (4.20)$$

Where R_Y = MD where the GMM is of order M and dimension D. Here, the Languages is represented by the offset y(s) from the Language independent (or UBM) concatenated mean super-vector m, scaled by the $R_Y \times R_Y$ diagonal matrix D. To represent the conditions of the particular recording, designated with the subscript h, an additional offset of $Uz_h(s)$ is introduced where $z_h(s)$ is a low dimensional

Representation of the conditions in the recording and U is the low-rank transformation matrix from the constrained session variability subspace of dimension R_Z to the GMM mean super-vector space of dimension R_Y.

With this formulation both the language offset y(s) and the session factors $z_h(s)$ are assumed to belong to a standard normal distribution, N (0, I).

Ideally, a training algorithm will be able to accurately discern the session-independent language model y(s) in the presence of session variability. Due to the complexity of this technique we comprehensively had shown the Design of the system by explanation for each component in our vision to make it as simple as possible Flowchart 4.

Chapter 5: GMM tokenization and Phonotactic Based system

Language identification (LID) is the process of identifying the language of a spoken utterance so it is good to focus on the most discriminative high level features of languages, such as phones and words. In recent years, some tokenization methods with higher level information have been attracted great interests. Tokenization used to break a stream of text up into words, phrases, symbols, or other meaningful elements called tokens [23]. In this project, we present single Gaussian Mixture Model (GMM) tokenization and phone tokenization method followed by language model which it is n-gram. Language model assigns a probability to a sequence of m words. In the fields of probability, an n-gram is a contiguous sequence of n items from a given sequence of text or speech. The items can be phonemes, syllables, letters, words or base pairs according to the application [24].

The organization of this chapter is as follows: Phone tokenization presented in section 5.1 and GMM tokenization with language modeling system is presented in Section 5.2. Section 5.3 describes N-gram model.

5.1 Phone Tokenization

Phone tokenization is used as a front-end followed by n-gram language modeling to generate an estimate of the phone sequence of the utterance to be identified and language models are used to estimate the probability of this phone sequence being generated by a speaker of a particular language [21]. In general, phonotactic systems use observed phone sequences to construct a statistical language model for each language of interest [22]. Phonetic recognizer is used to convert a test utterance into a phone sequence, which is then scored against each language model so the language of the model with the highest score is hypothesized to be that of the utterance. This technique known as phone-recognition followed by language modeling (PRLM) which it is tokenizer for a phone recognizer that tokenizes the incoming speech signal into a series of tokens from which a statistical *n*-gram language model is derived [22] as shown in Figure 5.1.

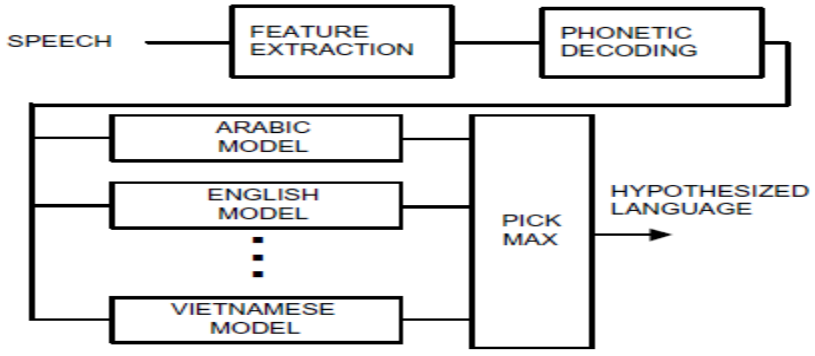


Figure 5.1: Diagram for a single-language PRLM [22]

5.2 GMM TOKENIZATION

GMM tokenizer converts the speech into a sequence of GMM token symbols which are the indexes of the Gaussian components scoring highest at every frame in the GMM computation [22] see figure 5.2. It is processes each acoustic frame and generates a sequence of indices of N-best GMM components. After we got the token symbols, we used n-gram scoring with n-gram language model to construct the back-end speaker classifier.

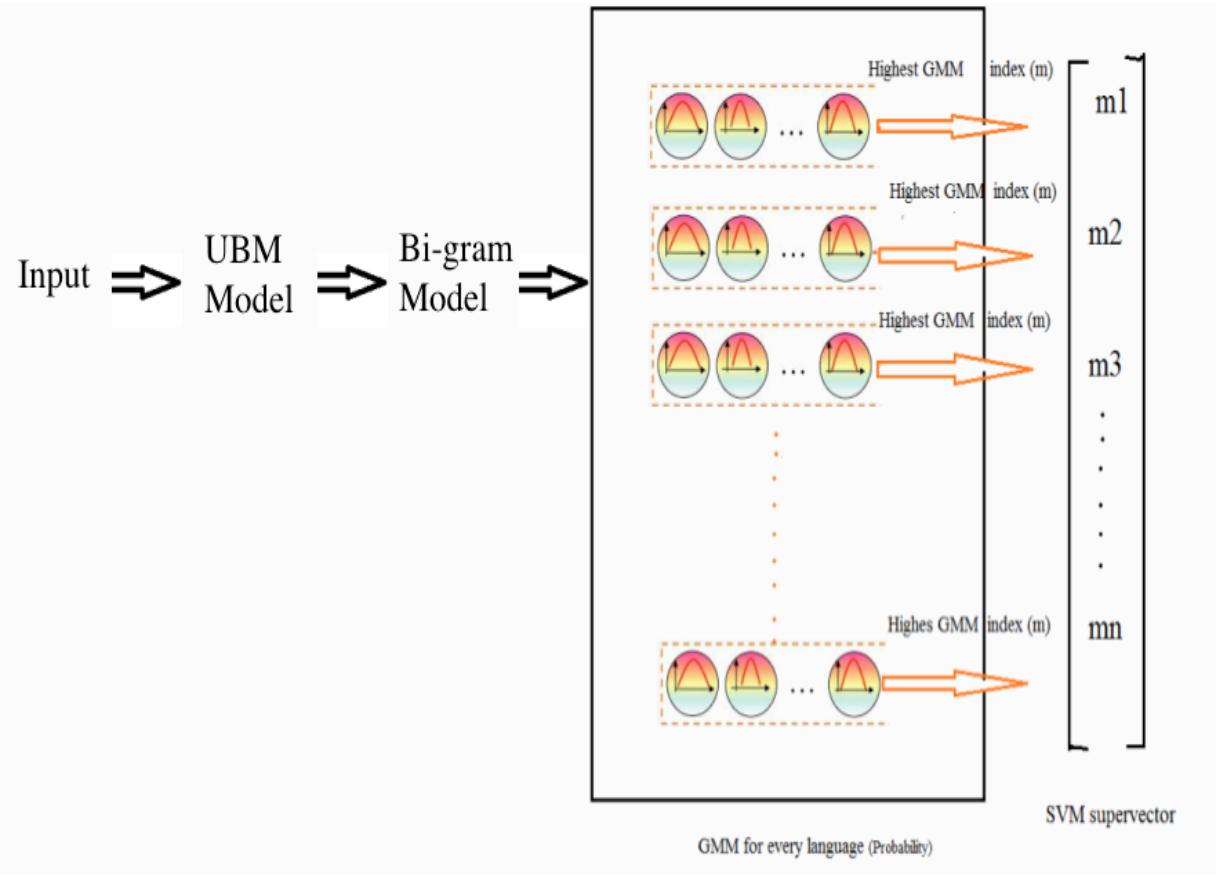


Figure 5.2: Block diagram for GMM tokenization.

The main components in the system of GMM tokenization are GMM toknizer and language model for each language as shown in Figure 5.3. Also we need to combine the language model scores (the back-end classifier) so Gaussian classifier was used. GMM tokenization works at frame level because the model training does not require the phonetic transcription of speech data so it can provide more tokens than those tokenizers at word and phone levels and this can solve the problem of limited speech data for the training and testing from one speaker [22].

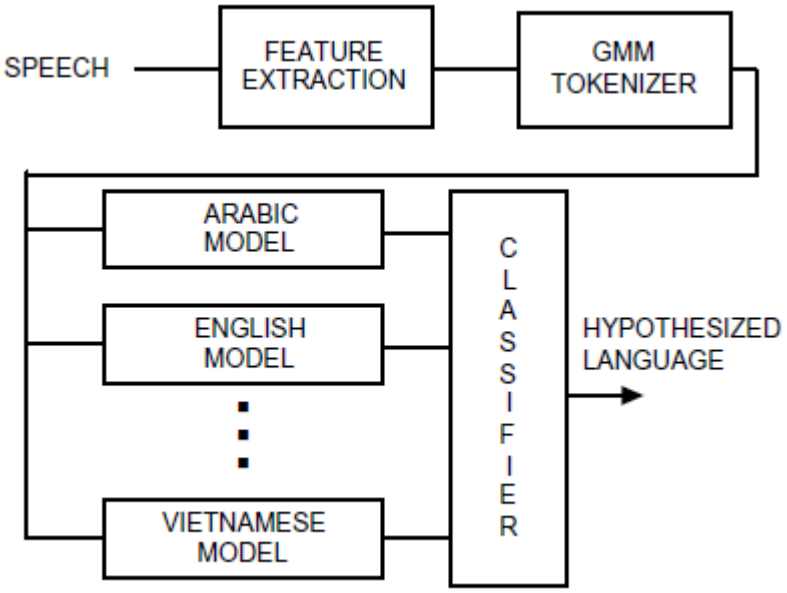


Figure 5.3: Diagram for LID System based on GMM tokenization and language modeling [22].

In general, the main idea of GMM tokenizer is that assigns the input feature vectors to partitions of the acoustic space. During training, speech is first processed by a feature extraction system that computes a vector of mel-warped cepstral parameters. The feature vector is created using the first ten cepstral parameters and delta-cepstra between two successive and two prior frames. The cepstral vectors are processed through a RASTA filter to remove linear channel effects. Next these features vectors are used to train a GMM. During testing, speech is decoded frame by frame. For each frame, the tokenizer outputs the index of the Gaussian component scoring highest in the GMM computation [22].

There are advantages for GMM tokenizer more than phone tokonizer even though they both characterize speaker at phonetic and spectral levels. Phone toknizer needs orthographically or phonetically transcribed training data and this is difficult for some languages so training of phone recognizers will be computationally expensive and laborious [23]. On the other hand, in GMM toknizer it does not require phonetically transcribed data, and it can be trained on the same acoustic data that is used to train the acoustic-based language recognition system so this is computationally less intensive than the phone recognizer [22].Another advantage for GMM toknizer that it captures another aspect of acoustic and phonetic characteristics of a speaker compared with phone tokenization. Since it is constructed at the frame level, GMM tokenizer can have more tokens than phone tokenizer from limited speech data of one speaker in speaker recognition task [22].

6.2 N-gram Scoring

N-gram considered a statistical high level features [8], it counts the repetition of the phoneme and its previous phonemes. We used Bi-gram –sometimes it is called 2-gram- for this purpose. Building bigram involves count the number of phoneme followed by other phoneme and this will lead to a two dimensional matrix with NxN, where N is the number of phonemes in the phoneme recognizer. The row in this matrix corresponds to the previous phoneme, while the column corresponds to the current found **phoneme.**

After getting the bi-gram matrix, we convert it to one-dimensional vector. This vector will be considered as a phonotactic feature representing the corresponding audio file. These features are used to build SVM classifier from these training sets.

In our project, we used bi-gram language model, which is governed by the following probability relation [2]:

$$\hat{P}(a \mid b) = \lambda_2 P(a \mid b) + \lambda_1 P(a) + \lambda_0 \tag{6.3}$$

Equation 4 was used to combine the form of recognition score η_i for a single GMM tokenizer case by using the n-gram scores from the hypothesized speaker model and the Universal Background Phone Model (UBMP) [12].

$$\eta_i = \frac{\sum_n (k(n)[S_i(n) - B(n)])}{\sum_n k(n)} \tag{6.4}$$

Where *n* is the index of n-gram GMM tokens, Si(n) represents the log-likelihood score from the *i*-th hypothesized speaker model, B(n) is the log-likelihood score from the UBPM, and K(n) is the weighting function based on the number of occurrences of a particular n-gram GMM token.

Chapter 6: Experiments and Results

The previous chapters describe the different type of LID systems, and the different components of an LID system, from speech enhancement and features extraction to classification methods. This chapter shows how we put into practice these different components to produce LID project.

This section describes the composition of the LID systems, which have been implemented, and how it has been designed for different types.

6.1. Experiments

The Experiments applied in this system comes in steps:

First, to find which acoustic feature is best able to distinguish one language from another, a test has been done which is for each feature to evaluate, in training a GMM for each language from a subset of the target languages and then testing them on a testing dataset. The feature that has the best average identification rate (the lowest error rate (ERR)) among the languages is selected.

The features are the 19 first coefficients of the MFCC, the previous coefficients plus their derivative (called delta coefficients), the previous coefficients plus the derivative of the derivative (called delta-delta coefficients) and the SDC 7-1-3-7.after that the GMM with 128 mixture (clusters) have been trained and then tested on 30 second speech files from NIST 1996 [see section 1.4].

Table 6.1 shows the result of the test. As we can see, the SDC seems to be the best feature to distinguish languages, as the different resource and papers we viewed [22]. Thus, SDC 7-1-3-7 will be used as the feature for the final system.

Table 6.1: Feature selection experiment: error rates and correct percentage of twelve languages by four different features.

	MFCC	MFCC+Delta	MFCC+Delta+Delta-Delta	SDC
ERR	42.33%	37.25%	40.06%	29.3%
Correct %	38.44%	44.91%	42.01%	48.15%

After the SDC computed and appended to the MFCC , now the MFCC+SDC is fixed for all the project now as the best type of feature which gain the best result.

Second, Our data set contain a lot of different type of noise and a long period of silence due to the telephone speech natural .the pre-processing stage is needed and mightily, to gain good performance for a recognition.

So second type of experiment is to distinguish the best technique for speech enhancement, which will fit the LID systems and give the best average identification rate.

The main technique we tested described in chapter 3, which is: Zero Crossing (ZC), Short-Time Energy, Cepstral Mean Subtraction (CMS), Mean-Variance Normalization (MVN), Feature Warping (FW) [12] and Rasta Filter.

Table 6.2: Best Enhancement method selection experiment: error rates and correct percentage of twelve languages by six different techniques.

	Rasta Filter	CMS	ZC+Energy	MVN	FW
ERR	30.14	28.96	29.06	28.66	26.25
Correct %	49.66%	52.1%	50.16%	52.83%	53.16%

After this Experiments we found that all of this techniques improve the so we decide to implement all of these technique to our system to improve the feature gained and to get better result.

6.1.1.GMM-UBM Experiment's and Result

Third, to initiate the GMM for each language model the K-mean algorithm [see section 4.2] was used to reduce the number of computations and improve the initial state to EM [see section 4.3.3] Algorithm to work until convergence is occurred .The number of iteration assigned to K-mean was 7 and for the EM Algorithm is 4 which was very general for the overall languages and prevent the over-fitting and worse result.

Table 6.3: the following table describe the two technique used in EM initialization which was the random initialization and k-mean initialization with number of EM iteration needed before convergence happened.

	Random	K-mean
Iteration for initial	1	7
Iteration Of EM	33	4

Fourth, to determine the number of components to use for the GMM for each language, a test was built. This test consists in training GMMs for the twelve languages with different numbers of components for each language. Then, each series of models was tested using the same testing data of (NIST96) and the correct percentage and EER are computed to evaluate which series best distinguishes each language.

The chosen numbers of components are 32,64,128,256, 512, 1024 and 2048 to cover a wide range of possibilities and keeping a reasonable computation time. Table 6.4 shows the different correct rate and the EER for this test.

Table 6.4: Result of a parameter selection experiment: Error rates of twelve languages with seven different numbers of components.

	32 component	**64 component**	**128 component**	**256 component**	**512 component**	**1028 component**	**2048 component**
ERR	30.14	28.96	24.61	23.28	23.06	22.52	21.92
Correct %	38.48%	42.16%	49.69%	52.83%	55.71%	59.94%	64.13%

As regards the result above, it appear that the larger number of components is the more successful in recognition rate is, which agrees with the results in different papers we reviewed. But in experiments we noticed that the recognition rates for Arabic and French language is still sharply bad regardless to the number of components figure 6.1 and figure 6.2 show the number of correct recognition for each language. One explanation for this result could be that there is not enough training data for these languages or there is need to enter a new enhancement method to speech or the 2 languages test data do not fit their training data.

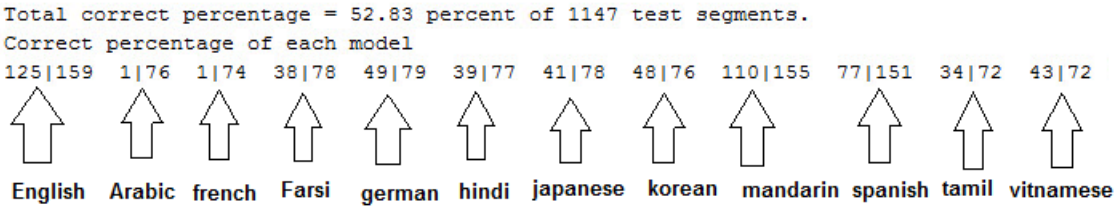


Figure 6.1: the recognition rate from the 256 component.

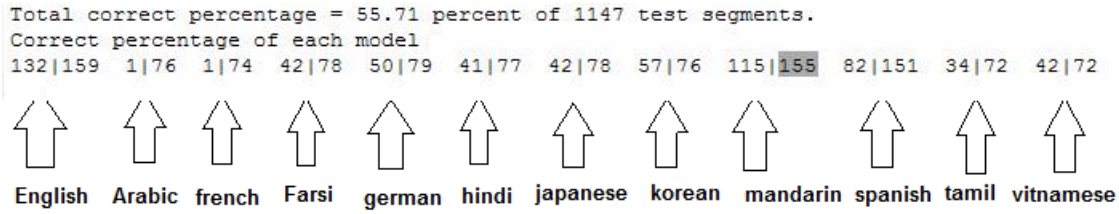


Figure6.2: the recognition rate from the 512 component

From the results of the test we conclude that more Gaussian components would be better for the GMM model, the following paper used 2048 components. However, at this time we only had access to personals computer, which does not have enough memory to deal with so many components.

In the end, the 128, 256, 512, 1028, 2048 has been chosen as the number of GMM components for each model. As regards the UBM, two were trained: one for male and the other for female speakers, both with the same number of components. The same training data was used as for the language models, to train a smaller GMM for each language. Then they were merged to create a UBM of 128, 256, 512, 1028, 2048 components for each gender to be used in map adaption later.

Several tests were conducted to evaluate the performance of the trained models. The miss and false alarm probabilities, for the whole GMM-UBM system, are used to evaluate the system for each number of components. As a result of this test, figure 6.3 shows a DET Curve of these probabilities, computed from each system.

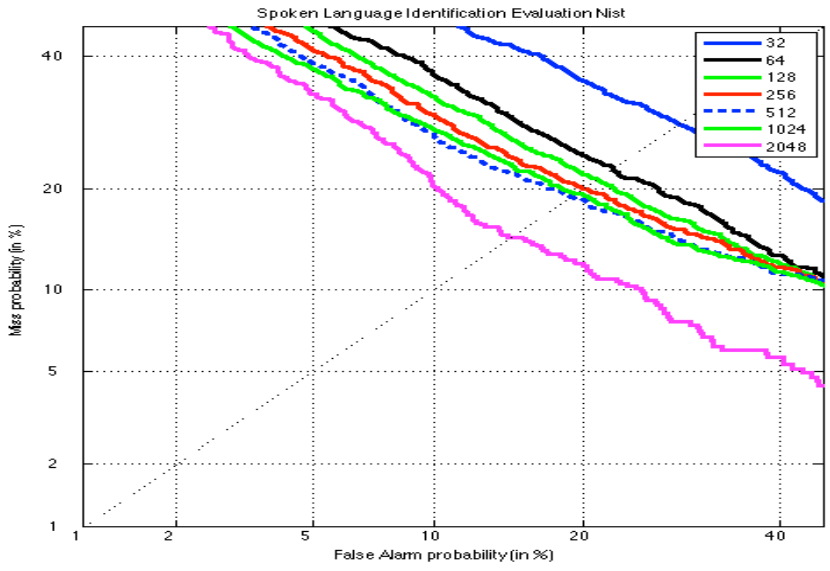


Figure 6.3: Plots of miss and false alarm probabilities for the detection performance of the GMM-UBM system.

Overall, the miss probabilities when false alarm probability is high (means a very small threshold) are in the same range as error rates from the first test with the EER and the correct % in table [6.4]. Which is still encouraging for the performance of the LID system.

The main feature we can obtain from this graph that when the curve become closer to the origin the efficiency of the whole system Increase as well. But, it's still not satisfactory result. However, to get satisfactory results we need to introduce new modeling technique. The following section describes the different Experiments and result.

6.1.2.GMM-SVM Experiment's and result

For the GMM-SVM system, MFCC+SDC was used for feature extraction due to the result in 6.1.1 (Table 6.1), and Rasta-Filter, MVN, CMS, FW and ZC-Energy for feature enhancement due to the result in 6.1.1 (Table 6.2).

In this step we did 7 experiments on different number of GMM components, which used to build the SVM super-vector by concatenating the GMM mean as described in section [4.3.4].

Table 6.5 shows the result obtained in this experiment's.

Table 6.5: Result of SVM evaluation on different number of components: Error rates of twelve languages with seven different numbers of components.

	32 component	**64 component**	**128 component**	**256 component**	**512 component**	**1028 component**	**2048 component**
ERR	27.15%	25.96	23.61	21.28	20.06	19.52	16.92
Correct %	64.48%	70.16%	79.32%	82.83%	83.71%	85.94%	88.13%

As regards the result above, it appear that the larger number of components is the more successful in recognition rate is, which agrees with the results in GMM-UBM system. But in experiments we obviously noticed that the recognition rates were improved than GMM-UBM. One explanation for this result could be that the natural of the SVM technique, which discriminative fit the telephone data speech more than the generative phenomena, which GMM-UBM offer.

For evaluation systems DET curve was obtained. The miss and false alarm probabilities using the DET curve [see 2.5], for the whole GMM-SVM system, are used to evaluate the system for each number of components. As a result of this test, From this curve we can decide which threshold is the best , it's obviously clear that the EER is lower than GMM-UBM system

which declare the overall system performance. figure 6.4 shows a DET Curve of these probabilities, computed from each system.

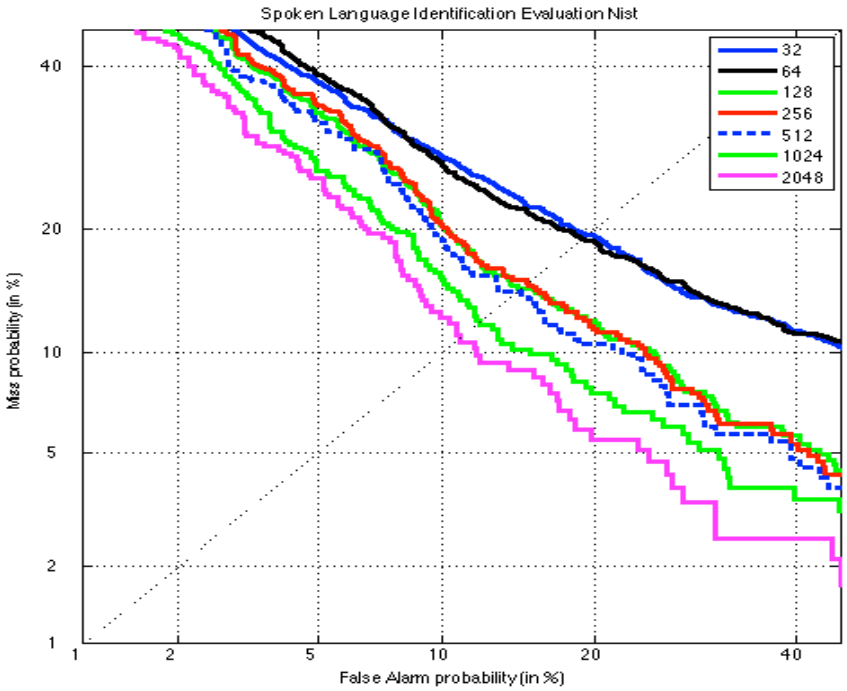


Figure 6.4: Plots of miss and false alarm probabilities for the detection performance of the GMM-SVM system.

Overall, the miss probabilities when false alarm probability is high (means a very small threshold) are in the same range as error rates from the first test with the EER and the correct % in table [6.5]. Which is still encouraging for the performance of the LID system.

The main feature we can obtain from this graph that when the curve become closer to the origin the efficiency of the whole system Increase as well. The result obtained was satisfactory result. However, we still seeking more accurate system to get satisfactory results we need to introduce new modeling technique. The following section describes the different Experiments and result.

6.1.3.Push-Back Model Experiment's and result

After introducing the both systems GMM-UBM and the GMM-SVM a new technique was shown ahead, In this technique we will mix the two systems above and getting the best properties found in each of them this technique called Push-Back Model [see section 4.4], another seven experiments was applied here to check if there is improvement to the systems. Table 6.6 shows the results obtained.

Table 6.6: Result of Pushback evaluation on different number of components: Error rates of twelve languages with seven different numbers of components.

	32 component	64 component	128 component	256 component	512 component	1028 component	2048 component
ERR	22.83%	20.96%	19.11%	17.05%	16.66%	15.32%	13.23%
Correct %	64.91%	77.16%	80.32%	84.75%	85.43%	86.94%	89.13%

As regard to the result above, it's appear that there is enhancement on the system and the system is more successful in recognition rate is, which agrees with the results we viewed in different papers such that [19]. But in experiments we obviously noticed that the recognition rates were improved than GMM-UBM and GMM-SVM But there is still error which caused this error rate. One explanation for this is the variability in the data set due to the differences in environment, hardware etc., so a new technique was introduced to solve this problem which is Intersession Variability compensation which discussed and explained in section [4.5].

6.1.4.Inter-Session Variability Compensation Experiment's and result

This technique was the hardest in understanding and in the implementation not to forget the exhaustive time in the computation to get the result, what make it as a challenge to introduce it and to be close from the state of art in this field.

As regard to all of these obstacles, the implementation was done and to evaluate this technique we start in steps:

First, to determine which size of U-Matrix is a dominant in view of Enhancement rate and computation times we fixed the number of the component in the GMM to 128 and start varying in the u-Matrix size. Table 6.7 shows the result where Figure 6.5 shows the DET curve for different size of U-Matrix.

Table 6.7: Result of Inter-session Variability Compensation evaluation with different size of U-matrix and fixed GMM component 128.

U-Matrix Size	**50**	**100**	**200**	**300**	**400**	**500**	**600**
Result	84.12%	85.67%	86.22%	86.45%	86.58%	86.91%	85.72%
Enhancement	1.8%	3.32%	3.9%	4.13%	4.26%	4.59%	3.37
Time Computation	~2-day	~2 day	~3day	~5day	~6day	~1week	~10day

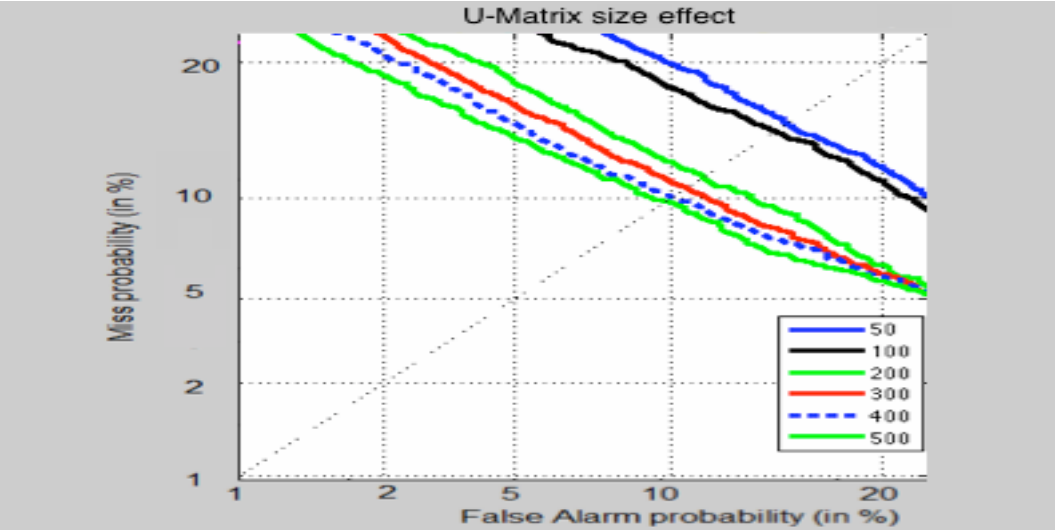


Figure 6.5: DET Curve for the U-matrix size Effect on the system Efficiency

From the data obtained we can plot the relation between the Size of the U-Matrix and the enhancement of the system which shown in the Figure 6.6.

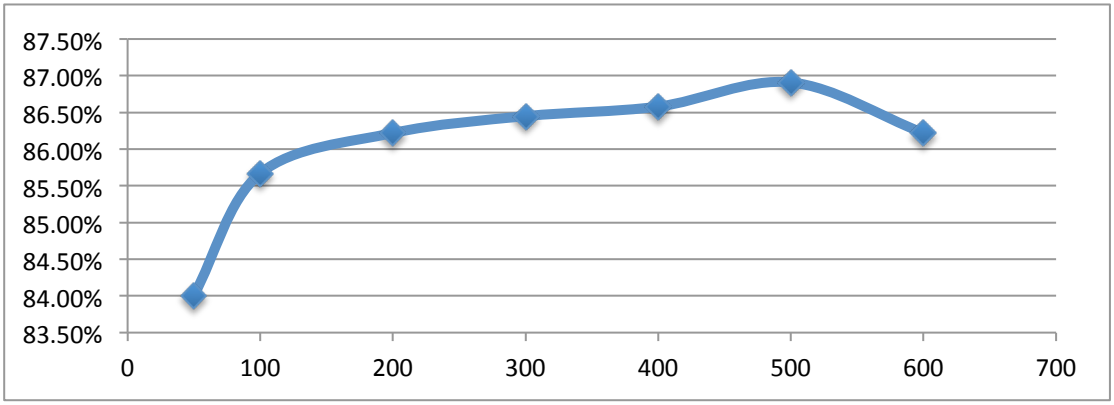


Figure 6.6: Plot defined the relation between the U-Matrix size and the efficiency

From the results of the test we conclude that more size in -matrix means better Enhancement, but after 300 we noticed that the improvement start to be fixed until 600 where the enhancement start to decrease, this phenomena happened when the U-matrix start to considered some segment from the speech in the Variability model. So Depends on the data observed the 200 size is picked up where better enhancement in efficiency and lower time consumption.

Second, to check the maximum efficiency to our system we did an experiment where the number of component is 2048 (maximum result from the previous systems) and with 200 U-Matrix size fort the intersession variability compensation model. The result was shown in table 6.8.

Table 6.8: Result of Inter-session Variability Compensation evaluation with fixed size of U-matrix which 200 and different GMM component.

	32 component	64 component	128 component	256 component	512 component	1028 component	2048 component
ERR	20.15%	21.96	19.61	17.28	17.06	15.52	10.92
Correct %	66.42%	70.81%	82.32%	84.26%	87.26%	89.91%	93.52%

As regard to the result above, it's appeared that there is a noticeable enhancement on the system and the system is more successful in recognition rate. The efficiency of overall system is very acceptable now over the 12 noisy language data set. The new technique improved the system as expected.

In this technique we can said that we finished the state of art technique in the Acoustic-Based System which done with a very satisfactory result with 93.5%

6.1.5.GMM Tokenization Model Experiment's and result

In the acoustic Based system we focused on the Acoustic feature only where there another method now appears in the new recognition world, the experiments in this chapters will be on the GMM –Tokenization technique that described in section. Table 6.9 shows the result of these experiments.

Table 6.9: Result of GMM Tokenization evaluation on different number of components: Error rates of twelve languages with seven different numbers of components.

	32 component	64 component	128 component	256 component	512 component	1028 component	2048 component
ERR	28.22%	26.36	24.71	21.06	20.91	18.42	15.23
Correct %	55.48%	62.16%	72.32%	80.83%	83.71%	86.54%	89.34%

As regard to the result above, it's appeared that this type of system has good recognition rate as new technique, whereas , the time in computation is less than GMM-UBM and GMM-SVM in modeling and in the evaluation process, while this system still need the MFCC and the GMM to calculate the super-vector.

In this technique we can said that we finished this type of techniques which privilege the GMM to be more realizable to describe the grammar of the language and gain very acceptable result with 89.34%.

The Figure 6.7 shows the DET curve for the GMM-Tokenization over different number of GMM components over NIST evaluation data.

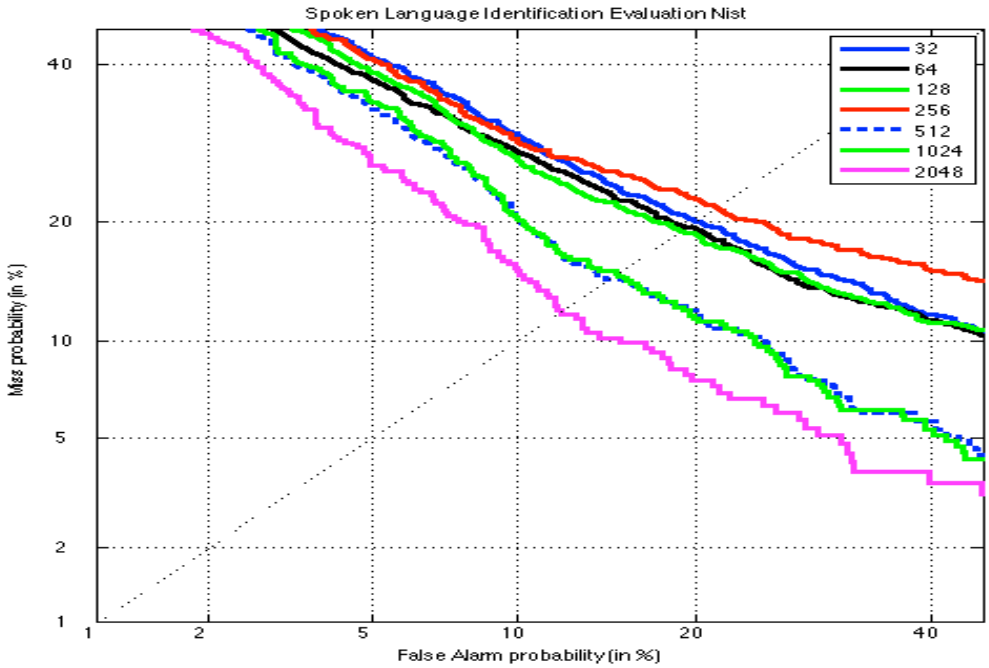


Figure 6.7: DET Curve for the U-matrix size Effect on the system Efficiency

Overall, the miss probabilities when false alarm probability is high (means a very small threshold) are in the same range as error rates from the first test with the EER and the correct % in table [6.9]. Which is encouraging for the performance of this technique.

6.1.6.Fused System Experiment's and result

After introducing and testing the different type of LID systems ,recently, a new concept with idea of mixing all the systems in one system, this kind of system Featuring that if one system fail at some point the second system won't.

so we fused the Best system from each following system : GMM-UBM ,GMM-SVM, Push-Back Model, Intersession-Variability Compensation, GMM-Tokenization. The Table 6.10 shows the result of this experiment.

Table 6.10: Result of Fused System evaluation for mixing all the systems

	Result	**ERR**
GMM-UBM (2048)	64.13%	21.92
GMM-SVM (2048)	88.13%	16.92
Push-Back (2048)	89.13%	13.23
Intersession Variability Compensation (U-Matrix 200, GMM 2048)	93.52%	10.92
GMM-Tokenization (2048)	89.34%	15.23
Fused System	96.37%	6.3

As shown in the table 6.10, we can see that the fused system is the most powerful approach. That shown in the result offered which 96%, where no system independently can reach it. By this result we finish our project by reaching the state of art in this field and get promising result with 96%

Chapter 7: Conclusion

This report was about automatic spoken language identification. It shows the technical background and steps necessary for understanding and building an LID system: including relevant features, classification methods and features enhancement techniques.

For the purpose of this project, LID systems were built by implementing some of the all the techniques mentioned in this report. Over more than 120 different Experiments on 6 different systems, the implementation was tested on twelve languages: English, Hindi, Japanese, Korean, Mandarin (Mainland and Tai-wan), Spanish (original, Mexican) and Tamil, German, Vietnamese, French, Arabic (Egyptian), Farsi. The Final result of the system reach (96% on NIST 2001 test data), this result obtained from the Fused System, which was the best approach after this long trip from testing and implementation different systems. Our goal in the introduction 520 was achieved successfully and we hope that this work can help and benefits another people in the future.

Bibliography

[1] Zissman, M. and Berkling, K. (2001). Automatic language identification. Speech Communication, [Accessed date: 26/5/2014].

[2] Zissman, M. (1996). Comparison of four approaches to automatic language identification of telephone speech. Speech and Audio Processing, IEEE Transactions on,[Accessed date: 26/5/2014].

[3] Reynolds, D. (2008). Gaussian mixture models. Encyclopedia of Biometric Recognition, [Accessed date: 26/5/2014].

[4] Marc A. and Zissman, k. (2001), Automatic Language Identification of Telephone Speech, [Accessed date: 27/5/2014].

[5] The Linguistic Data Consortium (LDC), https://www.ldc.upenn.edu/about.

[6] Tong, R., Ma, B., Zhu, D., Li, H., and Chng, E. (2006). Integrating acoustic, prosodic and phonotactic features for spoken language identication. In Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on, volume 1, [Accesssed date: 27/5/2014].

[7] Shannon, C. (1949). Communication in the presence of noise.Proceedings of the IRE, [Accessed date: 27/5/2014].
[8] Brigham, E. (1988). The fast Fourier transform and its applications. Prentice Hall, [Accessed date: 29/5/2014].

[9] A. Martin, G. D. (2005). The 2005 nist language recognition evaluation plan http://www.nist.gov/speech/tests/lang/2005/LRE05EvalPlan-v5-2.pdf, {Accessed date: 29/5/2014].

[10] Md Jahangir Alam, Patrick Kenny, Douglas O'Shaughnessy, "ROBUST SPEECH RECOGNITION UNDER NOISY ENVIRONMENTS USING ASYMMETRIC TAPERS" . Canada, [Accessed date: 29/5/2014].

[11] Manish P. Kesarkar, (November2003). "FEATURE EXTRACTION FOR SPEECH RECOGNITON", [Accessed date: 30/5/2014].

[12] Matthew Gibson. (July 2004). "EFFICIENT MLLR" [Accessed date: 30/5/2014].

[13] Torres-Carrasquillo, P., Singer, E., Kohler, M., Greene, R., Reynolds, D., and Deller Jr, J. (2002). Approaches to language identification using gaussian mixture models and shifted delta cepstral features. In Seventh International Conference on Spoken Language Processing, [Accessed date: 30/5/2014].

[14] Berouti, M., Schwartz, R., and Makhoul, J. (1979). Enhancement of International Conference on ICASSP'79., volume 4, IEEE, [Accessed date: 31/5/2014].

[15] Hermansky, H. and Morgan, N. (1994). Rasta processing of speech. Speech and Audio Processing, IEEE Transactions on, [Accessed date: 31/5/2014].

[16] G. Saha. (2008) "A New Silence Removal and Endpoint Detection Algorithm for Speech and Speaker Recognition Applications" , Department of Electronics and Electrical Communication Engineering Indian Institute of Technology, (Accessed date 31/5/2014).

[17] Huang, X., Acero, A., Hon, H., et al. (2001). Spoken language processing. Prentice Hall PTR New Jersey, [Accessed date: 31/5/2014].

[18] Vapnik, V. (1998). Statistical learning theory. Inc., New York, [Accessed date: 1/6/2014].

[19] Reynolds, D., Quatieri, T., and Dunn, R. (2000). Speaker verification using adapted gaussian mixture models. Digital signal processing, [Accessed date: 1/6/2014].

[20] Reynolds, D. (1995). Speaker identification and verification using Gaussian mixture speaker models. Speech communication, [Accessed date: 2/6/2014].

[21] Abualsoud Hanani, Michael Carey and Martin Russell. (2010). "Improved Language Recognition using Mixture Components Statistics", [Accessed date: 2/6/2014].

[22] Pedro A. Torres-Carrasquillo1, 2, Douglas A. Reynolds2 and J.R. Deller, Jr.1, "Language Identification using Gaussian Mixture Model Tokenization", [Accessed date: 2/6/2014].

[23] http://en.wikipedia.org/wiki/Tokenization, [Accessed date: 2/6/2014].

[24] http://en.wikipedia.org/wiki/N-gram, [Accessed date: 2/6/2014].