

Evaluation of speaker adaptation algorithms for public access IVR server

Prepared by: O.P. Noah,

Prepared for: Prof Daniel Mashao

University of Cape Town



Speech Technology and Research Group
Department of Electrical Engineering
University of Cape Town
Rondebosch, Cape Town, SA

Submitted to the University of Cape
Town in fulfilment of academic
requirements for the Degree of
Masters in Science in Electrical Engineering

August 2006

Declaration

I declare that this dissertation is work undertaken by myself. I am submitting it for consideration to be admitted to a Master of Science degree in Electrical Engineering at the University of Cape Town. To my knowledge, it has not been submitted before for any degree or examination at this or any other University.

Signature: _____

Ofentse Noah

Summary

IVR servers are telephony-based applications that allow a user to interact with a series of menus in order to retrieve information. For many years, users have commonly used key pads to interact with IVR servers. As a result users have experienced numerous benefits in obtaining information from these servers. However, speech recognition is fast becoming a popular way of interacting with IVR servers.

In order to design a speech recognition system, two methods can be used. The system can be built using a speaker dependent (SD) method, meaning it is trained and tested on one speaker. Otherwise, the system can be built using a speaker independent (SI) method, meaning many speakers are used for training. SD systems perform very well on the speaker they are trained on. This is because the system models the speaker well. On the other hand, they do not perform well on a speaker they are not trained on. While SI systems do not perform as well as SD systems for the speaker they were trained on, they perform well on the speakers they have not been trained on. SD systems require many hours of training data per speaker. This is similar in SI system but they only require a small group of people. In light of the public access IVR server, it is not feasible to train SD system for a large population.

In order to get the performance of the SI system to approach SD-like performance, this study uses speaker adaptation. The purpose of the study is to evaluate the suitability of speaker adaptation for the public access IVR server. In the study's findings, the speaker adaptation algorithms require 30 seconds of adaptation data to yield improved performance. The study also shows that an extension of the Levenshtein edit distance to measure sentence correctness, shows a more realistic result compared to the existing HTK sentence correctness measurement method.

Email: ofentse@star.za.net

WWW1: <http://www.star.za.net/>

Acknowledgments

I would like to express my most sincere gratitude to my supervisor, Prof. Daniel. J. Mashao for all his guidance through the journey of this thesis. With his invaluable supervision, I have managed to gain insight as how to take initiative and edge towards progress in the field of research. It has given me valuable growth which an engineer needs. I am particularly grateful for the skills, discipline and determination I have acquired under his supervision. I thank my colleagues and friends in the STAR Group for support, useful insights and the happy laughs. I would also like to thank my family, in particular my parents who have nudged me on throughout my studying career. Most importantly this study is in memory of my late grandmother, Caroline Zwakala, who has encouraged me to reach the finishing line of this study. And not to forget my siblings who have given me assistance where needed. Lastly, I thank God, Almighty.

Contents

Declaration	ii
Summary	iii
Acknowledgments.....	iv
Contents	v
List of Figures	viii
List of Tables	x
Glossary of Terms	xi
1 Introduction.....	1
1.1 An overview of speech recognition.....	2
1.2 How speech recognition works	3
1.3 Public access IVR server.....	6
1.4 How does the public access IVR server work?	8
1.5 Objective of the study.....	9
1.6 Speaker adaptation	9
1.7 Scope and limitations	10
1.8 The approach	10
1.9 Structure of thesis.....	11
1.10 Summary	12
2 Speech Recognition Fundamentals	13
2.1 Feature Extraction: Front-end	13
2.1.1 Analogue-to-digital Conversion	14
2.1.2 Pre-processing.....	17
2.1.3 Feature extraction in the front end.....	18
2.1.3.1 Linear Prediction Coding	18
2.1.3.2 Filter-bank analysis	20
2.1.3.3 Auditory Modelling feature extraction.....	21

2.1.3.4 Other feature vector components	22
2.2 Speech modelling	24
2.2.1 Defining HMM's.....	25
2.2.2 The Three HMM problems	27
2.2.3 Problem 1 solution: <i>Evaluation</i>	28
2.2.4 Problem 2 solution: <i>Revealing the state sequence</i>	31
2.2.5 Problem 3 solution: <i>Training</i>	32
2.2.6 Continuous Density HMMS (CDHMM's).....	34
2.3 Applications of speech recognition	35
2.4 Summary	38
3 HTK	39
3.1 Data preparation	40
3.2 Training the system	47
3.3 Recognition	50
3.4 Evaluation.....	51
3.5 Summary	53
4 Speaker adaptation	54
4.1 Maximum Likelihood Linear Regression.....	55
4.1.1 Linear Regression: <i>Expressing the output as a linear combination</i>	55
4.1.2 Maximum Likelihood Estimations (MLE)	57
4.1.3 MLLR: <i>Computing the regression classes</i>	59
4.2 Maximum a-Posteriori (MAP) Adaptation.....	62
4.2.1 Conjugate Priors	63
4.2.2 Computing the MAP estimate.....	64
4.2.3 Solving the outstanding quantities: prior information	64
4.3 A combination of MLLR and MAP	65
4.4 Differences between the algorithms	66
4.5 Summary	67
5 Evaluation of the system.....	69
5.1 Experimental database and task	69
5.2 System design and implementation	70
5.3 Evaluation.....	71
5.3.1 Verification of the speaker adaptation algorithms	72
5.3.2 Performance of speaker adaptation algorithms on SI systems.....	74

5.3.3	System parameter variation	75
5.3.4	Using unsupervised speaker adaptation	83
5.3.5	A comparison between sentence accuracy and Levenshtein edit distance	86
5.3.6	A comparison of the amounts of data used	95
5.4	Discussion	97
6	Conclusions	100
	Bibliography	104

List of Figures

Figure 1-1: An example of an SQL database	2
Figure 1-2: A generalization of speech recognition applications	3
Figure 1-4: Fundamental speech recognition model	4
Figure 1-3: Speech recognition research milestones [20]	6
Figure 1-5: Public Access IVR server	8
Figure 2-1: The fundamental model of a speech recogniser [102]	13
Figure 2-2: Conversion of speech into parametric speech vectors [48]	14
Figure 2-3: Sampling of a speech signal	15
Figure 2-4: Quantization [101]	16
Figure 2-5: Coding example	16
Figure 2-6: Pre-emphasis of a speech waveform	17
Figure 2-7: Filter-bank analysis [48]	20
Figure 2-8: Mercedes Benz with Linguatronic II speech system [33]	37
Figure 2-9: Linguatronic II microphone array	38
Figure 3-1: An overview of HTK Tools [48]	40
Figure 3-2: A sample configuration file for HTK	42
Figure 3-3: A sample dictionary for HTK	43
Figure 3-4: List of utterances for speech recognition training	44
Figure 3-5: An HTK compatible list of words	45
Figure 3-6: An HTK compatible list of phonemes	46
Figure 3-7: A list of commands, usually called mkphones.led	46
Figure 3-8: Triphone cloning text file, mktri.led	48
Figure 3-9: Control script tree.hed	49
Figure 4-1: Speaker adaptation model transformation [48]	54
Figure 4-2: Global Transformation for MLLR	60
Figure 4-3: Binary class regression tree	61
Figure 4-4: Diagram for global transformation	62
Figure 5-1: Performance of the adaptation algorithms on SD systems	73
Figure 5-2: The performance of speaker adaptation algorithms on SI systems	74
Figure 5-3: Mixture incrementing of SI systems	76
Figure 5-4: A comparison between global MLLR and regression class MLLR with one regression class.	77

Figure 5-5: MLLR speaker adaptation with increasing number of regression classes.....	78
Figure 5-6: MAP adaptation with decreasing data scaling values.....	79
Figure 5-7: MLLR+MAP adaptation	81
Figure 5-8: A comparison between HTK sentence correctness and Levenshtein distance	87
Figure 5-9: Performance measurement comparison between global MLLR and Levenshtein distance.....	88
Figure 5-10: A comparison of sentence recognition evaluation methods	90
Figure 5-11: A comparison of sentence recognition evaluation methods after MAP adaptation...	91
Figure 5-12: A comparison of sentence recognition evaluation methods.	92
Figure 5-13: A comparison of sentence evaluation methods	94

List of Tables

Table 1-1: A comparison of telephones and computers in South Africa [18]	6
Table 3-1: A table specifying a batch of speech data files	41
Table 3-2: A comparison between a reference sentence and an output hypothesis of a speech recogniser.	52
Table 4-1: Differences between algorithms	67
Table 5-1: System performance after 60 seconds of adaptation data	82
Table 5-2: Results obtained after using 35 utterances for adaptation with 32 regression classes ..	83
Table 5-3: Results for unsupervised adaptation after using over 60 seconds of adaptation data. ..	84
Table 5-4: Results obtained after using over 60 seconds of adaptation data for adaptation.....	85
Table 5-5: Amounts of speech data used.....	95
Table 6-1: Leno's tests on Ofentse's speech recognition system	Error! Bookmark not defined.
Table 6-2: Ofentse's tests on Leno's speech recognition system	Error! Bookmark not defined.
Table 6-3: Francois' tests on Ofentse's speech recognition system	Error! Bookmark not defined.
Table 6-4: Ofentse's tests on Francois' speech recognition system	Error! Bookmark not defined.
Table 6-5: Hlonie's tests on Ofentse's data	Error! Bookmark not defined.
Table 6-6: Ofentse's tests on Hlonie's speech recognition system ..	Error! Bookmark not defined.
Table 6-7: Francois's tests on Hlonie's speech recognition system ..	Error! Bookmark not defined.
Table 6-8: Francois' tests on Leno's speech recognition system	Error! Bookmark not defined.
Table 6-9: Leno's tests on Francois' speech recognition system	Error! Bookmark not defined.
Table 6-10: Leno's test on SI 1	Error! Bookmark not defined.
Table 6-11: Leno's tests on SI 2	Error! Bookmark not defined.
Table 6-12: Ofentse's tests on SI 1	Error! Bookmark not defined.

Glossary of Terms

ADC	–Analogue-to-digital Converter
ASR	–Automatic Speech Recognition
ANN	–Artificial Neural Networks
DTMF	–Dual Tone Multi Frequency
EM	–Expectation Maximization
G	–Global
HMM	– Hidden Markov Models
HTK	–Hidden Markov Model Toolkit
IVR	–Interactive Voice Response
LPC	–Linear Prediction Coding
MAP	–Maximum a-Posteriori
MLE	–Maximum Likelihood Estimate
MLLR	–Maximum Likelihood Linear Regression
RC	–Regression Class
SD	–Speaker Dependent
SI	–Speaker Independent
SQL	–Structure Query Language
WER	–Word Error Rate

Chapter 1

1 Introduction

Speech recognition is naturally a focus of research [1] that many researchers believe can be applied to many spheres of life. This is indeed the case as researchers have managed to apply speech recognition in fields like medicine, manufacturing, telecommunication and automotive industries [2] [5], [10], [12], [14][30][31]. Current speech recognition performance does not reach the desired performance that many films and futuristic documentaries depict despite ongoing research efforts.

Research efforts have yielded different approaches that improve speech recognition performance. These approaches include speaker adaptive training, speaker normalization, gender separation, and speaker dependence and speaker independence. Each approach has its own contributing improvement towards speech recognition.

With these improvements, speech recognition has recently become important in Interactive Voice Response (IVR) systems. For over a decade, Dual Tone Multi Frequency (DTMF) has been used in IVR systems [98]. The goal of speech recognition is to allow callers to obtain information by speaking naturally to the systems instead of interacting with call menus using DTMF. Early speech recognition approaches were targeted at a specific user. However, this approach is not feasible when unknown speakers need to be recognised like in [30]-[33]. Today's speech recognition approaches are targeted at recognising unknown speakers. This modern approach is therefore more suited towards IVR system implementations for the public. However, the modern approach does not yield as good a performance compared to the early speech recognition approach. This study is concerned with evaluating the suitability of speaker adaptation methods for improving the performance of the modern speech recognition strategy using minimal time.

1.1 An overview of speech recognition

People frequently find themselves having to divert their attention from a certain task in order to control another task. In some cases, the complexity of the tasks can endanger the person's life. For example, if a motorist wants to dial a phone number while driving, the motorist could risk an accident. Another example the motorist still is driving and wants to find information about their flight departure at the airport. In both cases, the motorist cannot focus entirely on driving. In fact, the motorist could spend time retrieving information risking being involved in an accident. It is easier for the motorist to concurrently speak and drive. Therefore speech recognition can be used for dialling, reducing accident risk.

Human beings often conduct complex tasks, which require a high level of attention. To simplify these tasks speech recognition can be applied [70], [71]. It is therefore possible for people to simplify these tasks by using speech. People can now retrieve information, issue commands, conduct daily tasks and other human functions concurrently.

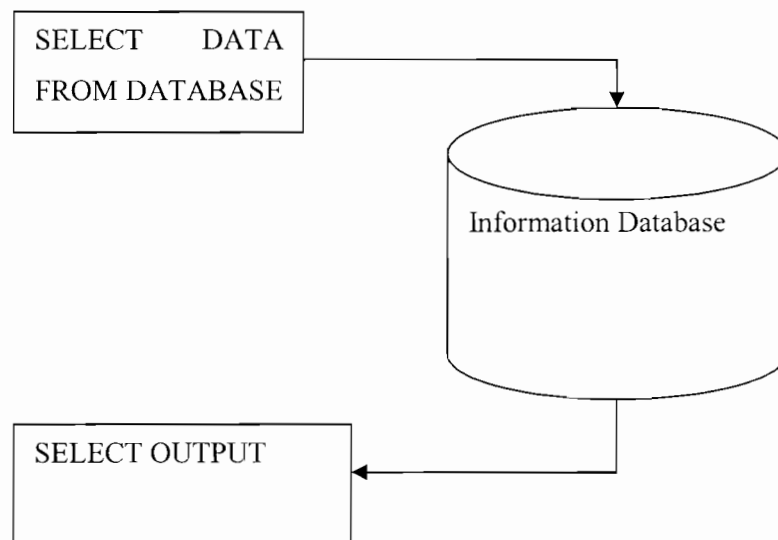


Figure 1-1: An example of an SQL database

In speech recognition, the aim is to perfectly convert speech into text and to use the generated text to achieve a task. For example, speech recognition can be used for retrieving information [15]. In such a case the generated text is then used as a search criteria in conjunction with a declarative language like SQL [73] to retrieve information from an SQL relational database like [74][75] (also see **Error! Reference source not found.**). In Figure 1-1, the SQL statement is used to select DATA from the database. This DATA is derived from the speech recognition output and is used in the SQL statement. Once this information is retrieved, it can then be relayed back to the requester using speech synthesis like in [24]. Another example is when using speech recognition for phone dialling. The generated text is used for a dialling application in order to dial a telephone number. In a similar way, the generated text from input speech is used in the application components for specific tasks.

Applications that use speech recognition can be divided into two components. There is a speech recognition component and there is the specific application component. This application component depends on the specific task.

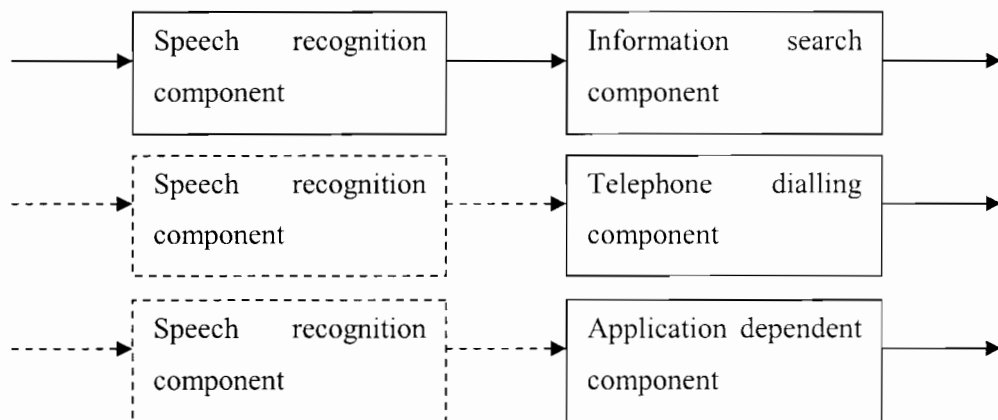


Figure 1-2: A generalization of speech recognition applications

1.2 How speech recognition works

Basic speech recognition functionality has been depicted in films like *Knight Rider* and *A Space Odyssey* [21], [27]. In such films, man speaks to machine, where the machine recognises the speech. The machine then responds to the man accordingly.

As with pattern recognition problems, speech recognition is about matching speech patterns to a reference pattern. Pattern recognition is regarded as “*the study of how computers can observe the environment, learn to distinguish patterns of interest from their background, and make sound and reasonable decisions about the categories of the patterns*” [78]. From [78] there are three key areas in pattern recognition. These areas are data acquisition and pre-processing (both front-end), data modelling and lastly classification (back-end). At any time of operation, the pattern recognition system (back-end) is in either training or testing mode. In training, the system “learns” the different categories of patterns while in testing it distinguishes between patterns, given the “learnt” material [79]. Speech recognition is similar to pattern matching.

A speech recognition model is depicted in Figure 1-3 with respect to the above working definition. The model shows the training and testing configurations. In the training configuration data acquisition and pre-processing occur, and then data modelling (in this case speech modelling) follows. In the testing configuration, data acquisition and pre-processing occur then classification (in this case *recognition*) follows. It’s important to note that there is a common front-end where *feature extraction* occurs and a back-end where *speech models* and *classification* mechanisms are implemented. Chapter 2 provides details.

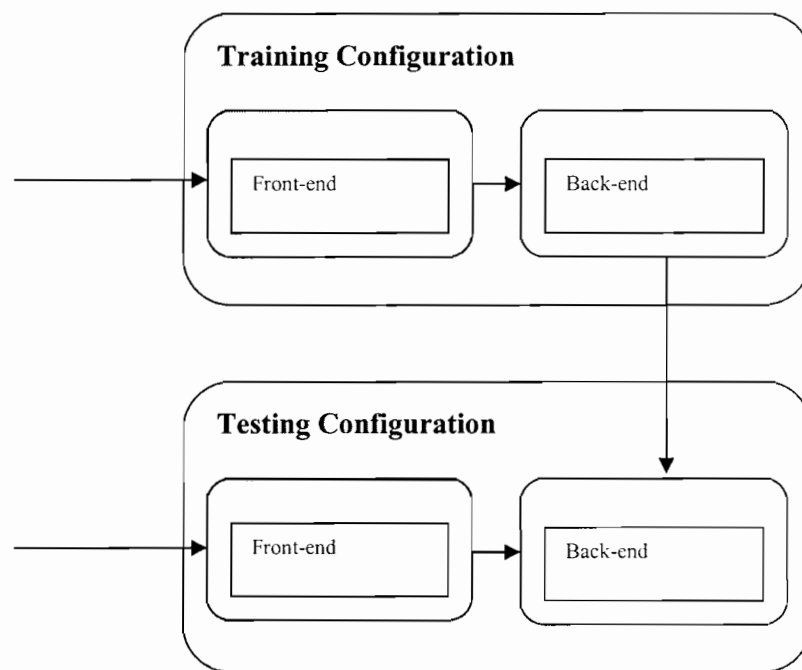


Figure 1-3: Fundamental speech recognition model

There are several different considerations, that need to be made prior to implementing the model [1][3], [4]. It is important to know the location, environment, equipment and other performance enhancing factors with regards to model performance. With thorough considerations, an optimized model can be implemented. For example in this study, since the speech recognition model shall be used to recognize unknown speakers it is then trained with many speakers as opposed to training with one speaker. But prior to expanding and illustrating this point in a later chapter, it's important to grasp the concept of a speaker dependent model and speaker independent model.

If a speech recognition system is trained on one person, this system is called *speaker dependent* (SD). This system is essentially an SD model. However, if a system is trained on many speakers, it is called a *speaker independent* (SI) system. This system is essentially an SD model. The former system is suitable for the speaker it is trained on as it ideally models that speaker. It therefore performs very well with that one speaker but does not perform well on a different speaker. However, the latter system is ideal for situations where many speakers will use the system. Generally it is not as suited for a particular speaker as is the SD system because it does not ideally model one speaker. Furthermore, the SD system requires a large amount of training data in order for it to work well. However, since the SI system generalizes the speakers, it only needs samples of the speaker population in order to train it well. This design consideration plays a pivotal role in this study and is important in the design planning, as it shall be clear later in this study. In the next sub-section the public access IVR server is introduced.

Figure 1-4, shows the key milestones in speech recognition development. Novel products initially are accompanied by hype, where the public sees them as “nice-to-haves”. However later in time, such products proved their usefulness and became essential to have. Speech recognition products continue to prove commercially viable as they prove their usefulness in people’s lives [8].

Late 1950s: Speech recognition research begins
1964: IBM demonstrates <i>Shoebox</i> for spoken digits at the New York World's Fair
1968: The HAI, 9000 computer in 2001 — <i>A Space Odyssey</i> introduces the world to speech recognition
1978: Texas Instruments introduces the first single-chip speech synthesizer and the <i>Speak and Spell</i> toy
1993: IBM launches the first commercial speech recognition product, the <i>IBM Personal Dictation System</i> for OS/2
1993: Apple ships <i>PlanTalk</i> , a series of speech recognition and speech synthesis extensions for the Macintosh
1994: DragonSystems' <i>DragonDictate for Windows 1.0</i> is the first software-only, PC-based dictation product
1996: IBM introduces <i>ModSpeak/Radiology</i> , the first real-time, continuous speech recognition product
1996: OS/2 Warp 4 becomes the first operating system to include built-in speech navigation and recognition
June 1997: DragonSystems ships <i>NaturallySpeaking</i> , the first general-purpose, continuous speech recognition product
August 1997: IBM ships <i>ViaVoice</i>
Fall 1997: Microsoft CEO Bill Gates identifies speech recognition as a key technological advance

Figure 1-4: Speech recognition research milestones [20]

1.3 Public access IVR server

An IVR server is a telephone-based application. It can provide information to users by allowing them to select from a series of menus [15]. Such a system is ideal for several situations where 24-hour information access is required, better customer service is required, and also where productivity is to be improved. Speech recognition is seen to be a more progressive way of information interaction [82], [83], [87] for these systems and has many benefits than the traditional DTMF.

Table 1-1: A comparison between telephones and computers in South Africa on average per province as in 2005 [18]

Household goods	Numbers in South Africa
Computers	968 741
Telephones	2 743 836
Cellular Telephones	3 615 241

As seen from Table 1-1, there are more telephones (fixed and cellular) than there are computers per household in South Africa [18]. Thus more people have access to telephones than they have to computers. With telephones, more people will have access to information. The public access IVR server is intended to give people access to information.

The server needs to meet several essential requirements prior to it being able to successfully deliver information. These requirements are such that the system adds value to the information provider as well as the public that is seeking information. Such requirements are as follows.

The public access IVR server should:

- provide information access for more people;
- provide information access to people anywhere in South Africa;
- have the ability to process many information requests from the public;
- be available 24 hours a day, 7 days a week.;
- be user friendly;
- provide accessibility in the language suitable to the information requester; and
- provide accurate and current information.

These requirements will yield a system with real benefits to the information provider as well as the public. Such benefits are:

- improved lives through country wide information access;
- minimized risk of Repeated Stress Injury incurred from keyboards;
- rapid information access facilitated by speech that is a natural means of human communication. There will be no need to learn how to retrieve information;
- no need to experience engaged telephone signals. It will be possible to service more telephone calls at any time instance;
- consistent user friendly information retrieval. The speech interface will be consistent and thus easy to use; and
- accurate information retrieval. It will be possible for the system to adapt to the caller.

The system should be able to offer real benefits to the public. The requirements placed on the system will ensure the benefits. These benefits will ensure the system is of value to the information provider and the public.

1.4 How does the public access IVR server work?

With the aid of a block diagram in Figure 1-5, this sub-section will look at how the public access IVR server works. In Figure 1-5 is a user with a telephone handset. This user could be sitting anywhere in the world, as the IVR server is not affected by location. As seen in Figure 1-2, Figure 1-5 has a speech recognition component and a voice response component (speech synthesis in this case). The user can retrieve information using a telephone and speech. The information is retrieved from a database similar to that depicted in Figure 1-1. The information is given to the user as speech similar to a voice response system like in [24].

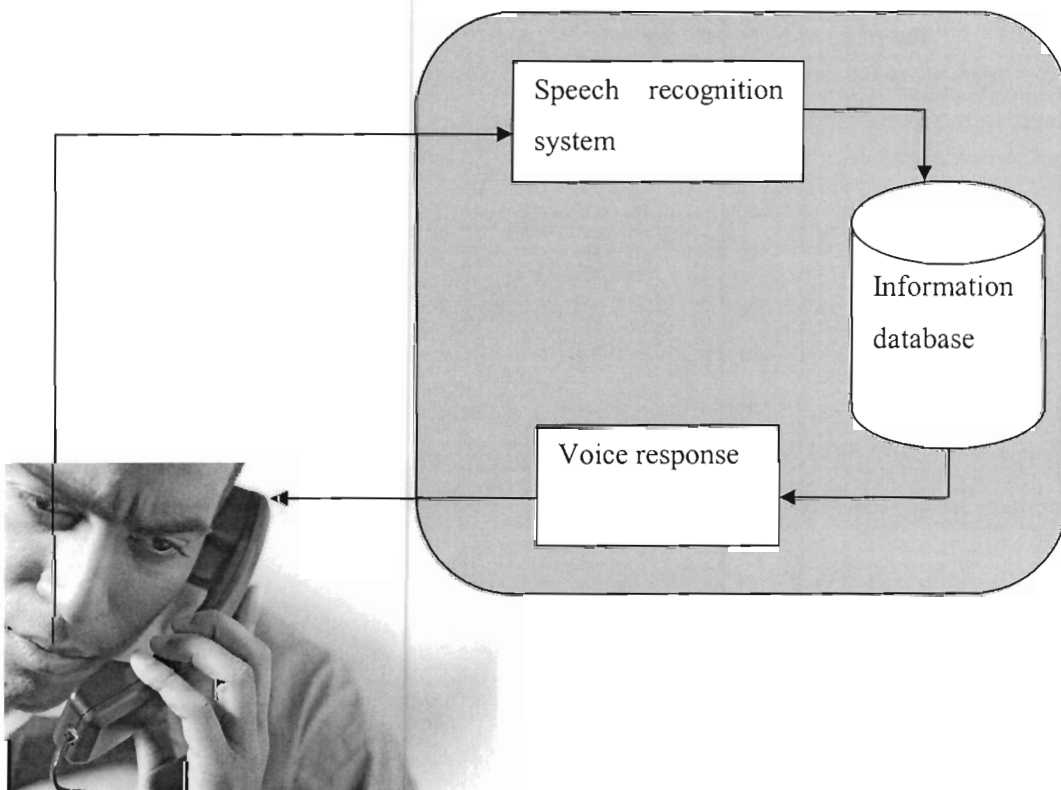


Figure 1-5: Public Access IVR server

A possible way of improving the system is via the way information is given to the server. Implementing speech recognition as a means of information input, has three distinct advantages as the IBM group [85] shows. Firstly more devices other than personal computers can be used since the server is linked to a communication network. Therefore, this can contribute towards solving the digital divide. Secondly, since speech is used, the system will naturally be user-friendly. Thirdly, the system will also provide service beyond human capabilities.

An SI public IVR server will be appropriate since it will be used by unknown speakers [1],[3], [7]. This study evaluates the suitability of speaker adaptation for improving the performance of an SI IVR server.

1.5 Objective of the study

The public access IVR server will offer access to information. It will serve a population of at least 40 million people. To implement SD systems for each individual person in the population would require many million hours of data. This is equivalent to many thousands of years worth of speech data to be collected. It is therefore not feasible to design the server as an SD system. However, in an SI design, several thousand people can be used in order to generalize for the population. This would require several thousand hours of data. This is equivalent to about several days' worth of speech data. It is therefore feasible to implement an SI server.

In implementing an SI system, SD-like performance can be achieved with speaker adaptation. This study evaluates the suitability of speaker adaptation concerning the public access IVR server. The aim is to deliver a speech recognition application that offers improved performance for the public with all the benefits of an SI system.

1.6 Speaker adaptation

In order to have a well-trained speech recognition system several hours of speech data are needed. In the case of the speaker dependent speech recognition system, a few hours of speech data are needed from the speaker the system is intended for. This system performs very well for the speaker it is intended. For the SI speech recognition system, a few hours of speech data per speaker are required. This amount of data increases in proportion of the number of speakers used for training. Speaker adaptation will be used to improve the performance of the SI systems to approach SD system performance.

Speaker adaptation works using various methods like model or feature transformation. Model transformation involves the transformation of the speech models which were described earlier in section 1.2 such that they suit the new speaker. Feature transformation transforms the features such that they match the features of the new speaker. Speaker adaptation is the adaptation of models based on speaker data, to suit the enrolling speaker (the new speaker providing the data).

The main aim of speaker adaptation is to produce a speech recognition system that better recognises the speech from a new speaker [3]. The main benefit from speaker adaptation is the reduction in the amount of data required to get performance gain, when compared to the amount of training data used. This is very vital for the public access server as the speakers and designers of the system wouldn't want to spend too much money in adapting the system than retrieving information. Generally the less data required the more desirable speaker adaptation is. Speaker adaptation is described in greater detail in Chapter 4. However, for the objectives of this study, the interest is in the suitability of speaker adaptation for the public access IVR server.

1.7 Scope and limitations

As motivated before, the public access IVR server is best implemented as an SI speech recognition system. An SD speech recognition system would perform better than SI speech recognition systems for the speaker it was intended. It is not feasible to implement an SD speech recognition system for each person in the South African population because it will require an enormous amount of data. This was found to be many years worth of speech data. While an SI speech recognition system may not perform as well as the SD speech recognition system, it only requires about a hundred hours of data in total from all the speakers recruited for training the system [41]. Due to time and cost, it is therefore the only way to implement the IVR server as an SI speech recognition system rather than as an SD speech recognition system. Speaker adaptation shall be used to achieve improved performance for the IVR server.

The evaluation work shall be done using a custom database. This custom database shall be described in more detail in Chapter 5. The database provides a suitable medium with which the suitability of the algorithms can be evaluated.

1.8 The approach

The custom database used for this study allows for the evaluation of the suitability of speaker adaptation algorithms for the public access IVR server. Several speakers were recorded in order to build the database. From this database, several SD and SI systems are built. The SD systems are built for each speaker. The performance is then measured. The next step tests this system on a different speaker and this performance is measured. Then the system is adapted to the different

speaker and again this performance is measured. The main purpose of the SD systems is to verify the use of speaker adaptation techniques and to also have on record the expected performance of speaker adaptation algorithms. Then SI systems are built such that at least one speaker is not involved in the training process. The reason for this is to emulate the real world scenario where speech recognition systems are used by people who were not involved in the training process. The system is tested with a speaker who is not involved in the training. This performance compared with the expected performance of a new SI system. The SI system is then adapted to the new speaker and this performance is compared with [3][7]. The overall results of both SI and SD systems are then compared in relation to each other and the results are analysed.

1.9 Structure of thesis

This thesis is organised as follows:

This study investigates the use of speaker adaptation for a public access IVR server. Therefore the fundamental area of focus is speech recognition. Chapter 2 will describe an in-depth technical view of speech recognition where the fundamentals of speech recognition will be described. A generalized model of speech recognition will be given and this model will form the basis of the discussion for the rest of the chapter. Various aspects of the model like feature extraction and modelling will be discussed. More focus will be placed on techniques that are more commonly deployed in state of the art systems as the evaluations utilize state of the art techniques.

Chapter 3 will describe the tools used for the evaluation. In describing the tools, a general process on how to conduct the evaluation is given. The techniques, which are used for data preparation, will also be described. Common training techniques will be described where emphasis is placed on the determination of training termination. Again this thesis focuses on speech recognition, and so this chapter will also present how speech recognition is done. Common performance measurement techniques, which are commonly used to measure recognition performance will be presented.

Chapter 4 will describe the algorithms used for this study. This chapter will describe MLLR, MAP and MLLR+MAP. MLLR is a transformation-based algorithm, while MAP is an algorithm that adapts in a more detailed manner. MLLR+MAP is a combination of the two algorithms

where, MLLR is applied first then MAP is applied. The details in which the algorithms work will be described. In this description, the suitability of the algorithms in certain situations will also be analysed. Lastly a comparison of the main algorithms is given.

Chapter 5 presents the results that shall be achieved by this study. In this presentation, first the results to be achieved with the SD systems shall be presented. Then the results to be achieved with the SI systems will be presented. The various settings of the different algorithms will also be presented. The results where the Levenshtein is used shall be presented. Finally Chapter 6 presents the conclusions and recommendations.

1.10 Summary

This chapter has given an introduction to this study. Firstly, the idea of speech being a primary communication tool was presented. Over the years speech recognition has presented numerous applications. While speech recognition was initially an isolated word recognition exercise, modern day systems recognise continuous speech. As a result, applications like Interactive Voice Response (IVR) Systems have become more common in modern day. While this application can be implemented as a speaker dependent (SD) system, commercial public applications are feasible only as speaker independent (SI) systems. Due to the fact that SI systems are better suited for recognising unknown speakers, such applications are designed as SI systems. However, SI systems do not offer good performance for a particular speaker, compared to when that speaker uses an SD system. So speaker adaptation is used to change the system. The changed system better suits the new speaker. This study is concerned with the ability and rate at which the system is changed to better suit the new speaker. It generally follows an approach which was described earlier. Most importantly, this study is interested in the suitability of speaker adaptation for the public access IVR server.

Chapter 2

2 Speech Recognition Fundamentals

The fundamental model of the speech recognition system remains unchanged, despite the improvements in speech recognition. This model has two major components namely, the front-end and back-end. The front-end performs feature extraction as shown in Figure 2-1, while the back-end performs the decision-making. The right end of Figure 2-1 shows the final output, which is a hypothesis.

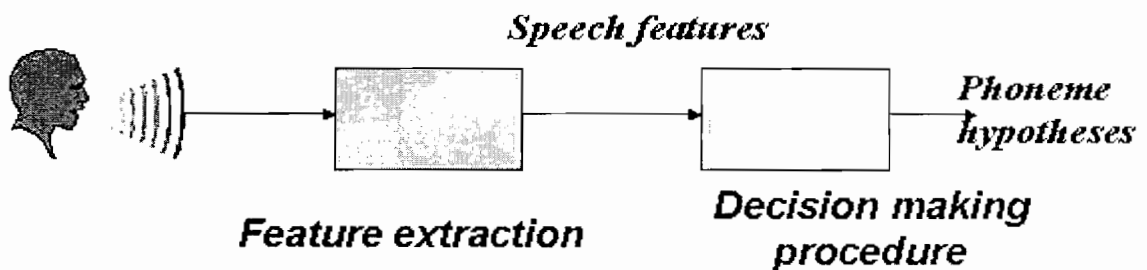


Figure 2-1: The fundamental model of a speech recogniser [102]

This chapter discusses the speech recognition fundamentals. This includes a description of the front-end and the back-end. It also describes the methods used for feature extraction. Finally, it will give a summary.

2.1 Feature Extraction: Front-end

Due to variations in speech, speech recognition on raw speech signal is not possible. The front-end must perform feature extraction on the signal [1]. In feature extraction, unique information about the signal is obtained. This information is used to generate the speech features (feature vectors). The speech features are used by the back-end for decision making.

Feature vectors are represented using various methods. Examples of such methods are short time energy, zero crossing rates and level crossing rates [1]. Short time energy is the most popular method. These methods form a basis for many common feature vectors [45][48].

Feature vectors are produced in three steps. The first is analogue-to-digital conversion, the second is pre-processing and the third is feature extraction. Figure 2-2 shows the overall process of the front end. The top part of Figure 2-2 shows the original signal. The bottom part of Figure 2-2 shows the result, the feature vectors (parametric speech vectors). The next sub-section describes this process.

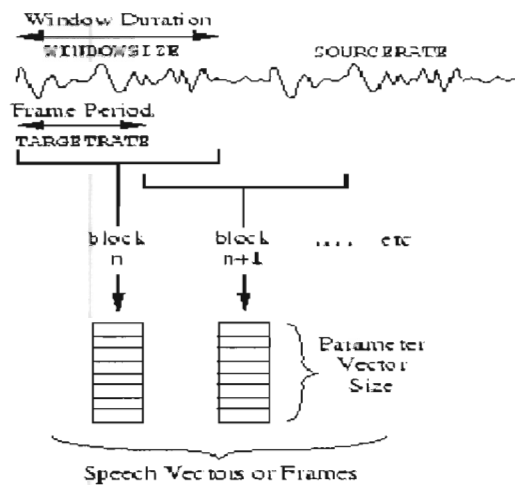


Figure 2-2: Conversion of speech into parametric speech vectors [48]

2.1.1 Analogue-to-digital Conversion

Speech recognition systems can only manipulate digital signals. Therefore analogue signals must be converted into digital signals. The converter is called an Analogue-to-Digital converter (ADC). An ADC is functionally separate from speech recognition systems. However, it is usually included in a speech recognition system. Therefore speech recognition systems include the ADC's function.

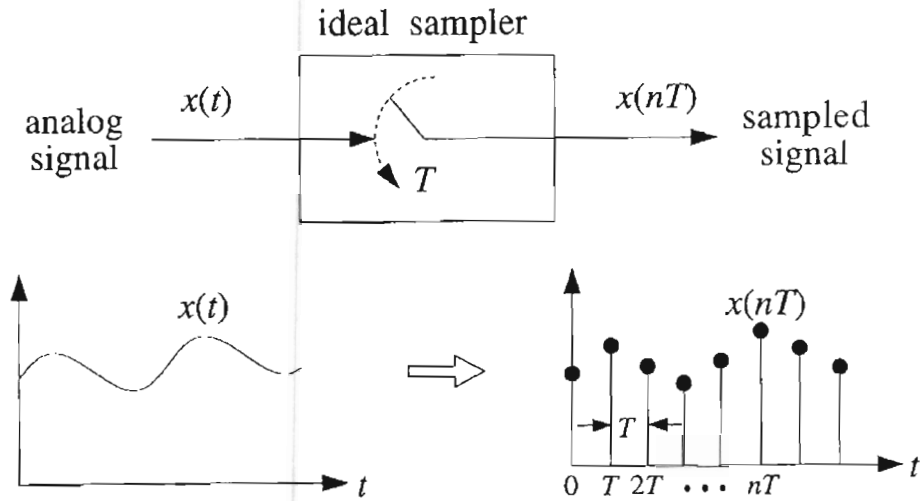


Figure 2-3: Sampling of a speech signal

Analogue-to-digital conversion is achieved in three steps. Firstly, the signal is sampled as seen in Figure 2-3 above. In this step, the analogue signal is represented in a discrete format. Redundant information is omitted while important information is retained. This is ensured by applying the Nyquist theorem during conversion. As seen in Figure 2-3, $x(t)$ is the continuous analogue signal which is then sampled. The resultant signal $x(nT)$ is as a discrete signal on the right in Figure 2-3. Analogue-to-digital conversion is used in many fields e.g. telecommunications, instrumentation, digital audio, and speech signal processing.

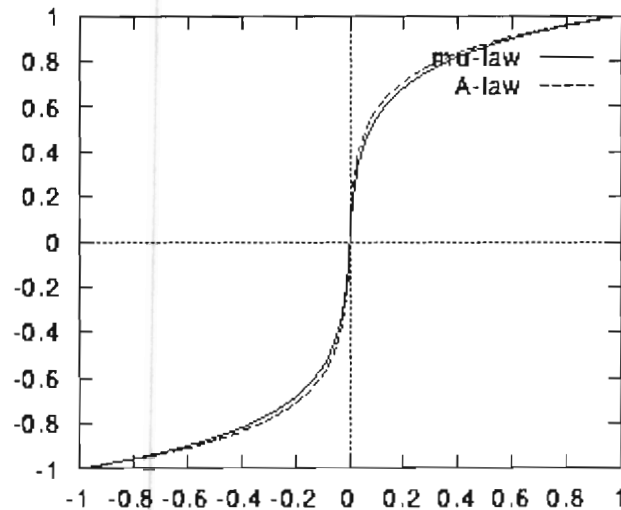


Figure 2-4: Quantization [101]

Secondly, the sampled speech is quantized. This process means that the samples are given a value in this new sample domain and an example of this is shown in Figure 2-4. This means that if a certain speech value in the analogue domain represented 6 volts in the range of 15 volts, then in quantization a new value is allocated within the discrete finite range or quanta within the digital domain that represents that range of 15 volts. In the figure above, the range is mapped from -1 to 1 on the vertical axis. However, due to the finite nature of quantization, samples may introduce phenomena like quantization noise.

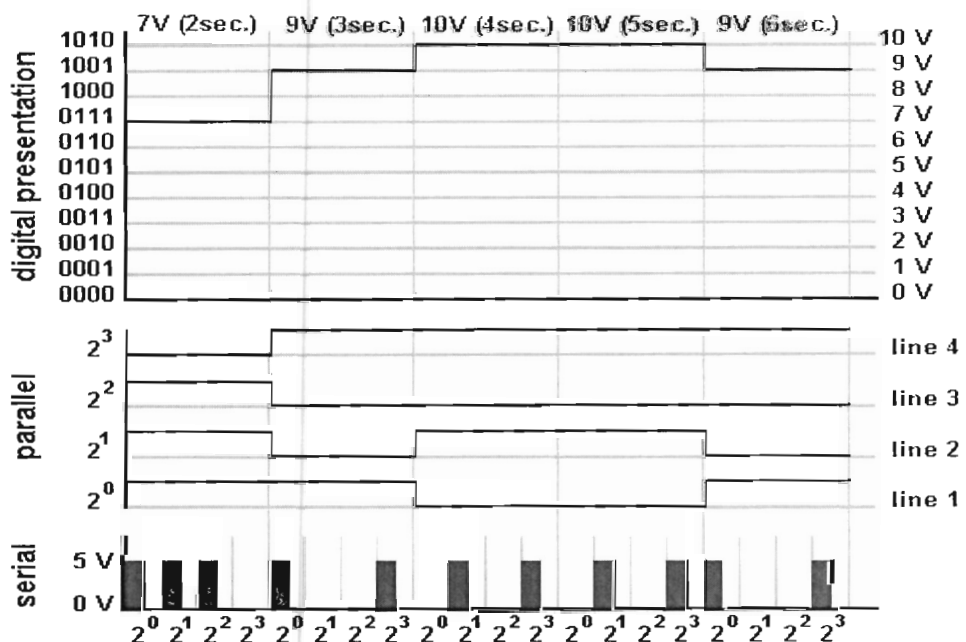


Figure 2-5: Coding example

Lastly, the signal is coded. This is the assignment of unique binary numbers for each quantization level. As seen above, each level within the voltage level is assigned a digital representation according to the vertical axis. This digital representation is a better form which machines can manipulate.

2.1.2 Pre-processing

Speech pre-processing happens after Analogue-to-digital Conversion. This process is done on the digital signals. It captures important speech information for producing feature vectors. The information contributes toward speech recognition performance.

Pre-emphasis is applied in order to emphasize high frequencies in a speech signal. This is achieved with a first order high pass filter that increases the relative energy of the high frequencies. The spectral slope of the voiced sections is tilted to the right so that high frequencies are emphasized [54]. Once the pre-emphasis filter is applied to these frequencies, the frequency energies are then boosted. The transfer function of such a filter is specified as follows:

$$H(z) = 1 + az^{-1} \quad (2-1)$$

where a is a constant in the range of -1 to -0.4 [54][68]. In some cases, pre-emphasis is represented as shown in Figure 2-6 [48].

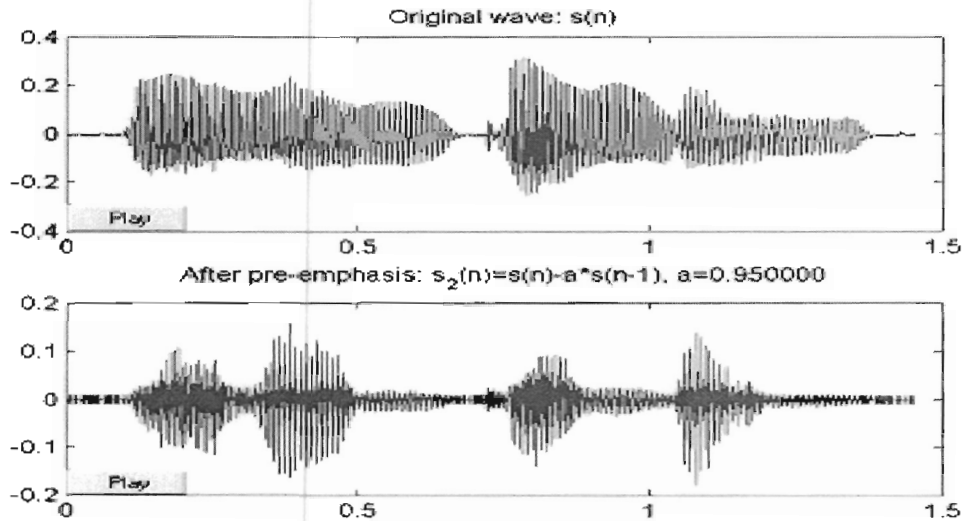


Figure 2-6: Pre-emphasis of a speech waveform

The tapering of the speech samples in each window is beneficial in order to rid of any discontinuities at the edges of the windows. This is achieved by applying the following transformation:

$$w_n = \begin{cases} 0.54 - 0.46 \cos\left(\frac{2\pi n}{N}\right) & 0 \leq n \leq N \\ 0 & \text{otherwise} \end{cases} \quad (2-2)$$

to each sample in the window, where the number of samples in the window is N , n is the sample and w_n is the transformed output sample n .

Generally windowing is applied, by using the window transformation using equation 2-2 above. Once this windowing is achieved, pre-emphasis is then achieved within that windowing although the order does not matter [68]. Once the pre-processing has been employed, further spectral analysis is done. As it is known, no two utterances from one speaker are identical. Therefore it is important to use feature generation in order to capture information from speech. This is described next.

2.1.3 Feature extraction in the front end

Once speech pre-processing is completed, the speech information is converted into feature vectors. This conversion process is categorised into 3 broad classes, namely, Linear Predictive Coding, Filter Bank based analysis and also Auditory based feature extraction methods. This subsection shall describe these classes.

2.1.3.1 Linear Prediction Coding

Linear Prediction Coding (LPC) is a process that models the physiological characteristics of a speaker [54]. It results in Linear Prediction Coefficients being derived from the human vocal tract model. Speech samples are expressed as a linear combination of the previous speech samples in the acquired speech as shown by equation 2-3.

$$\hat{s}(n) = \sum_{k=1}^p a_k s(n-k) + Gu(n) \quad (2-3)$$

The LPC model order is denoted by p . It shows the number of previous samples used to model the n -th sample. G is a gain scaling factor, while $u(n)$ is the present input and a_k are coefficients of the linear combination (known as linear prediction coefficients) [54]. The present input is usually ignored in speech applications.

There is a difference between the actual signal and its LPC-model. This difference is expressed as follows:

$$e(n) = s(n) - \hat{s}(n) \quad (2-4)$$

where $\hat{s}(n)$ is the LPC model and $s(n)$ is the actual signal. This difference is also known as the *prediction error* and is denoted as $e(n)$. If the z-transformation of equation 2-4 is taken

$$H(z) = \frac{1}{1 - \sum_{k=1}^p a_k z^{-k}} = \frac{1}{A(z)} \quad (2-5)$$

$A(z)$ is the p -th order inverse filter. The mean square prediction error (MSE) [54], $E = \sum e^2$ is minimized in order to optimize LPC analysis. Minimization is then done using the autocorrelation method as well as the Durbin recursive algorithm in order to produce the best predictor coefficients. Ultimately, the LPC coefficients are then determined using equation 2-6.

$$c_n = a_n + \sum_{k=1}^{n-1} \frac{k}{n} c_k a_{n-k} \quad (2-6)$$

2.1.3.2 Filter-bank analysis

The human way of hearing uses a non-linear method of processing the frequency spectrum. Therefore, Filter-bank analysis emulates the human way of processing the frequency spectrum. Evidence shows that this method performs well in speech recognition [1], [48]. Filter-bank analysis is achieved by placing band-pass filters from zero to the Nyquist frequency in the frequency scale.

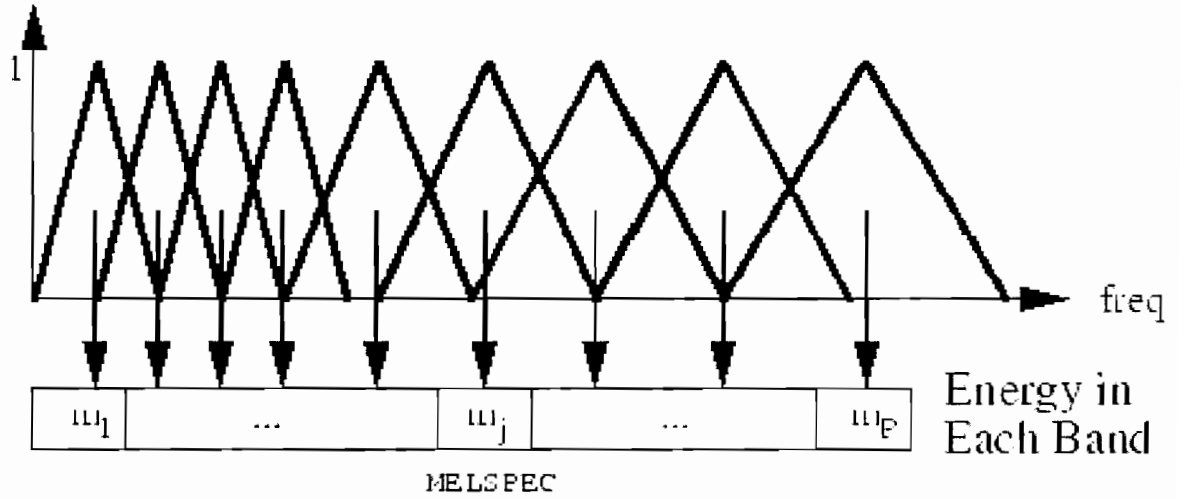


Figure 2-7: Filter-bank analysis [48]

The band-pass filters are placed non-linearly in the frequency spectrum as seen in Figure 2-7. The Erb, Bark and Mel scales are used to determine the width, centre frequency and spacing of the filters. Therefore, important information can then be extracted using the filter bank approach.

The most popular filter bank based feature extraction process is Mel-Frequency Cepstral Coefficients (MFCCs). This process uses triangular filters, spaced according to the Mel scale. For frequencies below 1000Hz, the triangular filters are spaced linearly. For frequencies greater than 1000 Hz, the filters are spaced logarithmically. This is achieved in three steps in conjunction with equation 2-7.

$$Mel(f) = 2595 \log_{10} \left(1 + \frac{f}{700} \right) \quad (2-7)$$

Firstly, the speech samples in the analysis window are transformed using the Fast Fourier Transform (FFT) and the magnitude of the FFT taken. Secondly, the magnitude coefficients of the FFT are then multiplied by their corresponding filter gains. The result is then accumulated for further scaling computations. A weighted sum that represents the spectral magnitude of that filter bank channel is achieved. This is done by utilizing the accumulated result to scale the spectral magnitude for the filters. Thirdly, the filter bank coefficients are yielded. The third step involves the conversion of the filter bank coefficients into the required cepstral coefficients using equation 2-8. It may be desirable to remove unwanted frequencies or frequencies where there is no useful information [48]. Therefore, a pass-band within the filter-bank can be implemented by specifying which frequencies are desired for capture. Thus, the cepstral coefficients are attained as follows:

$$c_i = \sqrt{\frac{2}{N}} m_j \cos \left(\frac{\pi i}{N} (j - 0.5) \right) \quad (2-8)$$

where N is the number of filter bank channels and m_j are the output coefficients of the filter banks. This equation is also known as the Discrete Cosine Transform (DCT) and is also represented as follows:

$$c_i = \sqrt{\frac{2}{N}} \sum_{j=1}^N \log(m_j) \cos \left(\frac{\pi i}{N} (j - 0.5) \right) \quad (2-9)$$

again, where N is the number of filter-bank channels and m_j are the output coefficients of the filter bank as in the previous formula. Due to the correlated nature of the filter bank coefficients, it is mandatory to transform the filter bank outputs using equation 2-9 prior to using the output for HMM speech recognition [1],[48][54].

2.1.3.3 Auditory Modelling feature extraction

Auditory Modelling feature extraction improves speech recognition performance in noisy environments by modelling the human auditory system [54]. Since humans can discriminate between noise and speech very well, research has explored auditory modelling feature extraction. The main motivation is that if human hearing can be modelled, then the advantages of such discrimination can also be emulated in the models. However, this modelling is not capable of

modelling the human interpretation of auditory signals by the brain. This method differs in philosophy from Filter Bank Analysis in that it has noise discrimination as the benefit.

A popular auditory modelling feature extraction technique is Perceptual Linear Prediction (PLP) [66]. In principle, an auditory spectrum is computed and from there, the LPC model is then computed [54]. To compute the auditory spectrum, the power spectrum of a given speech segment must be convolved with a critical band-masking pattern. This auditory spectrum (critical band spectrum) is then pre-emphasized using a simulated equal-loudness curve and then compressed using a cubic non-linearity. Finally the resultant auditory spectrum is then an LPC model can be computed. The LPC coefficients can be taken as the extracted features, where the Cepstral coefficient can be computed as sometimes done in practise.

It is possible to combine the Mel-Filterbank, with PLP in order to achieve more robust features. This method is called Mel-Filterbank PLP (MF-PLP). Basically, Mel-Filterbank coefficients are computed from the power spectrum of a speech frame. These coefficients are then weighted using an equal-loudness curve and then compressed by taking their cube root [54]. The final result is an auditory spectrum, and an LPC model may then be generated from this. MF-PLP differs from PLP in that MF-PLP computes is features from Mel-Filterbank coefficients rather than the power spectrum as seen in PLP.

Another version of PLP like RelAtive SpecTrAl (RASTA) PLP exists [54],[66],[67]. Since this study is concerned with speaker adaptation, this version is not described here.

2.1.3.4 Other feature vector components

This sub-section discusses additional components in feature vectors. In order to obtain robust speech recognition, it is necessary to consider additional components from the feature vectors. These components help model additional information with regards to the observed speech.

The other additional speech vector components are appended to feature vectors to make the recognition process robust. These components capture more information about the speech and ultimately improve the robustness of the speech recognition system performance. The components

encapsulate energy and time derivatives and are commonly included in speech recognition systems [3], [4][7][10], [49].

The energy derivative is included by computing over a given window of N speech samples long, as follows:

$$E = \log \sum_{n=1}^N (w_n s_n)^2 \quad (2-10)$$

where w_n is the window function and s_n is the speech sample. It is also possible to compute the energy component without the window function [48], thus simulating the human ear response. This results in equation 2-10 being formulated as shown below in equation 2-11.

$$E = \log \sum_{n=1}^N (s_n)^2 \quad (2-11)$$

The time derivatives can be represented using different equations. Some of the equations that are used to derive time derivatives from the speech signals are:

$$s'(n) \approx s(n) - s(n-1) \quad (2-12)$$

$$\frac{d}{dt} s(n) \approx s(n+1) - s(n) \quad (2-13)$$

$$\frac{d}{dt} s(n) \approx \sum_{m=-N_d}^{N_d} m s(n+m) \quad (2-14)$$

The first two time derivative representations are commonly known as the backward and forward differences respectively [69]. Equation 2-14 is known as the regression analysis equation as it represents an approximation to the ideal differentiator [68]. As a result, second order time derivatives and other higher order derivatives are obtained by reapplying the regression analysis equation. More commonly, the first time derivative is also known as **delta**. The second order time derivative is known as **delta-delta**, or acceleration etc.

2.2 Speech modelling

Various methods of achieving speech recognition exist. Artificial Neural Networks (ANN) and Hidden Markov Models (HMM's) are examples of the various methods that are used [23]. In order for the fundamental speech recogniser to recognise speech, it uses one of the various methods to do the pattern recognition required for speech recognition.

With ANNs, the basic idea is to model the human nervous system as a set of models. In speech recognition the set of models can be trained with training data and then used to classify test data. In training, the set of models are told what the data is and then in testing the models are then used for classification of unseen data. Time Delay Neural Networks (TDNN) and Self Organizing Maps (SOMS) are examples of some of the types of ANNs that are used in speech recognition [54]. ANNs generally have three problems which affect their performance. Firstly, they have an inability to handle time warping, thus having lower performance when systems involve large time spans [54]. Secondly, they are less developed than HMM's when it comes to handling speaker adaptation. Thirdly, they are less capable of handling extraneous noise [54]-[56].

HMM's are a popular choice for designing speech recognition systems as they give state of the art performance [1]. The underlying idea behind HMM's is that speech can be modelled as a parametric random process and that the parameters of the process can be determined in a reliable scientific manner [1]. The basic theory of how HMM's worked was first published in the late 1960's by Baum and colleagues. The first implementation was done by Baker [62] and Jelinek [63]. Since then, many implementations of HMM-based speech recognition systems have been seen on the market. One such implementation is the Mercedes Benz speech recognition system implemented in their premium vehicles [22]. HMM's are the choice classifiers for this study as they address the key area of speaker adaptation better than ANN's.

The speech recognition systems built in this study are based on HMM's. HMM theory is therefore discussed in detail in the next sub-section. The subsequent sub-sections shall explain how HMM's are defined and solved.

2.2.1 Defining HMM's

HMM's started from Markov chains where the chain is defined with a set of N distinct states $S = \{s_1, s_2, \dots, s_N\}$ [1]. State transitions or the move from one state to the other will be associated with a certain time t , and will have a probabilistic description a_{ij} of moving from itself to another state or to itself. The state transition is only permissible where links within states are defined (where state transitions are defined). This probabilistic description is also known as a transition probability. The actual state that the system is in at time t is denoted as q_t . The subscript i indicates the current state and subscript j indicates the target state (the next state). The transition probabilities follow the two basic probability constraints where:

$$\sum_{j \in S} a_{ij} = 1 \quad \forall i \in S \quad (2-15)$$

and

$$a_{ij} \geq 0 \quad (2-16)$$

as a constraint that the probability of transition is either greater zero if there is a transition and zero if there is no transition.

With Markov chains, the output at each time t has a corresponding observation. For example, a Markov chain has 3 states ($N=3$). State 1 represents a sunny day. State 2 represents an overcast day and state 3 represents a rainy day. If a rainy day is observed, then the observation is from state 3. In this case, Markov chain is **observable**.

In defining HMM's, a set of states, S , where each state has a probabilistic output function must be present. The output is **not observable** and this means that a particular observation cannot be attached to a particular state. There is a probabilistic chance that an output came from a certain

state. The definition for HMM's maintains the two basic probability constraints described by equations 2-15 and 2-16. There are N states in the HMM. The set of M possible observations observable at each state is represented by the set $V = \{V_1, V_2, \dots, V_M\}$. The probabilistic description of moving from one state to the other is a_{ij} . There is the initial state distribution that is defined as π . HMMs have an output probability B . This defines the probability of observing a particular symbol out of the set of M possible observations.

A more convenient method of defining an HMM is as follows:

$$\lambda = \{A, B, \pi\} \quad (2-17)$$

which gives the complete set of parameters for HMM's. A represents the state transition matrix and is defined as:

$$A = a_{ij}, \quad 1 \leq i, j \leq N \quad (2-18)$$

B represents the output probability function, where:

$$B = b_j(k), \quad 1 \leq k \leq M \quad (2-19)$$

where M is the number of different symbols. The entries in B are defined as follows:

$$b_j(o_t) = \frac{1}{\sqrt{(2\pi)^n |\Sigma_j|}} e^{-\frac{1}{2}(o_t - \mu_j)^T \Sigma_j^{-1} (o_t - \mu_j)}$$

where k is interchangeable with o_t since they represent the same parameter. Further details about the output probability function are explained later in the chapter. π represents the initial state distribution and is:

$$\pi = \pi_i, \quad 1 \leq i \leq N \quad (2-20)$$

Through out this study, the word parameters will be used to indicate the parameter set λ (model λ) which collectively describes the HMM as shown in equation 2-17. The parameters play an important role in the probability measure for $P(O|\lambda)$. This probability measure states the probability of getting O given the parameters from the model λ , where O is the sequence of observations where $O = \{o_1, o_2, \dots, o_T\}$. o_i originates from the set V while T is the length of the sequence of observations. To use HMM's for speech recognition three problems must be solved [1], [48].

2.2.2 The Three HMM problems

The problems are as follows:

- **Problem 1:** Given the sequence of observations O and the model λ , how can $P(O|\lambda)$, the probability of having the given sequence of observations given the model be effectively computed?
- **Problem 2:** Given the sequence of observations O and a model λ , how can the corresponding state sequence $q = (q_1, q_2, \dots, q_T)$ be chosen that it best explains the observations?
- **Problem 3:** How can the model λ be adjusted so as to maximize $P(O|\lambda)$?

Problem 1 is about computing the probability that the observation sequence was generated by the model λ . Thus problem 1 is an evaluation problem and can also be viewed as a problem of scoring how well the given model λ matches the observation sequence [1]. Problem 2 is the process of revealing the hidden part of the HMM. The main idea is to find the “correct” sequence states that yielded the observation. It is important to note that the use of “correct” is rather loose and implies that there is no correct sequence of states that can be found. However, the purpose is to find the optimal sequence of states that best describes the sequence of observations. With problem 3, the attempt is to optimize the model λ so to best describe how a given observation arises. This problem is called training. It uses the training sequence [1] in order to optimize the model λ . This training sequence is used to adjust the parameters of the HMM.

2.2.3 Problem 1 solution: *Evaluation*

In solving problem 1, $P(O | \lambda)$, **all** the possible state sequences which are of length T (where T is the number of observations) can be listed –a method based on the law of total probability. Given that to evaluate problem 1 only a fixed-state sequence $q = (q_1, q_2, q_3, \dots, q_T)$ considered, equation 2-21 is used [1].

$$P(O | \lambda) = \sum_q \pi_{q_1} b_{q_1}(o_1) a_{q_1 q_2} b_{q_2}(o_2) \dots a_{q_{T-1} q_T} b_{q_T}(o_T) \quad (2-21)$$

It can be seen from the equation above as well as the definition of HMM's, that $2T$ calculations for each term in the sum of the above equation will be required. These calculations require $(2N-1)N^T$ multiplications and N^T-1 additions. For typical speech recognition applications it proves to yield a large amount of computations. To convince the reader an example is explored with an utterance from the custom database used in this study. For the HMM's in this study, N is mostly five for speech models. T can vary from about 800 to a very high number depending on the speech. In this case there will be in the order of $2*500*5^{800}$ computations and compared to the Forward-Backward algorithm, this is inefficient.

To overcome this limitation, a simple recursive algorithm is used to compute the probabilities. The commonly used algorithm is known as the Forward-Backward algorithm [1], [4],[48], [54]. A Forward algorithm computes a forward probability. A Backward algorithm computes a backward probability. First, the Forward algorithm is described and then the Backward algorithm is described.

The Forward Algorithm

Let the forward probability $\alpha_j(t)$ for the defined HMM model be as follows:

$$\alpha_j(t) = P(o_1, o_2, \dots, o_t, q_t = j | \lambda) \quad (2-22)$$

Thus $\alpha_j(t)$ is the joint probability of seeing the first t speech observations and being in state j when at time t given the model λ . The evaluation can be done in three main steps. Firstly, the initial condition is defined. Secondly, the recursive algorithm is used to compute intermediate steps until finally the last step, which is termination of the algorithm, is reached. The recursive algorithm is defined as follows:

$$\alpha_j(t) = \left[\sum_{i=2}^{N-1} a_i(t-1)a_{ij} \right] b_j(o_t) \quad (2-23)$$

The initial conditions are then defined as follows:

$$\alpha_1(1) = 1 \quad (2-24)$$

and

$$\alpha_j(1) = a_{1j}b_j(o_1) \quad 1 \leq j \leq N \quad (2-25)$$

The final condition of the algorithm is:

$$\alpha_N(T) = \sum_{i=2}^{N-1} a_i(T)a_{iN} \quad (2-26)$$

Finally, from the definition of $\alpha_j(t)$ it can be seen that

$$P(O|\lambda) = \alpha_N(T) \quad (2-27)$$

From the above equation it can be concluded that the evaluation problem is yielded by the computation of the forward probability. Note that with this method, many computations in the order of N^2T where $N(N+1)(T-1) + N$ are multiplications and $N(N-1)(T-1)$ are additions are needed. The evaluation problem is therefore solved efficiently by computing the forward probability.

The Backward Algorithm

The backward probability $\beta_j(t)$ is computed with the Backward algorithm, and is defined as follows:

$$\beta_j(t) = P(o_{t+1}, \dots, o_T \mid q_t = j, \lambda) \quad (2-28)$$

It is important to note that unlike the forward probability, which is a joint probability, the backward probability is a conditional probability. With the Backward algorithm, the aim is to calculate the probability of the partial observation sequence from $t+1$ to T given the condition of being in state j and given model λ . The backward probability can be computed recursively as follows:

$$\beta_i(t) = \sum_{j=2}^{N-1} a_{ij} b_j(o_{t+1}) \beta_j(t+1) \quad (2-29)$$

The initial condition is defined by:

$$\beta_i(T) = a_{iN} \quad 1 \leq i \leq N \quad (2-30)$$

and the final conditions given by:

$$\beta_i(1) = \sum_{j=2}^{N-1} a_{ij} b_j(o_1) \beta_j(1) \quad (2-31)$$

With the same efficiency of the forward algorithm, the backward algorithm can be computed. Once this is solved, the forward and backward probabilities are used to solve problem 3, which is described in elsewhere in this chapter. Note that:

$$P(O | \lambda) = \sum_{i=1}^N \alpha_i(i) \beta_i(i) \quad (2-32)$$

2.2.4 Problem 2 solution: *Revealing the state sequence*

In this problem, the state sequence that best describes the output needs to be revealed. Due to many possible best state sequences [1] a precise solution is not obtained. It means that there are many criteria that can be used to select the best state sequence. Therefore, for a problem that does not have precise answers, several solutions or selection specifications exist. One possible specification is to choose the states where each state on its own represents the solution path at each time t . This specification maximizes the expected number of correct states in the sequence. While this may maximize the expected number of correct states in the sequence, it may take into account invalid state sequences which may be caused by zero probability transitions in an HMM. The best specification that is in fact widely used is to find the single best state sequence that maximizes $P(q | O, \lambda)$ (which is the same as maximizing $P(q, O | \lambda)$). The formal name for the technique that is based on such a specification is called the Viterbi Algorithm [64]. Firstly the best score $\delta_t(i)$ along a single state sequence at time t , where the first t states have been seen and ending in state i is defined as follows:

$$\delta_t(i) = \max_{q_1, q_2, \dots, q_{t-1}} P[q_1 q_2 \dots q_{t-1}, q_t = i, o_1 o_2 \dots o_t | \lambda] \quad (2-33)$$

and this leads us to

$$\delta_t(j) = [\max_i \delta_{t-1}(i) a_{ij}] \bullet b_j(o_t) \quad (2-34)$$

To compute the states sequence, the above equation is computed recursively by first starting with the following initial condition:

$$\delta_1(j) = \pi_j b_j(o_1), \quad 1 \leq j \leq N \quad (2-35)$$

where an array $\psi_t(j)$ which is defined as:

$$\psi_t(j) = \arg \max_i [\delta_{t-1}(i) a_{ij}] \quad (2-36)$$

keeps track of the argument that maximizes equation 2-36 is initialised as:

$$\psi_1(j) = 0 \quad (2-37)$$

The recursion is then terminated by the terminating condition:

$$P = \max_{i \in S} [\delta_T(i)] \quad (2-38)$$

and the best state sequence is then described by:

$$q_t = \psi_{t+1}(q_{t+1}) \quad (2-39)$$

This recursion is used widely used in speech recognition and an alternate formulation of this Viterbi algorithm is used in recognising speech [48].

2.2.5 Problem 3 solution: *Training*

In this problem, the aim is to adjust the parameters of the model in subject such that it gives a higher probability of generating the observed sequence of data. Generally, the purpose is to maximize $P(O | \lambda)$. This is done by modifying the parameters of the model in order to satisfy some specification or criterion. This is difficult to do mainly due to the fact that the state transition sequence is hidden. While gradient techniques (where a gradient is descended in order to minimize error) exist for solving this problem [65], in this study a technique called Baum-Welch algorithm is described. This is used for this maximization [57]-[61] and was developed by L.E. Baum and his colleagues.

Consider $\gamma_t(i)$ to be a quantity which shows the number of times that the state i is visited when the quantity is incremented over time. Alternatively it can be said that this quantity amounts the expected number of transitions made from state i if the time index $t = T$ is disregarded. $\xi_t(i, j)$

can be summed up over t as well from $t = 1$ up until $t = T - 1$, and this is interpreted as the expected number of transitions from state i to state j . In a mathematically clearer way

$\sum_{t=1}^{T-1} \gamma_t(i)$ the expected number of transitions from state i having observed the sequence O and also

$\sum_{t=1}^{T-1} \xi_t(i, j)$ the expected number of transitions from state i to state j having observed the same

sequence O , then equations which can be used for the adjustment of the HMM parameters can be derived. The π parameter can be adjusted. The A and B parameters can be adjusted. To denote the adjusted parameters, $\hat{\pi}$, $\hat{A}$ and $\hat{B}$ are used respectively. In the order that the parameters are presented, the formulae are presented:

$$\hat{\pi}_j = \gamma_1(i) \quad (2-40)$$

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i, j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \quad (2-41)$$

$$\hat{b}_j(k) = \frac{\sum_{t=1}^T \gamma_t(j)}{\sum_{t=1}^T \gamma_t(j)} \quad (2-42)$$

For formula 2-37, $\hat{\pi}_j$ is the expected number of times that the model is in state i at the time $t = 1$. Then for formula 2-38, $\hat{a}_{ij}$ is equal to the expected number of transitions from state i to state j divided by the expected number of transitions from state i . Lastly, $\hat{b}_j(k)$ is equal to the expected number of times in state j and observing v_k (symbol) divided by the expected number of times in state j [1].

In order to maximize $P(O | \lambda)$, the basic training procedure is described below:

- I. Initialise the model λ (same as defined previously) by estimating the initial means and variances.
- II. Calculate $\xi_t(i, j)$, $\alpha_t(j)$ and $\beta_{t+1}(j)$ and $\gamma_t(i)$
- III. Estimate $\hat{\lambda}$ from $\xi_t(i, j)$ and $\gamma_t(i)$
- IV. Replace λ with $\hat{\lambda}$
- V. If Maximum Likelihood (ML) estimate hasn't converged, return to II and compute the steps again.

Mathematically speaking, in step I, λ can represent a point of inflection in $P(O|\lambda)$ or it can represent a starting point in $P(O|\lambda)$ where the ML estimate can be found [1]. This means that when λ represents a point of inflection, it is a local maxima such that $\hat{\lambda} = \lambda$. It is important to bear in mind that $P(O|\lambda)$ models an enormous amount of information that it has many local maxima. When λ represents a starting point where the ML estimate can be found, then $\hat{\lambda}$ is more likely to generate O than λ is. Another way of putting this is $P(O|\hat{\lambda}) > P(O|\lambda)$. These formulae and training procedures are used to improve the initial model that you start off with and better the parameters based on your training speech.

2.2.6 Continuous Density HMMS (CDHMM's)

Throughout the discussions, only discrete observations where the observations are defined as discrete symbols from a finite alphabet were considered. However, some applications in the world are often continuous signals or are vectors. Due to their nature, it is advantageous to model them as continuous signals or vectors [1], [2]. It is therefore of great benefit to use continuous output probability functions to model the continuous signals. Therefore the main difference between CDHMM's and the HMM's described in previous sections is that the output probability function which for CDHMM's is defined as follows:

$$b_j(x) = \sum_{k=1}^M c_{jk} \mathcal{N}(x, \mu_{jk}, U_{jk}) \quad (2-43)$$

where x is the observation that is being modelled, and c_{jk} is the output probability mixture coefficient for the k -th output probability mixture state j where the output probability mixture is finite. $\aleph$ can be some log-concave or elliptic symmetric probability function [1]. In this study the output probability function is the Gaussian function. μ_{jk} is the mean of the Gaussian function, while U_{jk} is the covariance matrix for the k -th output probability mixture.

The mixture coefficients must obey the probability constraints such that the output probability $b_j(x)$ can be normalized as follows:

$$\int_{-\infty}^{\infty} b_j(x) dx = 1 \quad (2-44)$$

and the constraints are:

$$\sum_{k=1}^M c_{jk} = 1, \quad 1 \leq j \leq N \quad (2-45)$$

$$c_{jk} \geq 0, \quad 1 \leq j \leq N, 1 \leq k \leq M \quad (2-46)$$

Therefore, in the speech recognition process, the feature extracted observation vectors x are then utilized in the formulae above in order to yield the final output in the recognition process.

2.3 Applications of speech recognition

Consider the accessibility to operating systems, using speech. Microsoft [30] has enabled their operating system to be operated using speech. It is possible to initiate functions like closing programs (terminating programs) using speech. The user simply utters the relevant words and the program will close as requested. This is useful if the user's hands are not available for typing. And to reciprocate the user's actions, the operating system can talk back. The second application is document typing. Microsoft, IBM and Dragon Systems have enabled the use of speech for

typing documents in word processing programs. The user can utter the sentences for the computer to type and the computer types the speech. The Microsoft, IBM and Dragon Systems are multi-purpose systems. This means that if a system is installed, the user may then use it in different programs like MS Excel, MS Word and all other windows applications.

An example of another application that is worth noting is the use of speech recognition at Ster Kinekor [72][103]. It is possible for Ster Kinekor clients to retrieve information about movie screenings using the telephone. The user can call in using any kind of telephone system and request for movie information. Generally there are many applications for speech recognition and more applications are being created every day.

Another different application is the telephone dialler as in Mercedes Benz cars (see Figure 2-8). The user of the application (usually the driver) can dial telephone numbers by using voice. The user can also conduct other vehicle functions which can be done without compromise vehicle safety. As can be seen in Figure 2-8 the user can keep driving while they use the voice to control certain tasks. The user can then safely see the output of the task in the display screen (just ahead of the gear lever). The input is processed via a microphone array as seen in Figure 2-9. The microphone array ensures that the speech is recognized despite all the in-car noise there may be. The microphone array is a design decision which is implemented after considering the environment, location and other mitigating consequences with regards to system performance.



Figure 2-8: Mercedes Benz with Linguatronic II speech system [33].

In looking at different systems the IBM system boasts a vocabulary of over 300 000 words at installation, while the more specialized Linguatronic II system boasts a smaller vocabulary. The IBM, Microsoft and Dragon Systems are large vocabulary speech recognition systems, while the Linguatronic II system is a small-vocabulary speech recognition system. Most of the systems also allow the user to confirm what they have typed by playing back the typing to the user. The IBM system is able to adjust to background noise while the Dragon system is able to filter out fillers and other irrelevant sounds in dictation speech. Modern systems are often more complex than the older systems. However, due to the fact that the complex and advanced features are implemented transparently, the user literally needs to install the system and start speaking. From the above text, it is easy to see that the user does not get involved with directly fine-tuning the applications. This is due to the fact that the applications are designed to fine-tune themselves without user intervention. They may require the user to speak more.

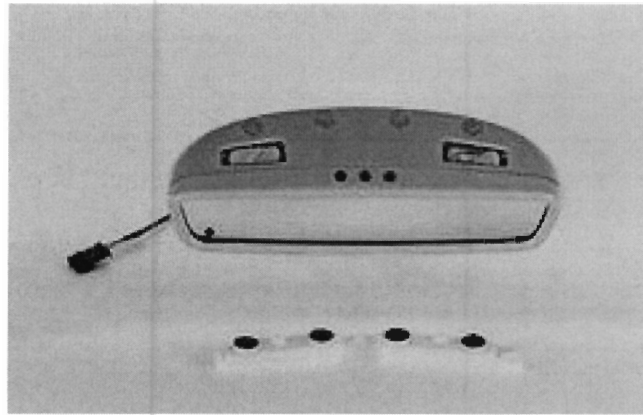


Figure 2-9: Linguatronic II microphone array

2.4 Summary

This section has looked at the fundamentals of speech recognition where the generic model for speech recognition is described. The generic model has two key components viz. the front and back end. The front end performs the capturing of information from the speech input. This speech input is converted by the front-end into speech feature vectors. The back-end uses the speech feature vectors in order to make decision making. Once decision making is done a speech recognition output will be finalized. Since the fundamental design of speech recognition hasn't changed, much of it is still found in many applications today. Various applications like the Mercedes Benz Linguatronic II system are based on such a fundamental design.

Chapter 3

3 HTK

While there are many speech recognition packages e.g. Sphinx, Dragon Talk, Microsoft Speech Tools and Hidden Markov Model Toolkit (HTK) Version 3.2.1, this study required a specific speech recognition package for obtaining results. The package was required to:

- allow for rapid development of speech recognition systems;
- allow for speaker adaptation of the systems;
- allow for rapid implementation of algorithms to be used with these systems; and
- allow for accurate performance measurement.

Hidden Markov Model (HTK) Toolkit version 3.2.1 met the above criteria. It was chosen for the study. The HTK Toolkit comes provided with a host of tools, used for building a fully functional speech recognition system. It is not a “ready-to-use” speech recognition system. But it allows for the building of the speech recognition in 4 steps. Firstly the speech data needs to be prepared. Secondly, the training needs to be conducted. Thirdly, the recognition happens. Finally, the analysis of the performance is described. These steps shall be described in this order.

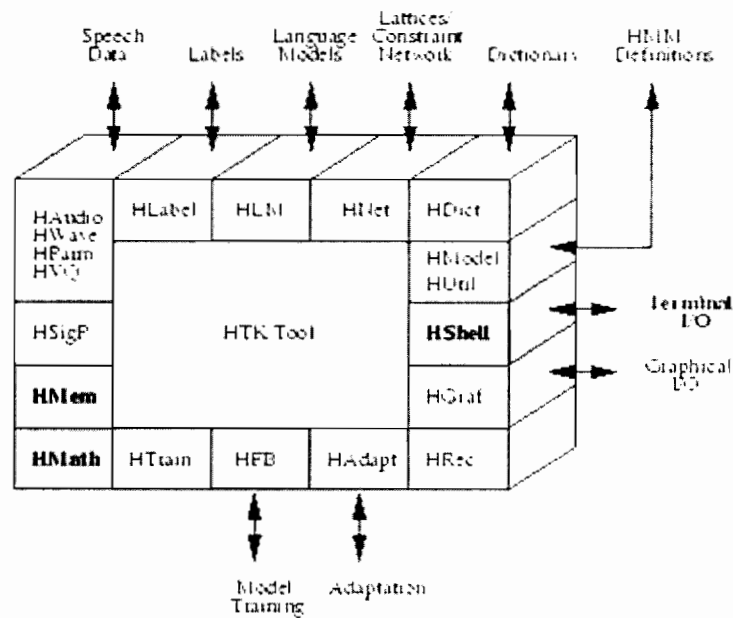


Figure 3-1: An overview of HTK Tools [48]

As seen from [48], HTK Toolkit is a collection of tools, which can be used for training a speech recognition system. Figure 3-1 shows us what components HTK comprises of. It also shows how the components inter-work with the rest of the computer system it is installed in. It is important to understand [48] in order to be able to build and improve speech recognition systems. Many of the tools can also be modified in order to implement new algorithms, which can add performance improvements to the system.

3.1 Data preparation

Prior to any speech recognition training and testing, the speech data needs to be prepared for use with HTK. The data preparation process involves converting the speech files into a parametric representation (feature extraction) that is computer friendly, production of a dictionary of words and pronunciations, the mapping of the words to pronunciation units (phones) and finally the transcription of all training and testing speech. While it is important to have transcriptions for training, it is also important to have test transcriptions in order to check how well the system performs. In this section, how the speech files are converted, how the dictionary is prepared and finally how the transcriptions are done shall be described.

Speech feature extraction

In order to extract the features from the speech wave files, a text editor, which will be used to manipulate the path file is required, a text file-manipulating program, which will be used for merging two files together and the tool **HCopy**, from the HTK Toolkit is required. In order to use the **HCopy** tool, it's necessary to understand how it is used.

HCopy can be used to copy a source file to an output file. In this case, the source file is the wave file and the output file is the MFCC file. In order to process a batch of files, the input and output files are specified as follows:

Table 3-1: A table specifying a batch of speech data files

/home/ofentse/Recog/data/train1.wav	/home/ofentse/Recog/data/train1.mfc
/home/ofentse/Recog/data/train2.wav	/home/ofentse/Recog/data/train2.mfc
/home/ofentse/Recog/data/train3.wav	/home/ofentse/Recog/data/train3.mfc
/home/ofentse/Recog/data/train4.wav	/home/ofentse/Recog/data/train4.mfc
/home/ofentse/Recog/data/train5.wav	/home/ofentse/Recog/data/train5.mfc
/home/ofentse/Recog/data/train6.wav	/home/ofentse/Recog/data/train6.mfc

Table 3-1 illustrates the complete file, which is called **codetr.scp**. Searching for all the speech files, which will be used, and listing them into a file generates the left of the above-mentioned file. Creating the left file and changing the .wav extensions to .mfc generate the right of the above-mentioned file. Inputting the two left and right files into the text file manipulation program mentioned above create the final file. Therefore on the left, the codetr.scp file points to the source speech wave file and on the right it points to the destination MFCC file that will be created by HCopy. In order to specify the kind of features and other parameters are required in the destination file, a configuration file shown below is used:

```
# Coding parameters
SOURCEFORMAT = NIST
TARGETKIND = MFCC_0_D_A
TARGETRATE = 100000.0
SAVECOMPRESSED = T
SAVEWITHCRC = T
WINDOWSIZE = 250000.0
USEHAMMING = T
PREEMCOEF = 0.97
NUMCHANS = 26
CEPLIFTER = 22
NUMCEPS = 12
ENORMALISE = F
```

Figure 3-2: A sample configuration file for HTK

The first line in Figure 3-2 specifies the type of wave file the speech data is encoded as. In this case the speech was recorded using the Microsoft RIFF format. The speech data was then converted using Sound Exchange (SoX) into the NIST format, which is the default HTK format. The second line in Figure 3-2 stipulates the type of features extraction is to be used for the destination file. PLP, MFCC and LPC are some of the options but in this study, MFCC is used. The other lines specified in the file specify other signal processing parameters, which were described in Chapter 2. At this stage, sufficient information has been discussed so as to be able to do feature extraction. In order to do feature extraction, the configuration file as well as the `codetr.scf` file must be input into the HCopy tool. The result will be feature files with the `.mfc` extension.

Dictionary preparation

Numerous dictionaries are available [42]-[44] for speech recognition. For this study, the CMU dictionary with 46 phones is chosen. The main purpose of the dictionary is to show how each of the words is pronounced. It also serves the purpose of indicating different spellings of the same words. The dictionary looks as follows:

AA	sil
CALL	k ao l sp
DIAL	d ay ax l sp
DIRSHAN	d er sh ix n sp
EIGHT	ey t sp
FIVE	f ay v sp
FOUR	f ao r sp
FRANCOIS	f r ah n s w aa sp
LENO	l iy n ow sp
LERATO	l eh r ah t ow sp
LYNN	l iy n sp
NINE	n ay n sp
OH	ow sp
ONE	w ah n sp
PHONE	f ow n sp
SENT-END	[] sil
SENT-START	[] sil
SEVEN	s eh v ax n sp
SIX	s ih k s sp
THREE	th r iy sp
TWO	t uw sp
ZERO	z ih r ow sp

Figure 3-3: A sample dictionary for HTK

The words are situated to the left of the dictionary in Figure 3-3, while the pronunciations are situated to the right of the dictionary. The words indicate a dictionary entry in the speech recognizer's vocabulary while the pronunciations indicate the pronunciation of the words. It is possible to have two entries of the same word on the left, however with different pronunciations on the right. It's important to also note that SENT-START and SENT-END have sil as the pronunciation meaning that they are modelled with the sil model. The main purpose of the dictionary is to store the vocabulary of the speech recogniser and to show how the words are pronounced. In order to generate the dictionary the tool, **HDMan** is used. HDMan is also used for remapping a monophone dictionary to one that consists of biphones and tied-state triphones as shall be seen in section 3.2.

Transcriptions

The transcription of files happens in three stages. The first stage of transcription is to transcribe what the speaker has said by listening to the speech files. There are two different occurrences of this stage. The first occurrence is that the text that the speaker has to speak is pre-defined. In this case the speaker only has to ensure that they speak only the pre-defined text only. Finally in this stage, the utterances spoken by the speaker need to be set out in a master label file. This master

label file will consist of all the utterances by the speaker and one line will be allocated to each utterance as shown below.

```
1. DIAL FIVE SEVEN SIX THREE THREE FOUR FOUR ONE EIGHT FIVE SIX TWO ZERO FOUR SEVEN SEVEN FIVE
2. DIAL FOUR ZERO NINE ONE
3. PHONE OH ONE ONE
4. DIAL NINE EIGHT SEVEN NINE ZERO NINE ZERO OH SEVEN ZERO ZERO SEVEN FIVE TWO THREE ONE ZERO FOUR TWO ONE
5. DIAL THREE FOUR THREE NINE THREE ONE FOUR ONE FIVE ONE FOUR TWO SIX SEVEN ZERO OH ZERO FOUR
6. PHONE FRANCOIS
7. DIAL TWO TWO OH SIX ONE OH THREE SEVEN SIX
8. CALL FRANCOIS
9. PHONE FRANCOIS
10. DIAL OH SIX ONE TWO ONE SIX FIVE ZERO EIGHT ONE
```

Figure 3-4: List of utterances for speech recognition training

The second occurrence is that a human transcriber transcribes what the speaker says. This occurrence is more prone to errors because it may not be easy to make out what the speaker may have said. Like in the first occurrence, the transcriptions also need be set out in a master label file, where one line is allocated per utterance as shown above.

The second stage requires the transcriptions in the label file to be converted into HTK compatible transcription file consisting of all the words in the different utterances. This file consisting of all the words will have all the words in each utterance allocated one line per word. The utterance label above every group of words shows that the words belong to the specified utterance.

```

#!MLF!#
"/sx3.lab"
this
was
easy
for
us
.
"/sx4.lab"
jane
may
earn
more
money
by
working
hard
.
"/sx5.lab"
she
is
thinner
than
i
am
.

```

Utterance words

Utterance label

Figure 3-5: An HTK compatible list of words

The file shown in Figure 3-5 is generated by running a Perl script called **prompts2mlf**. This script is provided with HTK and it requires the master label file that consists of all the utterances to be used in training and testing as input. This output file is usually called **words.mlf**. It is important to get the formats of the file correct as HTK will not work if it is incorrect.

The final stage is to create a list of HTK compatible phoneme transcriptions. One line is dedicated to one phoneme in the master label file that consists of the phonemes. To denote the differences between phonemes from one utterance and phonemes from another, the phonemes are labelled. To denote the end of the phoneme transcription of the utterance, a full stop is used. It is important to ensure that the format of the file is correct otherwise HTK will not work.

```

#!MLF!#
"/sx3.lab"
sil
dh
ih
s
w
ax
z
iy
z
iy
f
ao
r
ah
s
sil
.

```

Utterance label

Utterance phonemes

Figure 3-6: An HTK compatible list of phonemes.

In order to generate the file shown in Figure 3-6, **HLEd** from the HTK Toolkit is used. HLEd requires that the dictionary and the master label file of words as the input. This file is similar to the one depicted in Figure 3-6. However, this file lists the phones in the utterances. It also requires a file consisting of a list of HTK commands which will be used for replacing the words in the master label file with relevant pronunciation models. This file is shown in Figure 3-7.

```

EX
IS sil sil
RE sp

```

Figure 3-7: A list of commands, usually called mkphones.led

From Figure 3-7, three HTK commands which are used for generating the list of phonemes shown in Figure 3-6 can be seen. EX instructs HTK to replace all the words in the words.mlf file with the corresponding pronunciation in the dictionary. IS then inserts silences (sil) at the start and the end of every utterance. Finally RE removes all the short pause (sp) labels that are unwanted in the transcriptions.

3.2 Training the system

In training, there are two important processes. These processes are initializing the models and re-estimating the model statistics until the models are suitable for speech recognition. During the re-estimation process, the training data is re-aligned in order to deal with words that have the same spelling but different pronunciations. In HTK there are five main tools provided for the training process. These tools are HCompV, HInit, HRest, HVite and HERest. HCompV and HInit are used for initializing the models while HVite is used for re-aligning the data and the models. HRest and HERest are used for re-estimating the model statistics. In this section the training process shall be briefly looked at by describing the initialization process, the re-alignment and re-estimation processes.

Initializing the models

Prior to training, we need to initialize the models in order to have a starting point for training. When initializing the models, we are basically making a rough guess of what the HMM parameters might be [48]. HCompV and HInit can be used for this process. HCompV computes the global mean and covariance of the speech data to be used for training. The main purpose of this tool is to set all HMM parameters such that all component means and variances are set equal to the global means and variances [4]. This computation can be used for the flat-start initialization where all models are given the same means and variances. This computation can also be used as a basis for the Fixed Variance and Grand Variance training scheme. Finally HCompV can also be used for computing means and variances for a speaker when doing normalization techniques. HInit computes the parameters for a single HMM by using the Viterbi alignment algorithm. The essence of the algorithm is that it repeatedly segments the speech data and re-computes the HMM parameters by clustering the vectors in the speech segments. If no initial model is provided, then the speech data is equally segmented.

Re-estimation and re-alignment processes

Once the models have been initialised it is now necessary to re-estimate and re-align the data so to make the models more accurate. In making the models more accurate, the aim is to find the models that most likely generate a given speech observation [48]. The Baum-Welch algorithm is used to recursively find the best sequence of states that generates the speech. The main reason for using this algorithm is that it is possible to find the sequence of states even though the states are hidden. That is, since which state out of the set of states forming the model generates a particular observation is not known, the algorithm is used to compute the probability of generating a sequence of observations using the forward and backward probabilities.

HTK provides **HERest** and **HRest** in order to perform re-estimation while re-alignment is done using **HVite**. Generally when 3 iterations of the Baum-Welch algorithm are done, a silence model is estimated. A further 5 iterations of the algorithm are done, where monophone models are fully trained. Once monophone models are fully trained, **HVite** is then used to then re-align the speech data in order to take into account homographs (words with the same spelling but different pronunciation). Once re-alignment is done, 2 re-estimation iterations on the models are performed and then next create triphone models. **HHed** in HTK is used in order to clone triphone models from the monophones. This tool is a command driven editor where the commands are enclosed in a text file. In this case, the text file shown in Figure 3-8 below. Once the triphone models have been cloned, it is important to create tied-state triphones. Before the tied-state triphones can be created, a list of all the triphones that cover training and testing need to be generated by listing all the training and testing dictionaries into the **HDMan** command. Once this is done, the tied-state triphones can then be created from the output of the **HDMan** command. The main reason for creating tied-state triphones, unseen triphones can be synthesised by using decision tree clustering. The state-tying of triphones is controlled by commands embedded in a text file as mentioned before.

```
WB sp
WB s11
TC
```

Figure 3-8: Triphone cloning text file, mktri.led

In Figure 3-9 in the next page, the file `tree.hed` which is used for controlling the state-tying process is seen. Firstly the outlier thresholds are set using the **RO** command and then the statistics

file is then loaded. The QS commands creates the questions that should be used for clustering and the TB commands cluster the specified states. The TR commands set the trace levels of HTK. The AU command creates a list of the extended triphones, while the CO command merges the tied triphones. Lastly, the trees used for the decision process are then saves to a trees file using the ST command.

```

RO 100.0 stats
TR 0
QS "R_NonBoundary"      { ** }
QS "R_Silence"          { +sil }
.
.
.
QS "R_Voiced-Fric"      { +z, +sh, +dh, +ch
QS "R_Unvoiced-Fric"    { +s, +sh, +th, +f,
QS "R_Front-Fric"       { +f, +v }
QS "R_Central-Fric"     { +s, +z, +th, +dh }
QS "R_Back-Fric"       { +sh, +ch, +jh }
QS "R_aa"               { +aa }
QS "R_ae"               { +ae }
QS "R_ah"               { +ah }
QS "R_ao"               { +ao }
QS "R_aw"               { +aw }
.
.
.
TB 350.0 "ST_v_4_" { ("v", "+-v+", "v+", "+-v").state[4] }
TB 350.0 "ST_w_4_" { ("w", "+-w+", "w+", "+-w").state[4] }
TB 350.0 "ST_y_4_" { ("y", "+-y+", "y+", "+-y").state[4] }
TB 350.0 "ST_z_4_" { ("z", "+-z+", "z+", "+-z").state[4] }
TB 350.0 "ST_zh_4_" { ("zh", "+-zh+", "zh+", "+ zh").state[4] }

TR 1

AU "fulllist"
CO "tiedlist"

ST "trees"

```

Figure 3-9: Control script tree.hed

Once the tying process is done, two iterations of the re-estimation algorithms are implemented before a tied-state triphone system is built. This system can be evaluated using tools described in sub-sections 3.3 and 3.4. Once this step has been achieved, further improvements like mixture incrementing can be implemented and these can be re-evaluated using the recognition and evaluation tools.

3.3 Recognition

The trained system is now fully trained, even though further improvements can be implemented. It is now possible to run the recogniser to evaluate how well various training strategies work. However, the basics of recognition involve the transcription of the text from the speech. This is done using a variant of the Viterbi algorithm called the *Token Passing Model* [48].

Firstly before the Token Passing Model is described, a network of words must be defined. There is output at the HMM level. In this research, the HMM level output is the output based on sub-word units called phonemes. In recognition the HMM models are joined to form a word network. This word network is based on a grammar is used to aid the whole recognition process. A token is defined as a partial path through such a network which starts from time $t = 0$ to time $t = T$. As time progresses as speech observations are made, transitions between states are made until the final exit node of the network is reached. That is transitions or paths through the network that best represent the speech observations are made through each HMM model in the network. The transitions are recorded as they occur and in the final recognition result, the recorded transitions are used to represent the final result. This forms a basis of the Token Passing Model. The word network is used to define what a permissible output is and it is aided with the dictionary generated in the data preparation stage. The word network is defined generally in two ways. For small, limited vocabulary systems as in this case, a grammar can be used to define legal word orders and permitted words. This word network is assisted by the dictionary in order to determine which words are permitted (vocabulary). In other cases like large vocabulary systems, in as much a same way that it is difficult to collect data for large vocabulary systems, it is difficult to define grammars for large vocabulary systems. In this case, it is necessary to use *word-loop* based word networks. Typically such networks are augmented with a language model which helps predict which word is most likely to follow the current observed word. Such language models are implemented as N-gram language models where N is the number of look ahead words (when N is 2, 2 words are predicted). Such language models assist the token passing model to choose the best path in order to yield the most likely output. Recognition is done using **HVite**.

3.4 Evaluation

Generally the result of the speech recognition output should be what the speaker said. In many instances the output is not perfect and therefore research is interested in improving the output. In some cases, researchers look for the amount of difference between what the speaker said and what the speech recognition output was. Classically this could be measured using some form of loss function $\lambda(\alpha | \omega)$ [89] where a zero loss is awarded for a correct result and a one loss for an incorrect loss as described in [90]. This can be represented as follows:

$$\lambda(\alpha_q | \omega_r) = \begin{cases} 0 & (q = r) \\ 1 & (q \neq r) \end{cases}, q = 1, \dots, N \quad (3-1)$$

where q is the output while r is the reference.

Generally in the above case, despite the number of insertions, deletions or substitutions that a word may have at any recognition instance, a 1 loss would be allocated to it. However, such a result doesn't model the true performance of the system. Therefore a method based on less stringent penalties is needed and the Levenshtein distance [91] is such a method which many systems are based [93]-[97]. However, several other measures exist. The Levenshtein distance is defined as follows:

$$\lambda(\alpha_q | \omega_r) = \begin{cases} 0, & (q = r) \\ e(q, r), & (q \neq r) \end{cases}, q=1, \dots, N \quad (3-2)$$

where $e(q, r)$ is the Levenshtein distance. This loss function basically finds the smallest number of operations (insertions, deletions etc.) required in order to convert the system result to the ideal result. Two methods used in this study are derived from the Levenshtein distance. They are word correctness and word accuracy (both measured in percentage). These measurement methods are defined below.

$$Correctness = \frac{N - D - S}{N} * 100 \quad (3-3)$$

$$Accuracy = \frac{N - D - S - I}{N} * 100 \quad (3-4)$$

where *Accuracy* indicates the number of words correct including penalties on insertions. D is the number of deletions, while I is the number of insertions, S is the number of substitutions while N is the total number of labels (word units or sub-word units). From the above formulae, it can be seen that the main difference between correctness and accuracy is insertions [48]. In order to calculate the word error rate (WER) of the system, the difference between 100 and *Accuracy* is computed as shown below:

$$WER = 100 - Accuracy \quad (3-5)$$

To consolidate the principles behind the evaluation, the reference below is presented.

Table 3-2: A comparison between a reference sentence and an output hypothesis of a speech recogniser.

Reference:	HENRY IS IN TOWN
Hypothesis:	HNRY ATE IS IN TONN

In the above table the reference is the ideal result. The hypothesis is the actual result generated by the speech recogniser. The above performance measurement methods measure the output performance differently. The hypothesis has an insertion of “ATE”, a substitution of “W” for “N”. It also has a deletion of “E”. Correctness would not consider the insertion of “ATE”, while accuracy would consider it. Thus accuracy is a more representative method of performance measurement [48][54]. Thus accuracy for words is a method that is derived from the Levenshtein distance. Sentence performance is by definition [48] measured as follows:

$$Correctness = \frac{N - S}{N} * 100 \quad (3-6)$$

In the same notion, this does not measure the performance using all the edit operations to be considered and therefore doesn't measure the true performance of the speech recogniser as word accuracy does. This study also implements the Levenshtein distance in order to measure sentence accuracy. While different applications will have different requirements, the public access IVR server will place a premium on accuracy [100]. Therefore the results of the evaluations are based

on the Levenshtein's proposal [65]. For this study the proposal is also extended to the sentence level for HTK.

In order to implement this, the HResults program in HTK is used. Further modifications were made such that the Levenshtein measure is evaluated on sentences. The HResults is a general results program which can evaluate word-spotting and other important speech recognition measures.

HHed and **HEAdapt** are an example of tools that can be used to add performance improvements. The first is a script-based HMM editing tool which is used for cloning HMM's. By default, in training, there is only one mixture per Gaussian model. However, it has been found that incrementing the number of mixtures per Gaussian model improves the performance of the speech recogniser [3][7], [48]. The second tool is also a script-based tool which is used for adapting the HMM's such that it better recognises the speaker which it is intended. This technique changes the values of an SI trained HMM's such that they resemble the values of an SD HMM such that it seems like the HMM was trained by the speaker which the system is intended. Many improvements have been achieved using these tools [3], [7], [48],[49]. Once these tools have been used, evaluation can then recognition can be conducted, as described in sub-section 3.3 and then finally the evaluation as described above.

3.5 Summary

Within the HTK toolkit, a speech recognition engine was designed. The training data was prepared. The speech recognition system was trained using the prepared data. Recognition was performed on the data and finally completed the whole procedure by evaluating the recognition results. The tools which can be used for post-training performance improvement were described. Most of these tools are implemented after training and then are evaluated in the same manner that normal evaluation occurs. The Levenshtein distance was implemented for comparison for evaluating the sentence performance, alongside the existing sentence performance measurement method implemented in HTK.

Chapter 4

4 Speaker adaptation

While it is possible to build a reasonably accurate speech recognition system, it is often necessary to improve its performance. Improvements can be implemented by using more training data. More training data is costly and it is time consuming. State-of-the-art methods like histogram equalization can be used as an alternative. However, this is not standard in HTK. Even if it was standard, it would still require lots of training data. Speaker adaptation is a more suitable method because it requires less data than the other described methods.

Two algorithms are used in this study to evaluate the suitability of them for the public access IVR server. The algorithms are Maximum Likelihood Linear Regression and Maximum a-Posteriori.

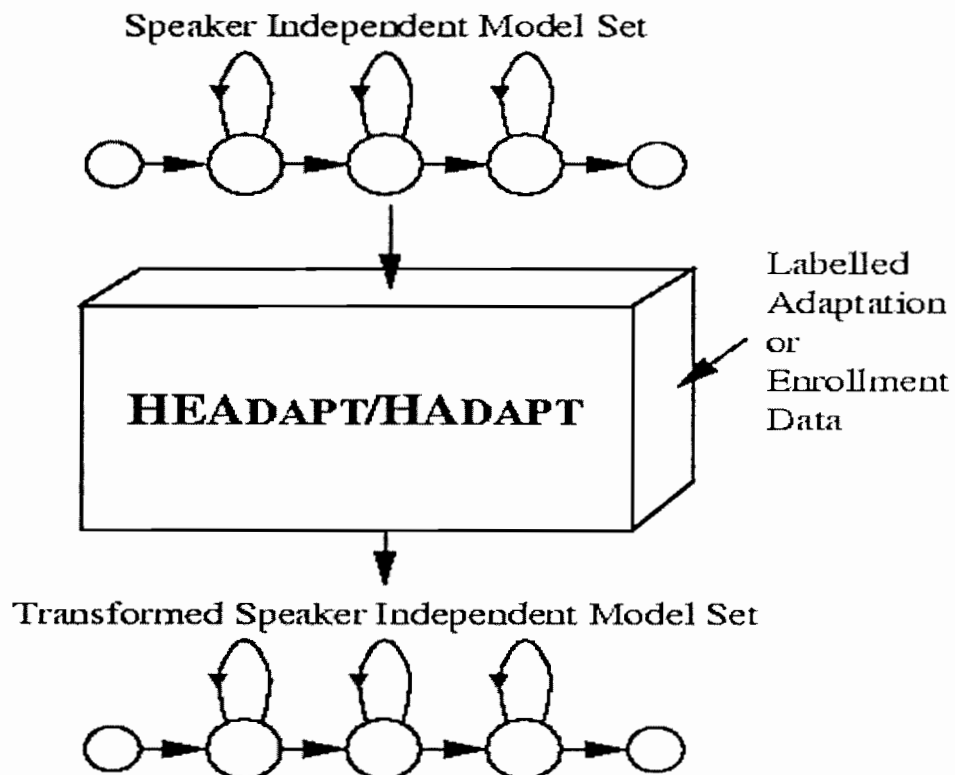


Figure 4-1: Speaker adaptation model transformation [48]

Speaker adaptation forms a continuation of a concept was derived many years ago [104]. The main idea was to estimate transformations from an incomplete set of data. From this idea, MLLR, MAP, DLLR and other variants have originated. All these algorithms estimate HMM parameters from a limited set of data. The way in which they do this is through Expectation Maximization (EM) algorithm. The EM algorithm calculates the maximum likelihood estimate (MLE). MLLR, MAP, and DLLR all have their own variant of the EM algorithm. This is not explained in this study.

4.1 Maximum Likelihood Linear Regression

Maximum likelihood linear regression (MLLR) is a model adaptation algorithm, which operates by generating a set of transformations, given some adaptation data. Firstly assume that the adaptation data consisting of a series of T observations, O is defined as follows:

$$O = o_1, o_2, \dots, o_T \quad (4-1)$$

The generated transformations are used to change the initial HMM parameters of an HMM speech recognition system to new HMM parameters [3], [7], [14][42]. Thus mathematically speaking, a description of the output parameters (new HMM means) is to be given as a linear combination of explanatory variables (as described in sub-section 4.1.1) and this is where the part of linear regression is derived. The parameters (mean and variance) of each state in the HMM speech recognition system are altered by the transformations such that each state has a higher probability of generating the adaptation data that is used. First the linear regression (expressing the solution as a linear combination) is obtained, then solution to the linear regression problem is done using MLE (described in sub-section 4.1.2). Finally, the MLE solution is used to solve the linear regression, hence calling the complete solution MLLR.

4.1.1 Linear Regression: *Expressing the output as a linear combination*

The general process of adaptation is formulated as follows:

$$\hat{\mu} = X\xi \quad (4-2)$$

where, $\hat{\mu}$ is the new mean vector computed from the transformations. X is the transformation matrix that is used to find the new mean, and it is an $n \times (n+1)$ matrix (n is the dimension of the data or simply the number of entries in the observation vectors). ξ denotes the extended mean vector which is derived by taking all the means from each of the entries in the observation vectors. Therefore ξ is defined as:

$$\xi = [\omega, \mu_1, \mu_2, \dots, \mu_n]^T, \quad (4-3)$$

where ω is representing a bias offset. As seen from Chapter 2 the output probability function for each state in the HMM is given as follows:

$$b(o) = \frac{1}{\sqrt{(2\pi)^n |\Sigma|}} e^{-\frac{1}{2}(o-\mu)^T \Sigma^{-1} (o-\mu)} \quad (4-4)$$

It is then easy to see that once the MLLR transformation has been applied, the above formula will then result in an HMM with the following output probability function which is labelled as adapted:

$$b_{adapted}(o) = \frac{1}{\sqrt{(2\pi)^n |\Sigma|}} e^{-\frac{1}{2}(o-X\xi)^T \Sigma^{-1} (o-X\xi)} \quad (4-5)$$

where the mean in the output probability function is now transformed using equation 4-1 as is now shown in the adapted output probability function. Generally equation 4-3 can be thought of as the initial HMM, while equation 4-4 can be thought of the output probability for the new SD HMM.

It is also possible to further represent the matrix X as follows:

$$X = [bA], \quad (4-6)$$

where b is a bias vector and A is an $n \times n$ transformation. And as seen, the new mean vector shown in equation 4-1, is an expression of a linear combination of the transformation and extended mean vectors. In order to solve this combination, we must first find a solution for the transformation matrix X using a technique called *Maximization Likelihood Estimations*. Once the transformation matrix X is solved, the new mean $\hat{\mu}$ can then be computed.

4.1.2 Maximum Likelihood Estimations (MLE)

In order to solve the linear regression problem for speaker adaptation, the transformation matrix that completes the equation defined in 4.1.1 is to be computed. Ideally this computation would be done by minimising the error rate. But since there is no closed-form description of this problem, such an ideal method does not work. A state-of-the-art approach is to modify the initial HMM so that the probability of generating the adaptation data is maximised [52]. To achieve this, a likelihood function shown below is maximised:

$$f_{mle}(\hat{\lambda}) \geq f_{mle}(\lambda) \quad (4-7)$$

where λ are the HMM parameters. In this case, a maximum likelihood estimation (MLE) problem shall be solved using Expectation Maximisation (EM), which has been encountered in previous chapters. An auxiliary function, which has ideal convergence properties and is computationally tractable, is defined as follows [3][52]:

$$Q(\lambda, \hat{\lambda}) = \sum_{\theta \in \Theta} P(O, \theta | \lambda) \log \{P(O, \theta | \hat{\lambda})\} \quad (4-8)$$

where Θ consists of all the possible states that could lead to the recognition of the adaptation data, O . Basically HMM parameters that maximise the auxiliary function above also maximise the likelihood function unless it is already maximised. Successive formation of a new auxiliary function with improved parameters from the HMM in concern successively maximises the likelihood function.

Since only the transformation matrix is to be re-estimated, only the output probability functions $b(o)$ are affected the auxiliary function in (4-2) re-formulated as follows:

$$Q(\lambda, \hat{\lambda}) = const + \sum_{\theta \in \Theta} \sum_{t=1}^T P(O, \theta | \lambda) \log b_{\theta_t}(o_t) \quad (4-9)$$

If S is defined as all the states in the system and the posterior probability of occupying the state s (commonly known as occupation likelihood), at time t , given that the observation sequence O is generated as,

$$\gamma_s(t) = \frac{1}{P(O|\lambda)} \sum_{\underline{\theta} \in \Theta} P(O, \theta_t | \lambda) \quad (4-10)$$

then equation 4-3 can be revised as

$$Q(\lambda, \hat{\lambda}) = \text{const} + \sum_{\underline{\theta} \in \Theta} \sum_{t=1}^T \gamma_s(t) \log b_{\theta_t}(o_t) \quad (4-11)$$

In order to get the total probability, the marginal probabilities over all S are summed up. Therefore the auxiliary function can be revised by re-stating it as follows:

$$Q(\lambda, \hat{\lambda}) = \text{const} + P(O|\lambda) \sum_{j=1}^S \sum_{t=1}^T \gamma_j(t) \log b_j(o_t) \quad (4-12)$$

After expanding $\log b_j(o_t)$ in the previous expressions the auxiliary function is then:

$$Q(\lambda, \hat{\lambda}) = \text{const} - \frac{1}{2} P(O|\lambda) * + P(O|\lambda) \sum_{j=1}^S \sum_{t=1}^T \gamma_j(t) [n \log(2\pi) + \log |\Sigma_j| + h(o_t, j)] \quad (4-13)$$

where

$$h(o_t, j) = (o_t - \hat{X}_j \xi_j)^T \Sigma_j^{-1} (o_t - \hat{X}_j \xi_j) \quad (4-14)$$

Then in differentiating the auxiliary function with respect to $\hat{X}_s$ the following is obtained:

$$\frac{d}{d\hat{X}_s} Q(\lambda, \hat{\lambda}) = -\frac{1}{2} P(O|\lambda) \frac{d}{d\hat{X}_s} * + P(O|\lambda) \sum_{j=1}^S \sum_{t=1}^T \gamma_j(t) [n \log(2\pi) + \log |\Sigma_j| + h(o_t, j)] \quad (4-15)$$

Thus after completing the differentiation, the auxiliary equation is equated to zero, in order to find its maximum.

$$\frac{d}{d\hat{X}_s} Q(\lambda, \hat{\lambda}) = P(O|\lambda) \sum_{t=1}^T \gamma_s(t) \mathbb{E}_s^{-1} [o_t - \hat{X}_s \xi_s] \xi_s^T = 0 \quad (4-16)$$

In finding the solution to the above equation the general form of computing the transformation matrix is obtained as shown below:

$$\sum_{t=1}^T \gamma_s(t) \mathbb{E}_s^{-1} o_t \xi_s^T = \sum_{t=1}^T \gamma_s(t) \mathbb{E}_s^{-1} \hat{X}_s \xi_s \xi_s^T \quad (4-17)$$

This equation yields only yields solutions for specific types of problems and the most notable type of problem is when the covariance matrices are diagonal. Since the matrix is diagonal, only parameters that correspond to the diagonal entries in the matrix are computed. It is possible to further expand the above equation so that transformations for can be computed over all the models in the initial HMM as shown below:

$$\sum_{t=1}^T \sum_{r=1}^R \gamma_{s_r}(t) \mathbb{E}_{s_r}^{-1} o_t \xi_{s_r}^T = \sum_{t=1}^T \sum_{r=1}^R \gamma_{s_r}(t) \mathbb{E}_{s_r}^{-1} \hat{X}_s \xi_{s_r} \xi_{s_r}^T \quad (4-18)$$

where $\hat{X}_s$ is shared by R states as indicated by the subscripts in the above equation. Sub-section 4.1.3 shall describe how the regression classes are computed.

4.1.3 MLLR: *Computing the regression classes*

This section shall describe how regression classes are computed. It will also describe how to effectively utilise the transforms generated using the equations described in sub-section 4.1.1. Now that the previous section has derived the equation for computing the transformation matrix, it shall be seen that there are two levels, which the transformations can be, computed [3],[4], [42], [48], [49]. The first level involves calculating only one transformation, which will be applied to all the output probabilities in the initial HMM. In this case, all the output probabilities in the initial HMM are considered to be representing the same acoustic information. The other level involves calculating many transformations using regression classes and these transformations will be applied to certain output probabilities in the initial HMM [3]. In this case, the output probabilities in the initial HMM are considered to represent different acoustic information, where groupings could be nasals, fricatives, silence, vowels etc. However, while these groupings can be efficient, a dynamic method for grouping, as shall be seen allows for better performance [48]. The dynamic grouping technique that is used to effectively utilise the transformations generated shall be described.

Firstly when there is little adaptation data present, it is possible to generate one transformation which will be used for transforming *all* the output probabilities in the initial HMM. This means that all the initial HMM parameters can be transformed with one transformation in order to generate new HMM parameters which are representative of the current adaptation data. As mentioned before, the initial HMM parameters are all considered to represent the same acoustic

information or grouping and are therefore with the dynamic grouping technique the output probabilities are made into one cluster as shown in Figure 4-2 below.

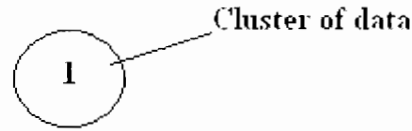


Figure 4-2: Global Transformation for MLLR

Therefore in the case of the global transformation, the new means are computed using equation 4-1. The next step is to compute a transformation that maximises the adaptation data, given that all the output probabilities in the initial HMM will be transformed using that same transformation.

In the case where more than one transformation is to be used, the dynamic grouping technique needs to determine which groups require transformations. Only then are transformations calculated using the acquired information. There are many dynamic grouping techniques available and can be grouped as hierarchical or partitional techniques [53]. In this research, divisive clustering is dealt with. This falls into the hierarchical category, and is called binary regression class clustering. With divisive clustering, it is possible to group the *initial* HMM parameters according to their acoustic similarity using a centroid splitting algorithm based on the Euclidean distance measure [48]. So this divisive process is independent of the adaptation data. Initially everything is grouped into one binary regression tree node. The algorithm then proceeds as follows [48]:

- Select the data (HMM components) in this node that has to be split.
- Calculate the mean and variances of the components at this node.
- Create two children since a binary tree is being used and initialise their (children) means to the parent mean off set in opposite directions by a fraction of the parent variance.
- For each component in the parent node, assign it to one of the children, using the Euclidean distance measure to verify which child the component is closest to.

- Once this assignment has been done, compute the means of the children based on the assignments.
- Continue the assignments and continue the mean re-estimations until there is no change to the assignments. Then finalise the split.
- Repeat the algorithm until the required amount of nodes is reached.

The building of the regression class tree, the SI data is used and as a result, is independent of the new speaker's data. In implementing the regression class tree, the number of required classes is specified. The above algorithm then iterates until the specified number of required classes are attained.

Generally the resulting tree looks as follows (continued in next page):

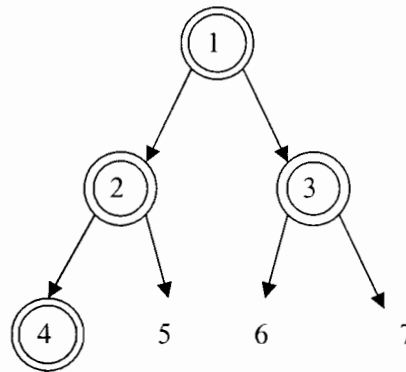


Figure 4-3: Binary class regression tree

where the top node is the starting point. At this starting point, all the data is treated as the same. As further splits are made, the tree grows downwards (hence the hierarchical clustering) and finally 4 terminal nodes at the bottom are reached. Therefore the tree yields 4 base classes, C_4, C_5, C_6, C_7 , representing nodes 4, 5, 6 & 7 respectively. It can be seen that C_4 has sufficient data (hence the solid boundary), while C_5, C_6 and C_7 do not have sufficient data as indicated by their lightness in node colour. In speaker adaptation transformations are generated for nodes with sufficient data and are terminal nodes or nodes with sufficient data have any children without sufficient data. In this case, a transformation X_4 for C_4 is generated. However since C_5, C_6 , and C_7 do not have sufficient data, the transformation X_2 is utilised for C_5 , while for C_6 and

C_7 the transformation X_3 is utilized. Finally in revisiting the case for global transformation (see in next page), again it can then be seen that the initial HMM parameters are placed into C_1 (representing node 1) as shown in Figure 4-4 in the next page. The transformation X_1 is then applied to all the parameters in the initial HMM in order to generate a newly adapted HMM (the faint nodes are only present for illustration purposes).

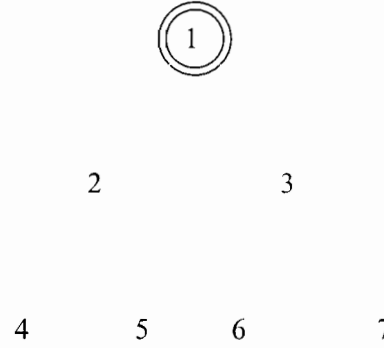


Figure 4-4: Diagram for global transformation

4.2 Maximum a-Posteriori (MAP) Adaptation

Maximum a-posteriori is an adaptation technique that makes use of the prior knowledge of model parameter distributions [48]. Recall from Chapter 2 that the problem 3 of HMM's was about adjusting the model such that $P(O | \lambda)$ is maximised. With the ML estimates, the following is computed:

$$\frac{\partial}{\partial \lambda} P(O | \lambda) = 0 \quad (4-19)$$

where stationary points that lead to the maximisation of the function are found. This is for the case where λ is assumed constant, yet unknown. However in this case the interest is in obtaining a MAP estimate for λ , where λ is assumed random and has a prior knowledge $P_o(\lambda)$ [1]. This is calculated as follows:

$$\frac{\partial}{\partial \lambda} P(\lambda | O) = 0 \quad (4-20)$$

where the adaptation data O is given. $P(\lambda | O)$ can be re-written using the classical Bayes rule as follows (continued in next page):

$$P(\lambda | O) = \frac{P(O | \lambda)P_o(\lambda)}{P(O)} \quad (4-21)$$

where the prior knowledge can then be clearly observed.

If the prior knowledge is showing a preference as to what the model parameter distributions are likely to be, then it is said that the prior knowledge is an *informative prior*. In the case where the prior knowledge does not show any preference as to what the model parameter distribution is likely to be, then it is said that the prior knowledge is a *non-informative prior*. The informative prior is a probabilistic distribution of the model parameters and this is used prior to adaptation in order to generate reliable MAP estimates. If an *informative prior* then is present then it can then be used in computing the MAP estimate in order to minimise Bayes risk [1]. If a *non-informative prior* is present, then the MAP estimate obtained is identical to the ML estimate [48]. It is important to note that in order to compute a MAP estimate given the adaptation data O , the prior must be computed first and then use it to compute the MAP estimate. It is possible to compute the prior from either the adaptation data or the SI data. But before continuing with MAP adaptation, there is an important need to understand the basics of the MAP priors.

4.2.1 Conjugate Priors

The first thing to note is that for MAP adaptation, conjugate priors are used mainly for mathematical tractability [1][48]. A conjugate prior for a random vector is a prior distribution for the HMM parameters of the probability density function of the vector, such that the posterior distribution $P(\lambda | O)$ and the prior $P(\lambda)$ all belong to the same distribution family for any adaptation data O . Since conjugate priors are prevalent in MAP adaptation, this study is limited to their discussion. Next a way of deriving the new MAP estimate shall be presented and then this will be followed by the derivation of prior information.

4.2.2 Computing the MAP estimate

To obtain the MAP estimate (new mean), the following equation is used (continued in next page):

$$\hat{\mu}_{jm} = \frac{N_{jm}}{N_{jm} + \tau} \bar{\mu}_{jm} + \frac{\tau}{N_{jm} + \tau} \mu_{jm} \quad (4-22)$$

where τ is a weighting parameter and μ_{jm} is the mean of the current SI data and $\bar{\mu}_{jm}$ is the sample mean of the adaptation data. $\hat{\mu}_{jm}$ is the newly computed mean which is to be used for the new speaker. At this stage 2 quantities, which are not solved, remain. These are defined next.

4.2.3 Solving the outstanding quantities: prior information

N_{jm} is the occupation likelihood of the adaptation data and this is computed as follows:

$$N_{jm} = \sum_{r=1}^R \sum_{t=1}^{T_r} L^r_{jm}(t), \quad (4-23)$$

and this, as described before, indicates the likelihood of being in a certain state at time t .

The speaker adaptation data mean is defined as follows:

$$\bar{\mu}_{jm} = \frac{\sum_{r=1}^R \sum_{t=1}^{T_r} L^r_{jm}(t) O_t^r}{N_{jm}}, \quad (4-24)$$

where O_t^r indicates a sequence observations at time t , R indicates the total sequence, T the total observations and L^r indicates the probability of state occupation for the r 'th observed adaptation data. When the occupation likelihood of the adaptation data is small, it can be seen, from equation 4-22 that the first term of the equation tends to zero, while the second term tends to μ_{jm} . This means that the newly computed mean $\hat{\mu}_{jm}$ is then based on the means of the current SI data. However, if the occupation likelihood of the adaptation data is big, it can be seen, from equation 4-22 that the first term tends to $\bar{\mu}_{jm}$, while the second term tends to zero. This means that the newly computed mean $\hat{\mu}_{jm}$ is then based on the means of the adaptation data. MAP is therefore able to weight the importance of the adaptation data depending on the amount of adaptation data available. And since the MAP algorithm updates every single mean of the system, the system needs large amounts of data in order for it to work well [48].

It is also possible to get added adaptation performance by first adapting using the MLLR algorithm described previously. In this case, μ_{jm} in equation 4-22 is replaced with the mean computed in equation 4-1. This shall be demonstrated in the experimental results section. However, in the next section, first the combination of the two algorithms, MLLR and MAP shall be described mathematically.

4.3 A combination of MLLR and MAP

As seen in the previous section, it is possible to combine MLLR and MAP in order to obtain improved performance. Again it is possible to improve the performance of the systems. However, it is also possible to experience performance degradation as with the individual algorithms.

It is therefore critical that the working of the combination of algorithms is well understood. In order to understand how the algorithms work, it is important to look at how the combination of algorithms is formulated. The formulation will immediately assist with the understanding of the algorithms, which can initially be confusing.

Recall from a previous section that MLLR adaptation is applied as follows:

$$\hat{\mu} = X\xi \quad (4-25)$$

where a transformation matrix is used to multiply an extended mean vector. It is also important to note that this adaptation can be applied using either one transformation or many transformations, depending on which one gives the best performance.

Next, recall that MAP adaptation is applied as follows:

$$\hat{\mu}_{jm} = \frac{N_{jm}}{N_{jm} + \tau} \bar{\mu}_{jm} + \frac{\tau}{N_{jm} + \tau} \mu_{jm} \quad (4-26)$$

and μ_{jm} is the mean for the SI data (the model parameters). In combining the algorithms together, the mean for the SI data is basically obtained from the mean from equation 4-25. First, equation 4-25 is applied to the SI data to yield a new mean. This new mean is then used by equation 4-26

in order to generate yet another new mean. This method yields additional performance. Next, the difference between the algorithms shall be described.

4.4 Differences between the algorithms

Two state of the art algorithms have been described. One of the key factors contributing towards this study is the differences between the two algorithms. Since the algorithms operate differently, they will therefore have differences. In this section the key differences between the algorithms will be looked at and how these differences are expected to operate under different conditions.

Firstly, the way in which the algorithms operate is looked at. In sub-section 4.1 it was seen that MLLR is a Maximum Likelihood-based algorithm while MAP is a prior-based algorithm. This difference can be seen when looking at equations 4-1 and 4-21. Both equations compute the adapted means but, the difference lies in how the means are computed. In equation 4-1, it can be seen that in order to compute the new means, a transformation matrix, which will be used to transform the old means, must be computed. However, in equation 4-21, prior information about the parameter distributions must firstly be obtained.

Secondly, while both algorithms are able to make use of the limited amount of adaptation data, they use different strategies for utilising the data. It can also be seen in MLLR, that one transformation can be applied to all the old means to obtain the new mean. In this case this transformation is called global. However, it is also possible to apply particular transformations for certain groups of means (where the groups are defined by regression classes) such that the groups of means get a more accurate transformation. In contrast, in MAP, all the means get a detailed treatment. In this case its possible weight the information prior so as to give either the adaptation data some priority or its possible to give the SI means the priority. However, MLLR is better suited at utilizing a limited amount of data.

Thirdly, the control of the priority emphasis of adaptation data is done differently by each algorithm [49]. For MLLR a single regression class is used to control the priority emphasis of the adaptation data in the sense where all the means are transformed based on one transformation generated from the adaptation data. Otherwise, a more detailed approach is where many regression classes can be used such that detailed transformations are generated for each class.

Finally a key characteristic about the algorithms is that they perform differently with the variation of the amount of adaptation data available. MLLR is well suited to situations where the adaptation data is limited. However, MAP is not well suited in such situations. MAP is better suited to situations where the adaptation data is more abundant and generally performs better as the data increases. In such situations, MLLR generally reaches its asymptotic performance.

Below is Table 4-1, where the differences between the two algorithms are tabulated. As mentioned about, the list the differences. While MAP adapts with a MAP formula, MLLR uses regression classes to generate transformations for calculating the new mean. As a result MAP is component specific while MLLR can be general or more specific. In MAP this is controlled using weighting while MLLR uses regression classes.

Table 4-1: Differences between algorithms

	MAP	MLLR
Adaptation	Adapts using formula	Transformation based
Process	Component specific	Flexible
Control	Data weighting	Regression classes
Required Amount of data	Lots	Little

All in all, it can be seen that the performance of the different algorithms means that MAP needs more data to perform well, while MLLR performs well with little data. This plays a very important role in this study as shall be seen in later chapters.

4.5 Summary

The two state of the art algorithms have been described in detail. It was seen that MLLR computed the new means by first generating a transformation matrix. This was achieved by generating transformation based on the adaptation data. When the transformation matrix is generated the means of the old HMM's can then be transformed using the transformation matrix.

This yields new HMM means. In MAP, it was seen that the mean of the old HMM's as well as the mean of the adaptation data was needed in order to generate the new HMM's means. Finally it can be seen that the key differences between the two algorithms lie in the way in which the means are computed, the flexibility of how the means are computed, how the flexibility is implemented and finally the different conditions which the algorithms perform.

Chapter 5

5 Evaluation of the system

In this chapter an evaluation of MLLR, MAP and MLLR+MAP (described in Chapter 4) is conducted. The results of the evaluation are analysed and an account for any improvements (or degradations) is given. The key aim is to evaluate the suitability of the speaker adaptation algorithms for the public access IVR server. In Section 5.1, a description of the speech database will be given. Section 5.2 will describe the general design of the experimental system. Section 5.3 will give the results obtained from the evaluation. Section 5.4 will finally give a discussion of the obtained results.

As explained previously, a large amount of speech data is required for training a public IVR server. Most commercial speech databases are compiled in countries other than South Africa. In the same way that systems trained on native speakers are not well suited for non-native speakers [76], systems trained on the speech databases are not very well suited for a South African speaker. In light of the population, building a speaker dependent (SD) public IVR server is not feasible; a speaker independent (SI) public IVR server is therefore the best choice. However, SI systems do not perform as well for an intended speaker as SD systems would be. Speaker adaptation shall be used to improve the performance of a SI system such that it ideally approximates the enrolling speaker. Speaker adaptation is chosen because it requires less speech data than the training stage requires. In the next section the speech database and the task used shall be described.

5.1 Experimental database and task

Prior to the study, an experimental speech database was constructed. The database was generated by recording several people reading out dialling prompts. The vocabulary of the database consists of all the fundamental digits in the English metric numbering system, proper nouns (mainly people's names) as well as the verbs for dialling. The main purpose of this database is to perform preliminary evaluations for a public access IVR server and as a result this database is similar in size to that implemented in an IVR system by MARCIF [82].

In further explaining the terminology, the order in which the names and digits follow is called a task [13]. The task therefore defines which word will follow another in an utterance. In other words, the task defines how the vocabulary is used. For this study, a telephone dialling task is used. This database also consists of verified transcriptions for each and every transcription in the database.

The database consists of 12 speakers. Each data collection session collects about 100 utterances from each speaker. In the data collection session the speaker will read out pre-set sentences using a dictation microphone. Each session requires about 1 hour 15 minutes per speaker, while verification of the speech and transcription requires 30 minutes per speaker. In summary the speech database has over 1200 sentences which are sufficient for this study.

While many commercial speech databases like TIMIT, NTIMIT and NIST are available for training SI systems [12],[14],[17], [19],[41], they do not cover South African English dialects. As a result these commercial speech databases are not suitable for this study [76]. The building of the speech database gives a good data set for evaluating the speaker adaptation algorithms. Again, the database is comparable to the one used in the MARCIF system [82]. In comparing the speech database with a commercial database like TIMIT, each speaker provides considerably more utterances than in TIMIT. This allows for speaker adaptation to be investigated thoroughly. In the next section, the system design and implementations shall be described.

5.2 System design and implementation

All the systems are based on the HTK 3.2.1. The flat-start procedure is used to initialise the models. This procedure is designed specifically for sub-word models [42] which are also used in the experimental work. The state-of-the art training technique, Baum Welch re-estimation (described in Chapter 3) is used while testing uses the popular Viterbi decoding [1][4][48]. Other aspects which were applied to the systems are listed below:

- Since the systems are HMM-based, all phones are initialized (flat-start) from a 5-state left-right HMM prototype. This means all HMM's representing phones are initialized with one global variance and mean. And the advantage is that all models get reasonable starting parameters. Each state initially has a single Gaussian mixture density, where all

but the first and last states are emitting. The Gaussian mixture density is incremented for performance gains.

- A total of 49 phones are used. This general guideline is obtained from the TIMIT database where silence (sil) and short pause (sp) are modelled.
- A custom database which was generated is used. This has been described earlier in this chapter.
- A dictionary was then generated by us based on the database. This dictionary is such that it maps to the 49 phones. Therefore the dictionary includes short pause (sp) and silence (sil).
- The main measurements are taken from the fully trained tied-state triphone systems as they yield the best performance [3],[4].
- Feature extraction is based on Mel Frequency Cepstral Coefficients (MFCCs). The front-ends performed this using a Hamming window which is 25ms with 10ms advances. The pre-emphasis coefficient (see Chapter 2) is set to 0.97, while 12 static features with energy and with delta and acceleration information appended. The final output of the features is a 39-dimension feature vector.

5.3 Evaluation

In this section, the results of the evaluations are presented. When presenting graphical results, the performance of an unadapted system is indicated by using a horizontal line unless stated otherwise. Such a line will be labelled as either speaker dependent word error rate (SD WER) or speaker independent word error rate (SI WER). This will indicate clearly the speaker adaptation performance in relation to the error rates of the starting (unadapted) system. In sub-section 5.3.1, the speaker adaptation algorithms shall be verified. Sub-section 5.3.2 shall present results obtained from adapting SI systems. Methods for performance improvements shall be presented in sub-section 5.3.3. Unsupervised speaker adaptation results shall be presented in sub-section 5.3.4, while sub-section 5.3.5 shall present a comparison between the current sentence correctness from HTK and the Levenshtein distance measure that is implemented for this study. The analysis of the amount of data used in obtaining the results shall be presented in 5.3.6.

5.3.1 Verification of the speaker adaptation algorithms

In this section the results obtained by using the HTK manual settings for the adaptation algorithms are presented. They represent the results obtained from SD systems, which are tested on other speakers rather than the intended speaker. The system is trained on one speaker. Since the speaker is going to be the one using the system, they are then known as the intended speaker. The system is then tested on the intended speaker and this performance is represented as SD WER. The system is then tested with another speaker (not used in training). This result is then represented as the SI WER. The system shall then be adapted to the other speaker using a varying amount of utterances (varying amount of data) using MLLR and MAP. In this case, the training and adaptation portion of the data is used for adaptation. This is not the case with other future tests, where only adaptation portion of the data is used. The results are then represented by global MLLR, regression class MLLR, MAP and MLLR+MAP WER. Global MLLR represents MLLR adaptation where one MLLR transformation is applied to all the parameters of the system. Regression class MLLR represents adaptation where regression classes are used to transform the parameters of the system. MAP represents the map adaptation applied to the parameters of the system, while MLLR+MAP represents the adaptation of the parameters of the system using MLLR adaptation then finally adapting with MAP adaptation. In this case, a default value of 32 regression classes is initially used. MAP represents results obtained after adapting using the MAP adaptation algorithm with a scaling factor of 12.0.

This section's purpose is to show that the algorithms work correctly and have been invoked as such. Therefore this section is accompanied with some expectations. It is expected that the WER will be high, when another speaker tests the system. However, it is also expected that the intended speaker uses the system, the WER shall be low [3]-[7]. It is expected that a system generally improves as it is adapted to another speaker. As more adaptation data is used, it is also expected that the performance will increase accordingly. However, due to factors linked to the model parameters e.g. overtraining of parameters, poor data to system parameters ratio, estimation errors [77] it's also expected that the performance to degrade at times [7] rather than improve.

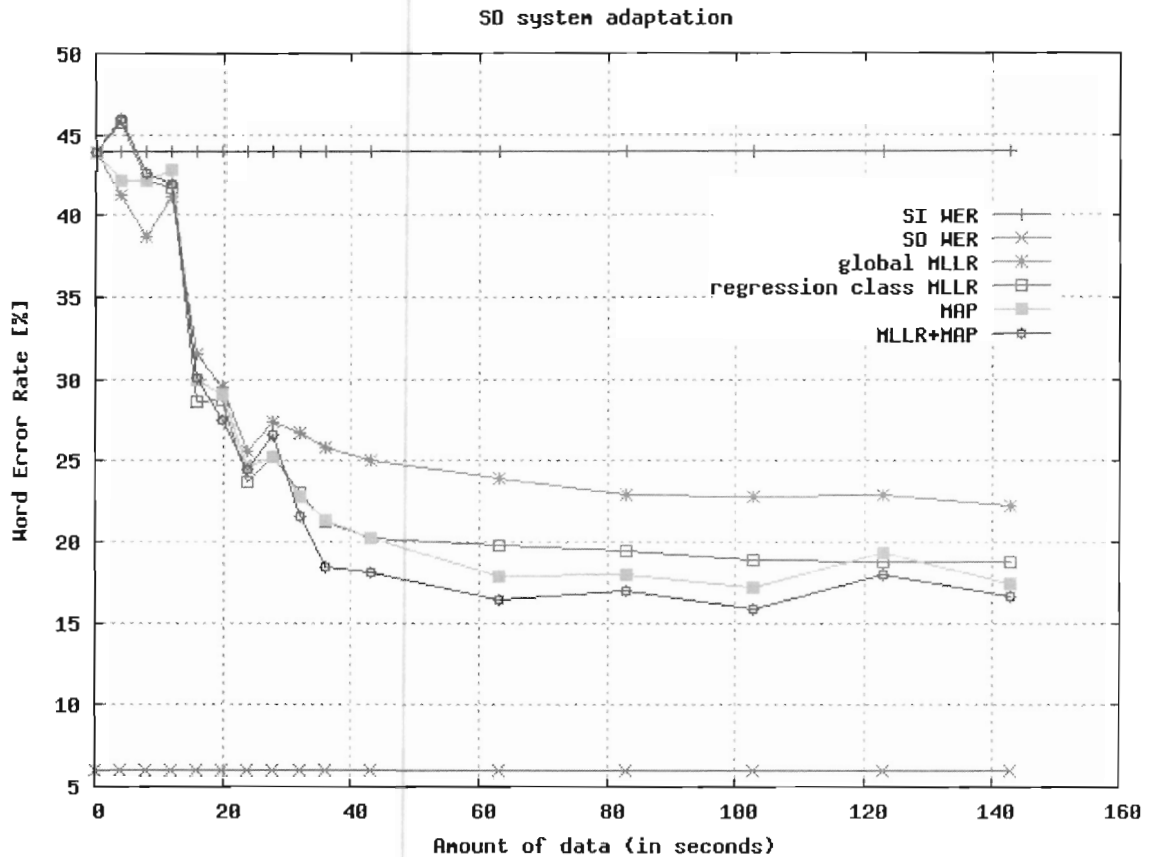


Figure 5-1: Performance of the adaptation algorithms on SD systems.

Figure 5-1 illustrates the results obtained from the system. As described before, the system is trained on one speaker, known as the *intended speaker*. It is tested on the intended speaker, and a SD WER is recorded. It is then tested on another speaker, and a SI WER is recorded. It is then adapted and tested on that speaker. The results of the adaptation are labelled in the figure using the names of the algorithms. The results show a low SD WER, as expected, since the system models the intended speaker very well. When the system is tested on another speaker, a high SI WER is attained as expected. The reason for the high WER is that the system does not model the speaker very well. When speaker adaptation algorithms are applied for this speaker, a reduction in WER is attained. Generally as adaptation is applied, the WER increases at some points and decreases at some points. When the WER decreases, good parameter estimates from the limited data were being obtained such that the speaker is modelled well. However, poor parameter estimates can also be obtained, raising the WER thus poorly modelling the speaker. For small amounts of speech data, it is observed that the WER of different algorithms is not too different. MAP and global MLLR adaptation show an initial decrease in the WER, while MLLR+MAP and regression class MLLR show an increase in WER. However, as the amounts of speech data increases, a general distinction between the WER of different algorithms becomes clearer. The key thing to observe is that global MLLR does not reduce the WER as much as the other

algorithms at larger amounts of data [3]. MLLP+MAP gives a general WER decrease even at greater amounts of data, as it first adapts with global MLLR and then MAP. If both algorithms give good estimates, then it is possible to have the additive effect give good results as in the case above. While regression class MLLR gives better estimates than global MLLR as expected, the MAP WER lies between the WER of global MLLR and regression class MLLR. However, as expected, the speaker adaptation algorithms do reduce the WER of the systems, when they obtain good estimates of the model parameters.

5.3.2 Performance of speaker adaptation algorithms on SI systems

This section presents the performance of speaker adaptation algorithms on SI systems. These systems are trained on many people, as would be the case with the public access IVR server. Since the speaker that will use the public access IVR server is highly likely not to be known, a “worst case” test scenario is taken in order to achieve results. In knowing that a system performs well for the speaker it was trained with, the tests ensure that the system is tested with a speaker who was not used in the training process. This is then the “worst case” test scenario and is applied throughout all the tests in order to produce results which are close to reality.

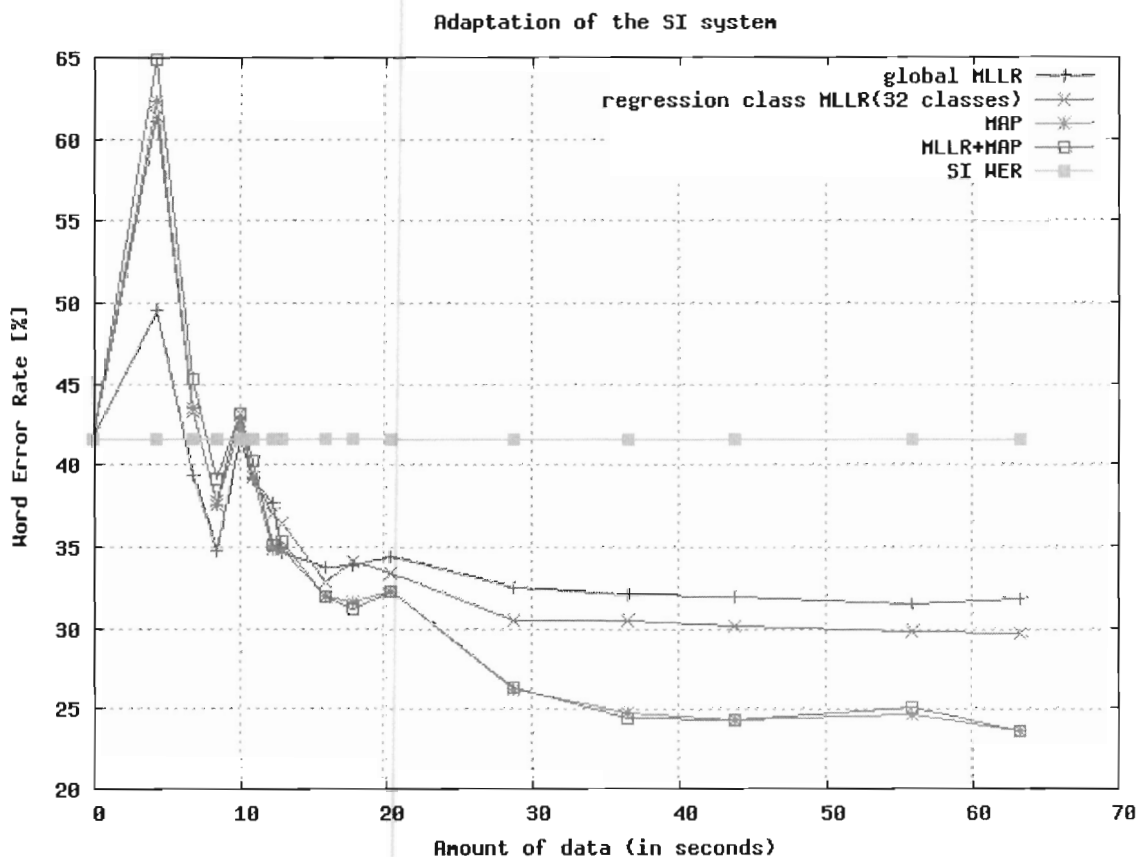


Figure 5-2: The performance of speaker adaptation algorithms on SI systems

The SI WER is illustrated as a blue horizontal line in Figure 5-2. This indicates the performance of the system, shortly after it has been trained and installed. Speaker adaptation algorithms are applied in order to reduce the SI WER. Thus if the speaker adaptation algorithms' WER is below the horizontal blue line, then they have improved the system. If the speaker adaptation's WER is above the blue line, then they have degraded the system's performance. Initial adaptation shows degradation in performance. In this case the small amounts of data do not enable the algorithms to obtain a good estimate for the new model's parameters. Applying global MLLR adaptation to the small amounts of data gives initial performance degradation which is less than for other algorithms. However, as the amounts of data increases, all the algorithms give a general performance improvement with respect to the SI WER. With amounts of data greater than 55 seconds, MAP adaptation with a scaling factor of 12 yields the lowest overall WER, while MLLR+MAP adaptation (derived from global MLLR) and to some extent regression class MLLR yields the next lowest. The WER of global MLLR is, as expected, the highest at larger amounts of data. Overall the algorithms result in a lower WER when compared to the reference point, the SI WER. However, as explained before, it is not always possible to reduce the WER as the algorithms may not estimate obtain the best model parameter estimations. In this case, this is observed when the amount of data is less than 10 seconds.

5.3.3 System parameter variation

This section presents the results obtained from system parameter variation. In system parameter variations, the performance of the system in sub-section 5.3.2 is improved by changing the system parameter settings. Firstly the HMM Gaussian mixture components are incremented. Secondly, the regression classes of MLLR adaptation are varied. Thirdly, the scaling factor of MAP adaptation is also varied. Finally the optimal performance using MLLR+MAP adaptation is obtained from the parameter settings derived from MLLR and MAP. A comparison of the results obtained after 60 seconds of adaptation data is provided.

Figure 5-3 shows the performance after mixture incrementing. The system shown in Figure 5-2 is used as a starting point (SI WER only). The Gaussian mixtures are incremented over a range of 1 to 10 mixture components per state since this work focuses mainly on speaker adaptation. In this case all HMM state mixtures were incremented globally as opposed to increasing particular states and models. The effects of incrementing the mixtures are such that the new speaker is better

modelled by the system [11][54]. If the speaker is not ideally modelled, it is also possible that mixture incrementing yields performance degradation. A particular case of this would be that the data may be insufficient for estimating additional mixture components as has been the case in this work. In some cases it is so severe that the system may become useless. In order to avoid this problem, default limit values were chosen to prevent the mixtures from becoming useless. As observed in Figure 5-3, the WER initially decreases until there are 5 mixture components per state. At this point the WER increases until 6 mixture components per state are used. Further increments yield decreasing WER as the system again models the speaker accurately.

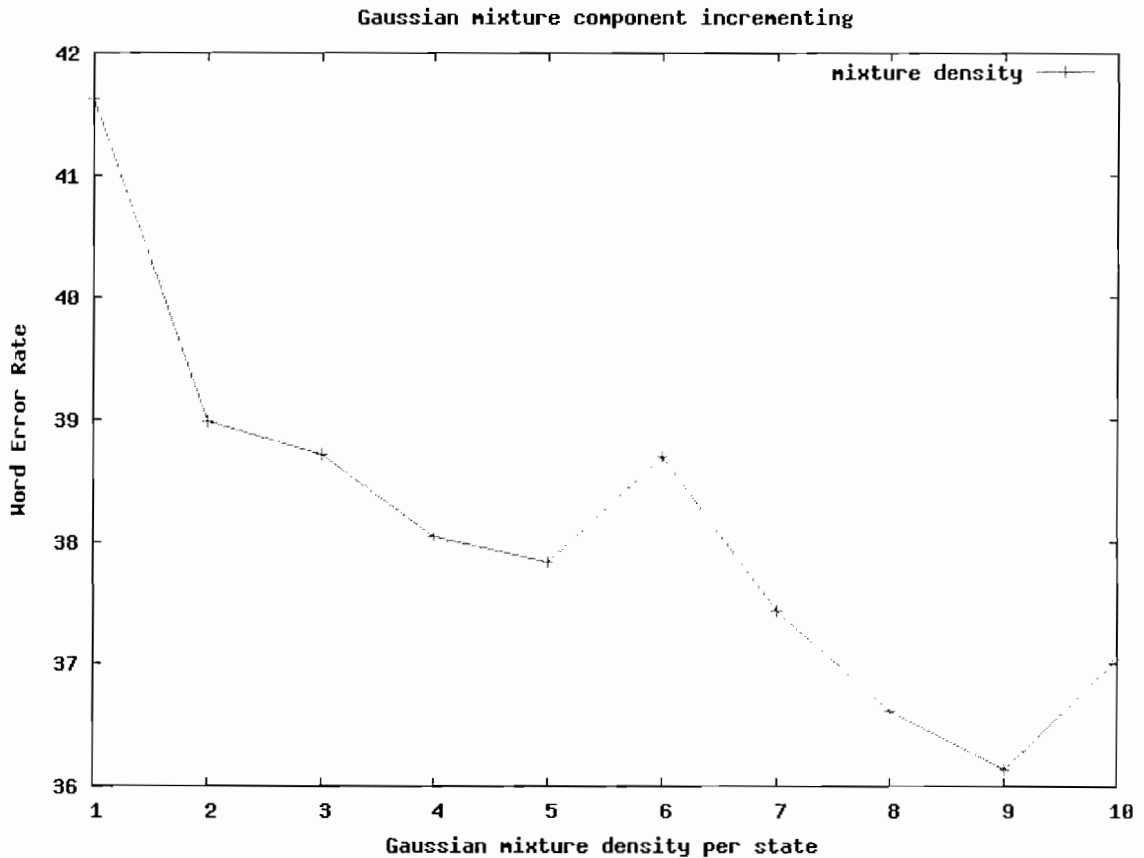


Figure 5-3: Mixture incrementing of SI systems

From the speech database used, 9 Gaussian Mixture components per state yielded the best performance on average. However, each system's configuration may differ from this average value. As a result all adaptation from this point on is done on systems with optimized Gaussian mixture densities in order to have the lowest WER in the defined mixture increment range. Such systems are collectively named optimized SI system and are labelled as such in future figures and tables.

The next figures illustrate the performance of MLLR speaker adaptation. As noted in Chapter 4, it is expected that regression class MLLR with one regression class will yield a global MLLR solution. Figure 5-4 confirms this attribute since the plots for global MLLR and regression class MLLR with one class are the same. When adapting with both MLLR adaptation algorithms the performance of the system generally improves relative to the SI WER. Initially the performance improves, however, after 8 seconds of data, the performance degrades. After 12 seconds of data, the performance increases once again only to decrease again after 16 seconds of data. Generally the performance of the initial system in Figure 5-4 is improved by the algorithms. In looking at the performance of regression class MLLR, the results are identical to the ones yielded by global MLLR. It can therefore be concluded that regression class MLLR performs the same as global MLLR in this situation. Both algorithms improve the performance of SI systems in this configuration.

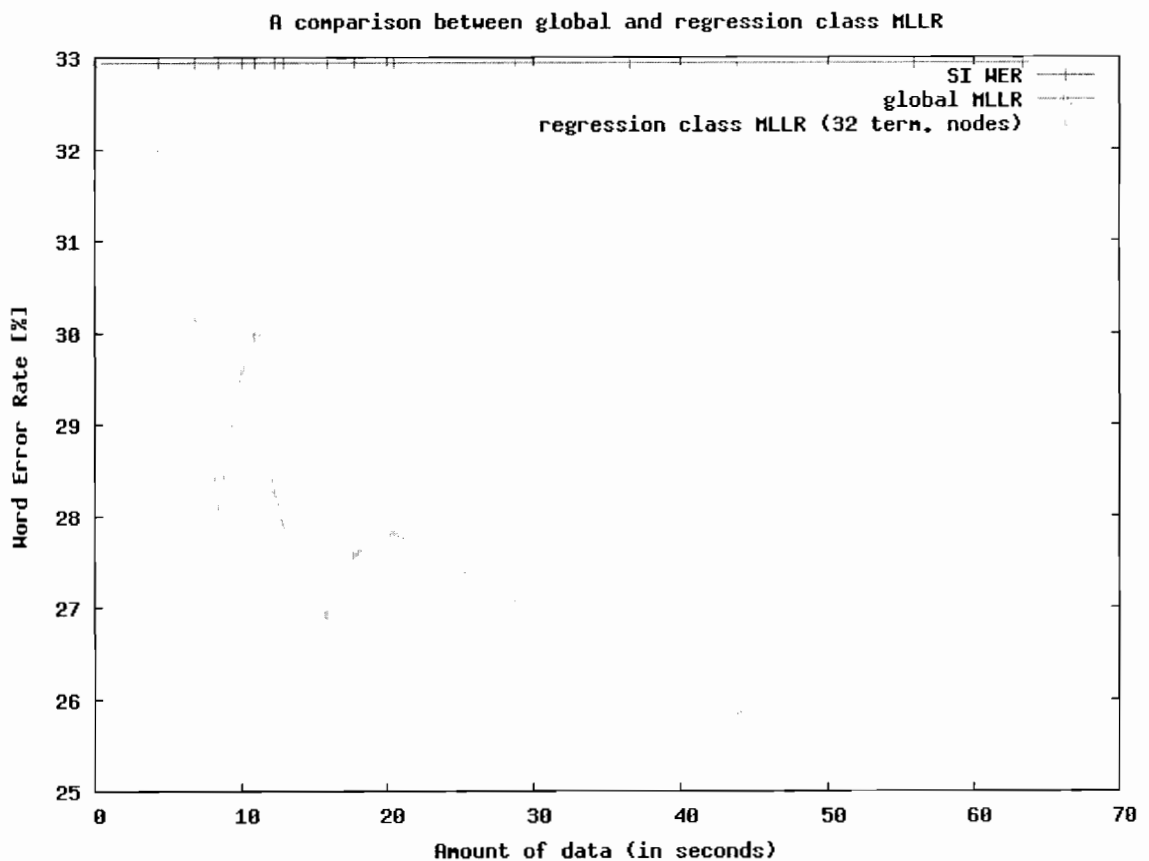


Figure 5-4: A comparison between global MLLR and regression class MLLR with one regression class.

The effect of increasing the regression classes is investigated next. Regression class MLLR results differ from the results in Figure 5-4 in that MLLR adaptation can be applied to particular

models of the system to get better results. Figure 5-5 presents the results from adapting the optimized SI systems. It also includes the results of global MLLR adaptation on the optimized SI systems. The regression classes are then incremented from 2 classes to the maximum of 256 classes. This comparison is presented in Figure 5-5.

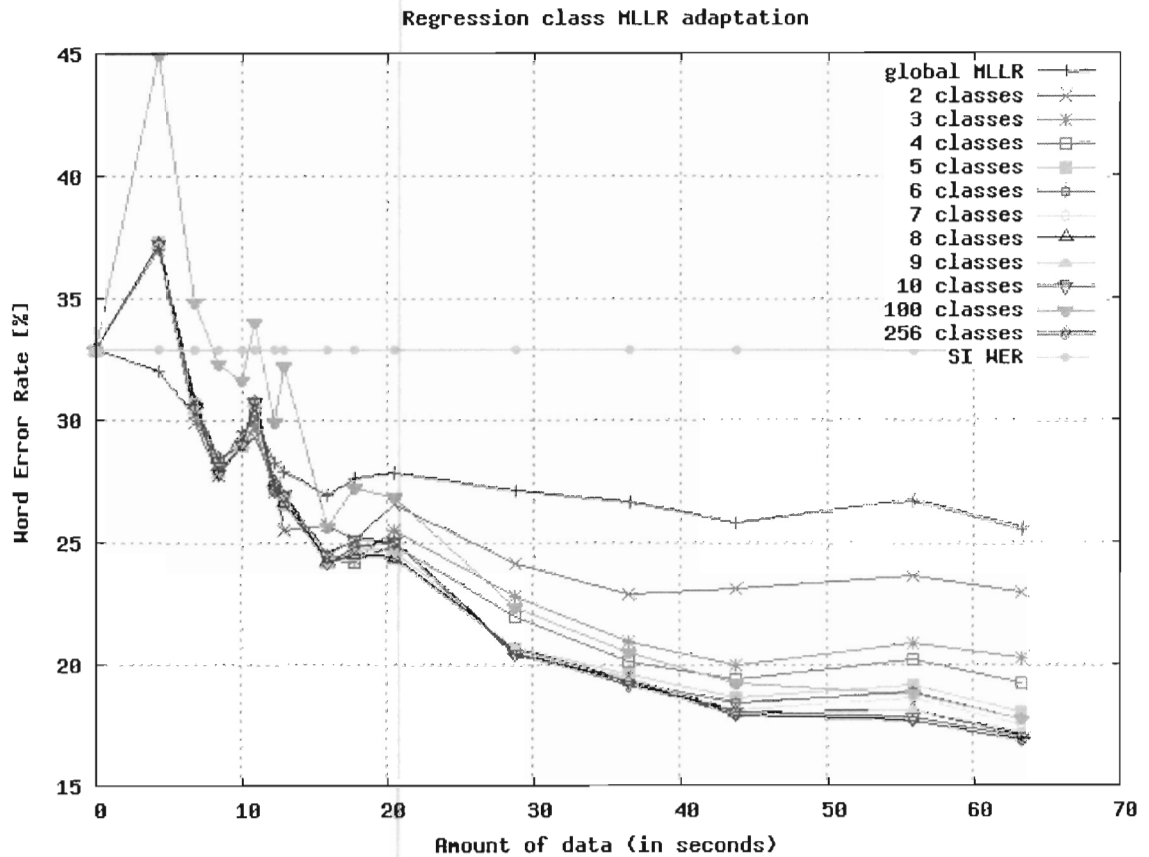


Figure 5-5: MLLR speaker adaptation with increasing number of regression classes

The global MLLR adaptation yields the same performance curve as shown in Figure 5-4 while the SI WER is the same as shown in Figure 5-4. As the number of regression classes is increased, the performance of the systems generally improves compared to the SI WER and global MLLR adaptation. However, when 100 classes are used, performance degradation is seen when compared to the general increase in performance as the number of classes increase. When more than 60 seconds of adaptation data is used, the greatest improvement in performance is achieved with 256 regression classes albeit marginal over 10 classes. In general the trend is that as the regression classes are increased, the performance of the systems increases. This is evident when over 60 seconds of data is used for adaptation especially when compared to global MLLR. Over the adaptation range, the performance fluctuates when smaller amounts of data are used (less than

20 seconds). This is attributed to the fact that the MLLR estimates obtained with those amounts of data are not sufficient for a performance increase. As the amount of data increases, the estimates fluctuate less often giving predictable performance. In this case it would be ideal to use at least 20 seconds of data in order to be within the area where the system's adaptation performance is predictable.

The effects of changing the MAP scaling factors are investigated next. For MAP, if there is insufficient adaptation data, the models are maintained as *prior* means, where the prior means are the means of the informative prior. However, if there is sufficient data, then the adaptation formulae described in Chapter 4 apply, where the informative prior is used. Importantly, if there is no informative prior, then the new estimate will tend towards the ML approach otherwise, which is principally the same as MLLR. Figure 5-6 below illustrates MAP adaptation.

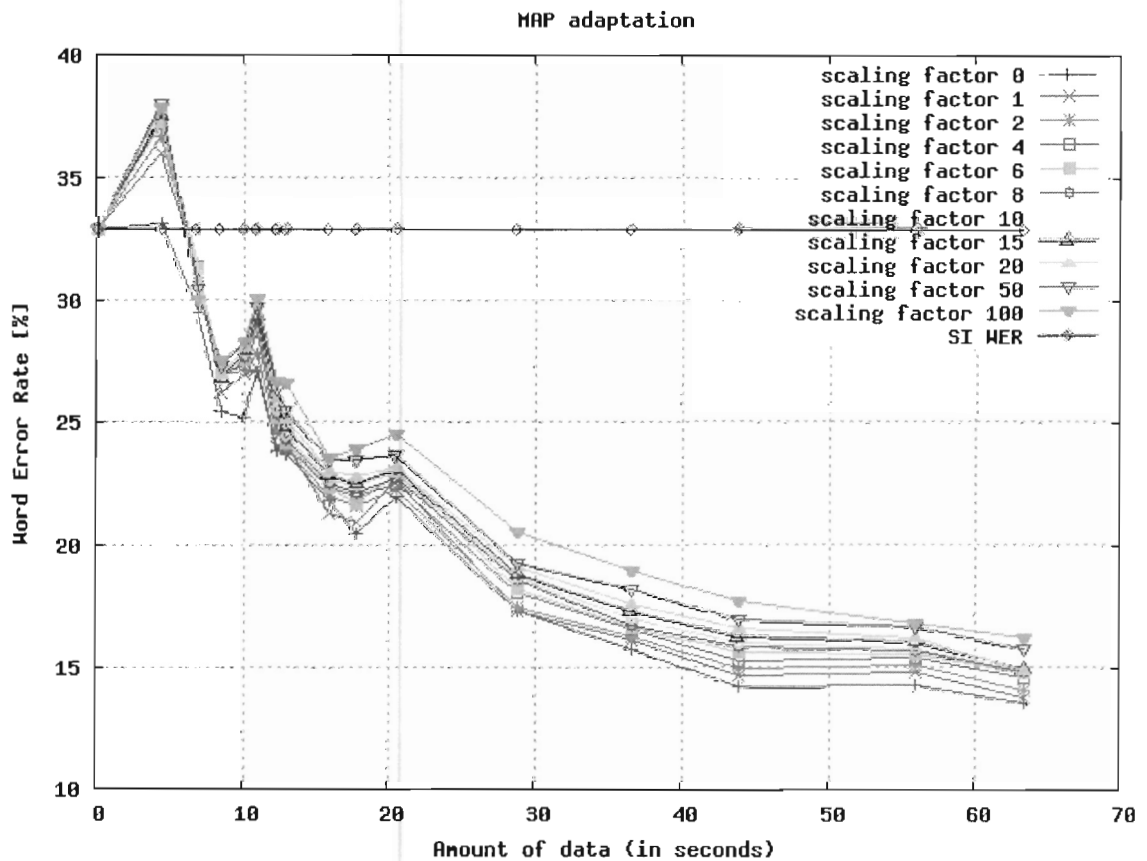


Figure 5-6: MAP adaptation with decreasing data scaling values

The scaling factor is varied to obtain different WER curves. The prior used for MAP adaptation is the SI model. The prior information is used in conjunction with the adaptation data in order to adapt all the models. With the scaling factor it is possible to give priority to the adaptation data or

the prior information. A wide range of scaling factors is possible for MAP adaptation. Since when the scaling factor is increased from 0, the general adaptation performance of the system decreases the scaling factor range was only chosen to run from 0 to 100. As with previous figures, it can be seen that for amounts of data less than 7 seconds, there is generally no improvement compared to the SI WER. However, after 7 seconds of adaptation, the performance increases gradually. This performance increase becomes a performance decrease after 9 seconds of adaptation. The performance increases again after 12 seconds adaptation, which then decreases again at 16 seconds of adaptation. From a little more than 20 seconds of adaptation the performance generally increases without out much further fluctuations. When less that 20 seconds of adaptation data are used, there are more fluctuations. After 60 seconds or more of adaptation, MAP adaptation produces the best results. On average, the best performance improvement was obtained by using a scaling factor of 0. This scaling factor basically relies entirely more on the speaker adaptation data means, rather than the SI informative prior (this can be shown mathematically using the formula in Chapter 4). Therefore if the scaling factor is non-zero, then MAP adaptation depends on the SI prior and also the adaptation data. However when it is 0, then MAP adaptation depends on the adaptation data only.

Figure 5-7 illustrates the results obtained from combining MLLR adaptation with MAP adaptation in order to get improved performance.

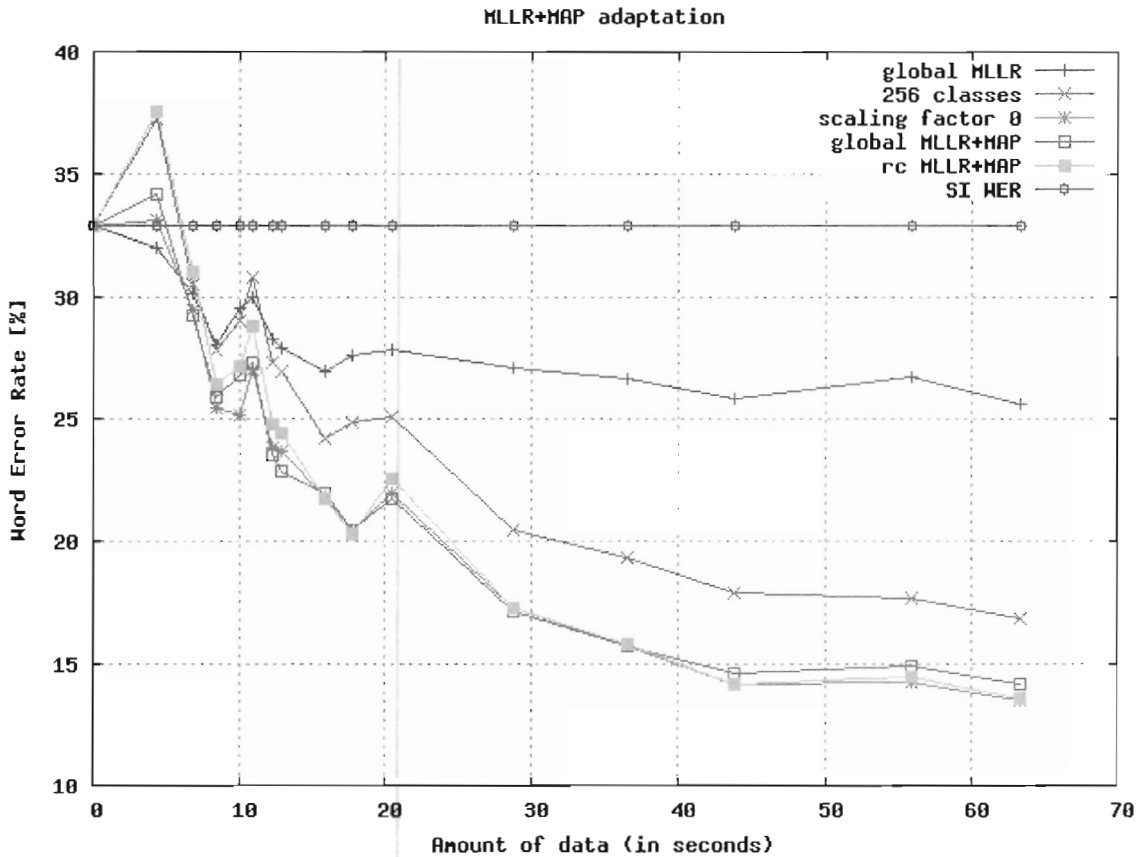


Figure 5-7: MLLR+MAP adaptation

The SI WER is the same result obtained in Figure 5-6 while the MLLR result is the best results obtained in previous MLLR tests. The MAP result is the best results obtained for MAP previously. The MLLR+MAP results are the main results of interest in the figure above. In comparison to the previous figures, MLLR+MAP and MAP offer reduced WER on adaptation when compared to the SI, the global MLLR and the regression class MLLR adaptation WERs. The success of the combination of the algorithms is attributed to the way in which the algorithms work. Initially when adapting with MLLR+MAP with regression class MLLR, a slight increase in the WER is observed. However, thereafter this increase becomes a decrease, showing some improvement. With adaptation data of less than 20 seconds, the WER increases at two points viz. starting at 8 seconds of data and also 18 seconds of data. When more than 20 seconds of adaptation data are used, then there is less increase in WER. Between 28 and 36 seconds of adaptation data, MLLR+MAP adaptation with global MLLR seems to have a similar performance as MLLR+MAP with regression class MLLR. However, MLLR+MAP with regression class MLLR have the edge thereafter. At this stage, MAP adaptation seems to do better than MLLR+MAP with regression classes.

Table 5-1: System performance after 60 seconds of adaptation data

System tested	Algorithm Used				
	MLLR(g)	MLLR (rc)	MAP	MLLR+MAP ^g	MLLR+MAP ^{rc}
Original system	41.63	41.63	41.63	41.63	41.63
Optimized SI system	32.94	32.94	32.94	32.94	32.94
Adapted System	25.58	16.87	13.47	14.21	13.60

Table 5-1 above illustrates the different levels of improvements in the systems. Firstly the original system is improved and is thus called the optimized SI system. Secondly, the optimized SI system is then adapted and the adapted system is named as such. Mixture incrementing is used to improve the original system to become the optimized SI system as seen in Figure 5-3. The WER is reduced considerably as a result of optimizing the number of Gaussian mixture components per state. This WER reduction is due the fact that the speaker is modelled ideally. Speaker adaptation is then used to adapt the optimized SI system in order to generate the adapted systems. As a result, the WER of the systems reduce. However, since different adaptation methods are used, differing results are obtained. When adapting using global MLLR, the WER is reduced from 32.94% to 25.58%. This improvement is due to the improved modelling of the speaker through mathematical transformations of models. When regression class MLLR adaptation is used, the WER reduces from 32.94% to 16.87%. In this case, detailed transformation of the system contributes in the WER reduction. Again with the case of MAP adaptation, the WER is also reduced, however, from 32.94% to 13.47%. This is also attributed to the better modelling of the speaker through mathematical transformations. Better settings chosen for the algorithms also contribute to the improvements. This is evident in the algorithms as they are iterated with various settings. MLLR+MAP also reduces the WER from 32.94% to 14.21 (for global MLLR) and from 32.94% to 13.60% (regression class MLLR) when applied on the optimized SI system. Since these results are improvements over the original system, it goes to show that mixture incrementing is an additive improvement to the adaptation algorithms.

^g These results represent MLLR+MAP with global MLLR adaptation iterated before MAP adaptation.

^{rc} These results represent MLLR+MAP with regression class MLLR adaptation iterated before MAP adaptation.

5.3.4 Using unsupervised speaker adaptation

This sub-section presents the results obtained after applying unsupervised speaker adaptation to SI systems. This adaptation is termed unsupervised because no transcriptions are available for adaptation. Previous adaptation tests done in this study are supervised. Unsupervised adaptation utilizes the utterances from the enrolling speaker in order to generate transformations to be used for adaptation. For unsupervised adaptation only MLLR can be used, due to the limitations with the HTK package. Since the enrolling speaker will unlikely to be known, the public access IVR server will need to generate transcriptions by using unsupervised adaptation. Initial results obtained from the SI system after over 60 seconds of adaptation data are shown below.

Table 5-2: Results for unsupervised adaptation after using over 60 seconds of adaptation data.

Unsupervised speaker adaptation	
Method	WER
Unadapted SI system	41.63
Global MLLR	31.914
Regression class MLLR	29.694
MAP	23.694
MLLR+MAP(g)	23.654
Unsupervised MLLR	41.905

Table 5-2 above illustrates a comparison between unsupervised and supervised adaptation. As explained before only MLLR unsupervised adaptation is done. Initially the settings obtained from the HTK manual are used. These settings utilize 32 regression classes and a MAP scaling factor of 12. The table shows results after an amount of at least 60 seconds of adaptation data is used. Unadapted SI system represents the SI WER of the optimized SI system. Global MLLR represents the performance of the optimized SI system after adaptation using global MLLR adaptation, while regression class MLLR represents the performance after using the regression class MLLR adaptation. Finally MLLR+MAP (g) represents the adaptation where global MLLR adaptation is used prior to applying the MAP adaptation, where the scaling factor is 12. All the algorithms described are supervised except where stated to be unsupervised MLLR. In

comparison to the SI WER, the supervised adaptation algorithms generally improve the system. This is consistent to tests in subsection 5.3.3, where the adaptation algorithms generally improve the SI system. Supervised adaptation allows for accurate estimation of transformations from adaptation utterances in order for accurate adaptation to be done. In the case transcriptions must be generated during adaptation in order to do accurate adaptation. As a result, the transformations that unsupervised adaptation produces are not as good as the transformations that supervised adaptation produces. Therefore, unsupervised adaptation in this case does not model the enrolling speaker very well, hence the low performance.

Table 5-3: Results for unsupervised adaptation after using over 60 seconds of adaptation data.

Unsupervised speaker adaptation	
Method	WER
Optimized SI system	32.94
Global MLLR	25.58
Regression class MLLR	16.87
MAP	13.47
MLLR+MAP(g)	14.21
Unsupervised MLLR (g)	32.35

Table 5-3 above illustrates the comparison between unsupervised global MLLR adaptation and its supervised equivalent. Adaptation is done on the optimized SI system. Thus in comparison to Table 5-2, the unadapted SI system WER has decreased from 41.63% to 32.94 which represents the optimized SI system WER. Other adaptation algorithms are included in Table 5-3 for comparison purpose. As with the results presented in the previous table, generally supervised adaptation algorithms generally improve the SI system. The best supervised adaptation result is obtained by using MAP adaptation with a scaling factor of 0 and the WER is then 13.47%. In the case of the unsupervised global MLLR adaptation algorithm, a WER of 32.35% is yielded. This is an improvement compared to the SI system's WER of 32.94%. However, when comparing the unsupervised global MLLR WER with the supervised global MLLR WER of 25.58%, again it is seen that supervised adaptation has an advantage over unsupervised adaptation. In an overall comparison between Table 5-2 and Table 5-3, it can be seen that an SI system which better

models the speaker allows for greater WER reduction from speaker adaptation as shown with the lower WER in general.

Table 5-4: Results obtained after using over 60 seconds of adaptation data for adaptation

Unsupervised speaker adaptation	
Method	WER
Optimized SI System	32.94
Global MLLR	25.58
Regression class MLLR	16.87
MAP	13.47
MLLR+MAP	14.21
Unsupervised MLLR	30.25

Table 5-4 above illustrates unsupervised MLLR adaptation after 256 regression classes are used. The majority of this table is the same as Table 5-3. The only difference is the performance of the unsupervised MLLR adaptation. Regression class unsupervised MLLR is used as the unsupervised MLLR adaptation. The supervised adaptation algorithm WER is better than that of the optimized SI system and thus shows an improvement. The lowest WER from the supervised adaptation algorithms is from using MAP adaptation with a WER of 13.47%. Unsupervised adaptation WER is also better than that of the SI system and also better than the unsupervised global MLLR WER in Table 5-3. Therefore unsupervised regression class MLLR adaptation performs better than unsupervised global MLLR adaptation. Although the unsupervised adaptation WER is an improvement relative to the SI system, it still doesn't better the performance of supervised adaptation algorithms in review.

5.3.5 A comparison between sentence accuracy and Levenshtein edit distance

This sub-section describes the results obtained from the comparison between HTK sentence correctness and the implemented Levenshtein edit distance. The current HTK method for measuring sentence correctness does not take into account the atomic differences between the reference sentence and the output sentence. For example, if the reference sentence is meant to be “the cat jumped over the wall” and the output is “the cat jumped over the woll”, then the system will be measured to be correct for five out of six words. This measure equates to a correctness of about 83%. This method of measurement penalizes the whole word that is incorrect, thereby reducing the overall scoring of that sentence.

This study compares the existing HTK sentence correctness measurement method with an implemented method –the *Levenshtein edit distance*. This distance penalizes the sentence according to the distance between the two sentences being compared. Basically the number of edit changes needed to convert the output sentence into the reference sentence is considered the distance. Since the implemented method takes into account the distance differently to HTK’s method, it scores higher than the existing HTK sentence recognition measurement method. It is also important to note that it is also possible for this method to score higher than the word accuracy of the adaptation algorithms because it also scores the spaces within the sentence, unlike the word accuracy. While the HTK sentence correctness measurement method penalizes the incorrectly recognized word, the Levenshtein distance only penalizes the word by considering the number of changes required to make the word correct. For the given example, the Levenshtein distance should score about 90%. In this study, the Levenshtein distance is compared to the word accuracy of the system instead of the word correctness. This is due to the fact that the way the Levenshtein distance is closely related to the way in which the word accuracy is computed [89]. However, since the Levenshtein can be extended to the sentence level, it is evaluated as an alternative to the HTK sentence correctness measurement method.

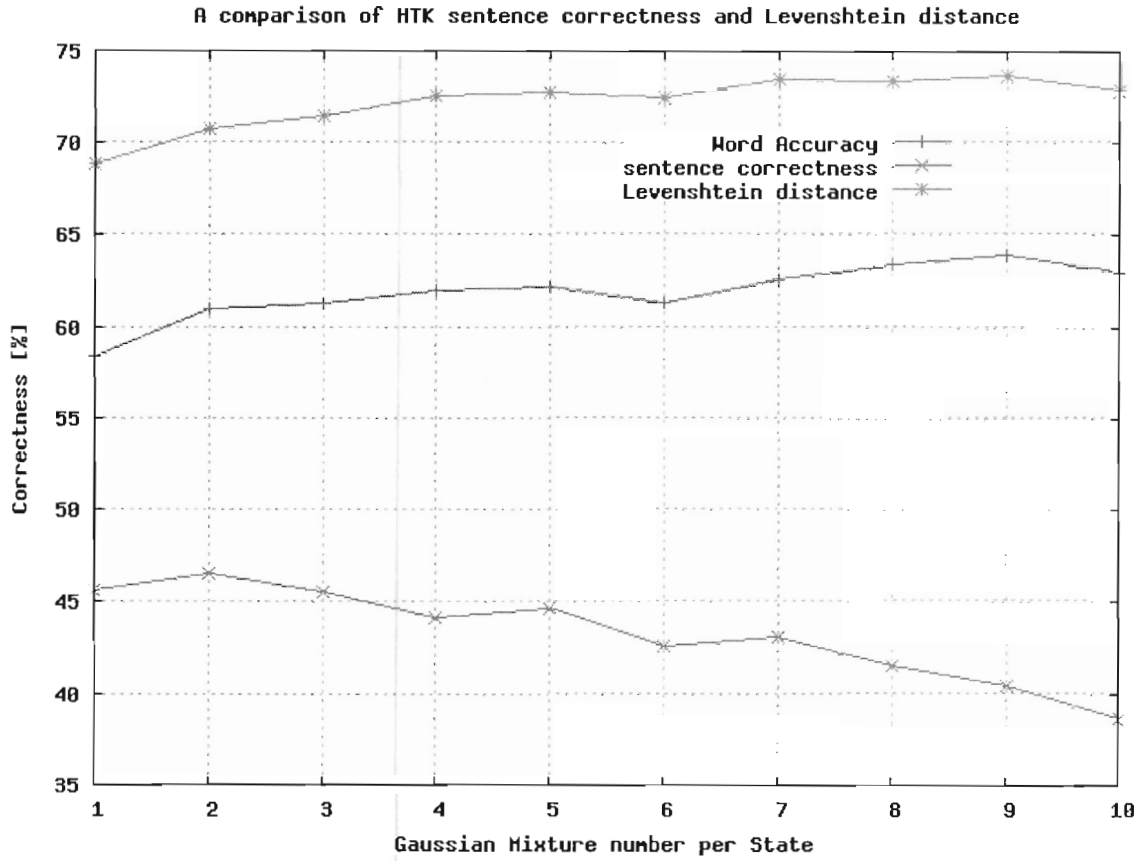


Figure 5-8: A comparison between HTK sentence correctness and Levenshtein distance

Figure 5-8 shows a comparison between the existing HTK sentence correctness measurement method and the implemented Levenshtein distance during the mixture incrementing process. The figure shows the word accuracy of the SI systems, the sentence correctness of the SI systems and finally the Levenshtein distance. The initial word accuracy is 58.4%, the sentence correctness is 45.6% and the Levenshtein distance is 68.8%. The word accuracy of the system increases the Gaussian mixtures per state is increased. This trend changes slightly when there are 6 Gaussian mixture components per state. At this stage, there is a slight decrease in performance. However, after this slight decrease in performance, the performance increases again. As the trend increases again, at 9 Gaussian mixture components per state the performance peaks, and then drops slightly at 10 Gaussian mixture components per state. This performance trend corresponds with the WER trends shown in Figure 5-3. When comparing this performance trend to the HTK sentence correctness measurement, a different story is told. It peaks when there are 2 Gaussian mixture components per state, and then shows a general performance decrease thereafter. The lowest performance is therefore at 10 Gaussian mixture components per state. While the word accuracy intuitively suggests that the sentence may be correct, the sentence correctness measurement by HTK shows otherwise.

The Levenshtein distance shows a similar trend to the word accuracy. Its performance shows a general performance increase which also has a slight decrease when 6 Gaussian mixture components per state are achieved. When further mixture incrementing occurs, the performance increases and peaks at 9 Gaussian mixture components per state. Finally the performance decreases at 10 Gaussian mixture components per state, just as with word accuracy. The Levenshtein distance therefore gives a more closely related result to word accuracy than sentence correctness is to word accuracy.

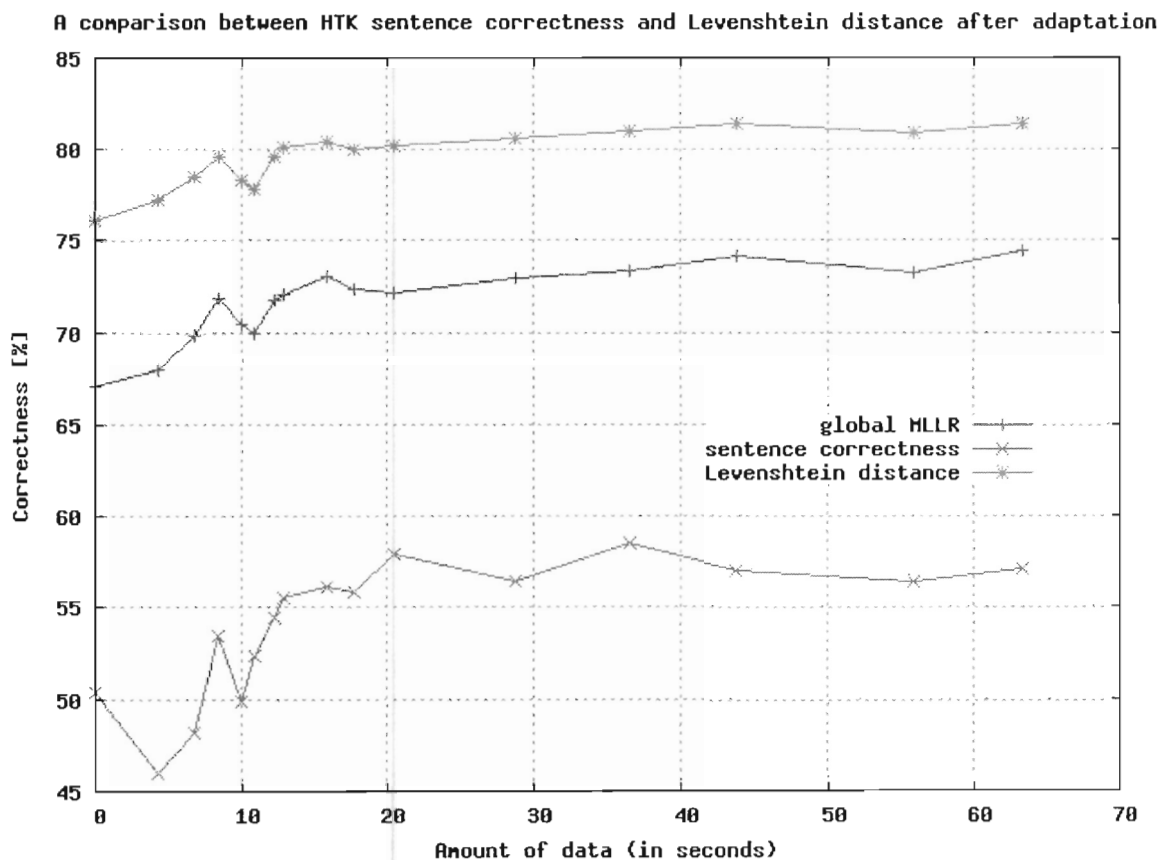


Figure 5-9: Performance measurement comparison between global MLLR and Levenshtein distance

Figure 5-9 above shows a comparison between sentence correctness, word accuracy and the Levenshtein distance after global MLLR adaptation. The initial word accuracy of the system is 67.1%, while the initial sentence correctness of the system is 50.42%. The Levenshtein distance is initially at 76.1%. Generally global MLLR adaptation shows an improvement as an increasing amount of adaptation data is used, much the same as the test in previous sub-section. However, it is important to note that the sentence correctness that is measured after initial adaptation shows

degradation in performance. As more adaptation data is used, the sentence correctness measurement increases to about 53%. The performance then briefly degrades before it then improves when 10 seconds of adaptation data is used. The performance of adaptation on the sentence correctness peaks only after 30 seconds of adaptation data has been used. However, due to the performance degrades slightly after 35 seconds of adaptation data and generally fluctuates throughout the adaptation data range. Overall the performance of the system when measured using sentence correctness does not peak at the same time as the word accuracy. The Levenshtein distance however, follows a similar pattern to the same pattern as the word accuracy. So in general where the word accuracy peaks, the Levenshtein distance will exhibit some kind of peak. For amounts of data less than 20 seconds, the performance is not predictable and varies. The sentence correctness shows a drastic drop in performance, while global MLLR experiences a smaller drop in performance. However, the Levenshtein distance experiences minimal variation in performance. As more data is used the performance varies to a lesser extent, and as the amount of data approaches 60 seconds the performance stabilizes. The main reason for the fluctuations is the fact that the global MLLR estimate does not give a good estimate with small amounts of data. Adaptation data may also not be representative of the new speaker models resulting in performance degradation. This has been a similar trend with previous tests shown in this study. As the data increases the global MLLR estimates for the new model for the speaker are more reliable hence less fluctuations.

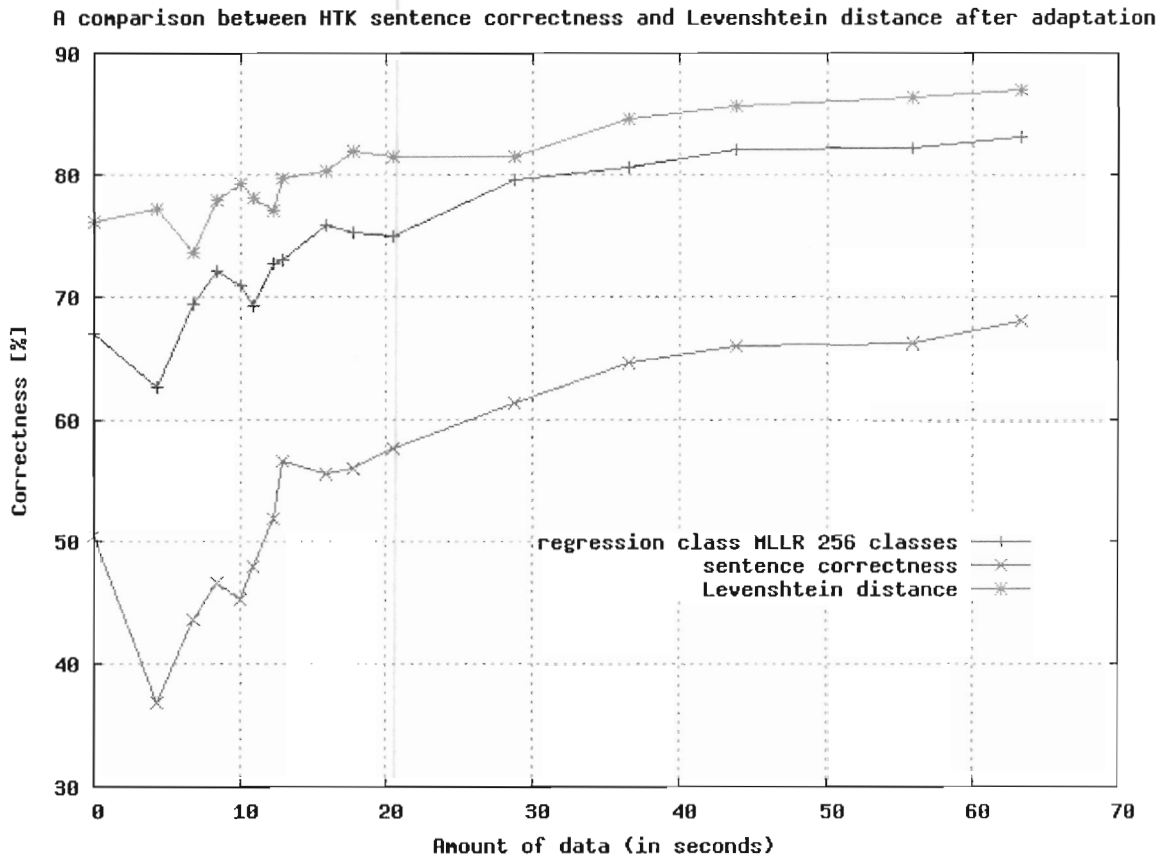


Figure 5-10: A comparison of sentence recognition evaluation methods

Figure 5-10 above illustrates a similar pattern as in Figure 5-9. Again the initial values for the optimized SI system (when adaptation data is 0) shown in Figure 5-9 are assumed. When regression class MLLR is applied to the SI system, the word accuracy generally increases. It initially decreases for amounts of data less than 7 seconds. However it increases thereafter and then decreases at 8 seconds of data. At 11 seconds of adaptation data the system then improves gradually until it experiences a slight decrease in performance when about 16 seconds of adaptation data are used. After 21 seconds of adaptation data the performance starts increasing again where it generally increases. The sentence correctness performance generally shows a similar pattern to the word accuracy although to a lesser extent. Initially after adaptation the sentence correctness degrades. The performance then increases up until 8 seconds of adaptation data are used. At this stage there is a slight degradation which then turns to a performance increase through out the rest of the adaptation range. The Levenshtein distance also follows a very similar trend of increase as the word accuracy. The initial performance shows an initial improvement. However, the word accuracy and the sentence correctness show a decrease. This is attributed to the fact that the words in the recognised sentences are in their correct order since the Levenshtein distance gives an improved score. However, due to insertions, deletions and

substitutions, the word accuracy shows a decrease. For the same reason, the sentence correctness also shows a decrease. A similar trend can be seen when 8-10 seconds of adaptation data are used. Between 20 – 30 seconds of adaptation data, the Levenshtein distance shows a slight increase in performance, while the word accuracy shows a definite increase. This is mostly attributed to the fact that there are fewer edit operations in the words thus an improvement in word accuracy, while the white spaces and word orders have not given the improvement required to show a drastic improvement with the Levenshtein distance.

Generally as with previous tests, for amounts of data less than 20 seconds, fluctuations in performance are seen. For amounts of adaptation data greater than 20 seconds there are fewer fluctuations. The word accuracy after adaptation follows a similar performance pattern as the Levenshtein distance after adaptation. However the sentence correctness measured by HTK does not. The Levenshtein distance accounts for edit operations of each word in the sentence as well as the spaces and context (dependence on neighbouring sentence constituent) within the sentence.

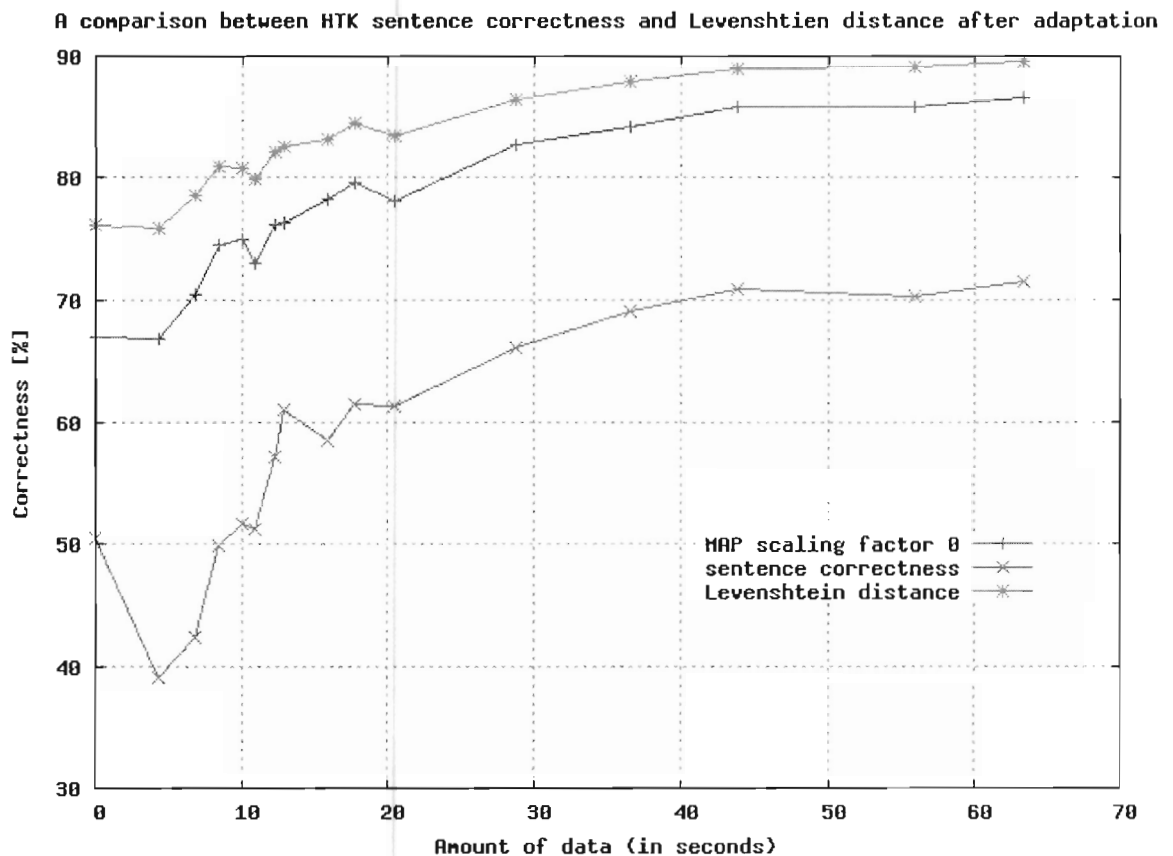


Figure 5-11: A comparison of sentence recognition evaluation methods after MAP adaptation

Figure 5-11 illustrates the performance comparison for HTK sentence correctness measurement and the Levenshtein distance after MAP adaptation where the scaling factor is 0. The illustration therefore includes sentence correctness, word accuracy and the Levenshtein distance after adaptation. The initial values for each of the measured performances are the same as the previous ones discussed in Figure 5-9. As in other previous tests illustrated in figures in this sub-section, a performance increase is generally seen as more adaptation data is used. Generally when adaptation data of less than 20 seconds are used, then there are some considerable fluctuations in performance. If more than 20 seconds of adaptation data are used then there are fewer fluctuations in performance. In the comparison, the spaces and context of the words are maintained for most of the adaptation range. This means that the order in which words should be according to the reference transcriptions is maintained by the system and the Levenshtein distance indicates this. The word accuracy shown is in agreement with this. The sentence correctness does not largely follow a similar pattern of performance as the other methods. Generally it is not expected to follow such a pattern due to the fact that it penalizes edit operations by declaring a word which has been affected by an edit operation incorrect.

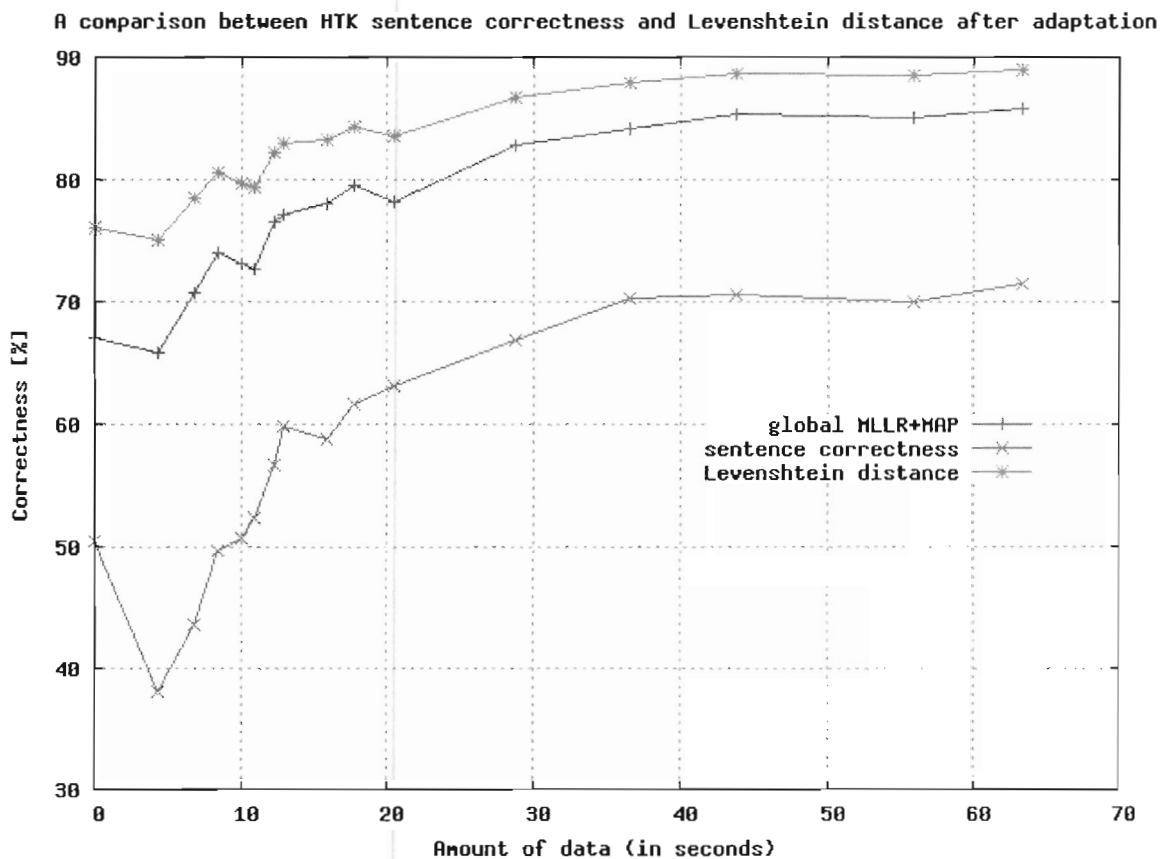


Figure 5-12: A comparison of sentence recognition evaluation methods.

Figure 5-12 illustrates the comparison of HTK sentence correctness and the Levenshtein distance after adapting with MLLR+MAP adaptation. As in Figure 5-9, the same initial SI performance values (when adaptation data used is 0) are assumed since adaptation is done on the initial SI systems. The MLLR+MAP adaptation used here first adapts with the global MLLR algorithm and generates an informative prior (adapted HMM parameters). Then the MAP algorithm with a weighting factor of 0 is used to adapt. As with previous comparisons, a similar performance trend is seen. There is a general performance increase of the word accuracy after adaptation. With all the measured parameters, there are performance fluctuations when the amounts of adaptation data are less than 20 seconds. However when the amounts are greater than 20 seconds, there are fewer fluctuations. Therefore initially the word accuracy after adaptation decreases when less than 7 seconds of adaptation data are used for adaptation. However, this increases again when more adaptation data are used. As with the fluctuation patterns from previous illustrations, the word accuracy decreases between 9-12 seconds of data, while it increases again up until 18 seconds of adaptation data are used. Between 18 and 21 seconds of adaptation data, the performance then generally increases throughout the adaptation range. The sentence correctness generally decreases in the first 7 seconds adaptation and also between 14 and 16 seconds of data. However the sentence correctness generally increases throughout the adaptation range with the exception of the slight performance decrease between 45 and 55 seconds of adaptation data. The Levenshtein distance generally follows the same pattern as the word accuracy. As a result distance decreases in the Levenshtein distance in the same intervals that the word accuracy also decreases. Consistent with previous descriptions, the Levenshtein distance scores a higher score than the word accuracy.

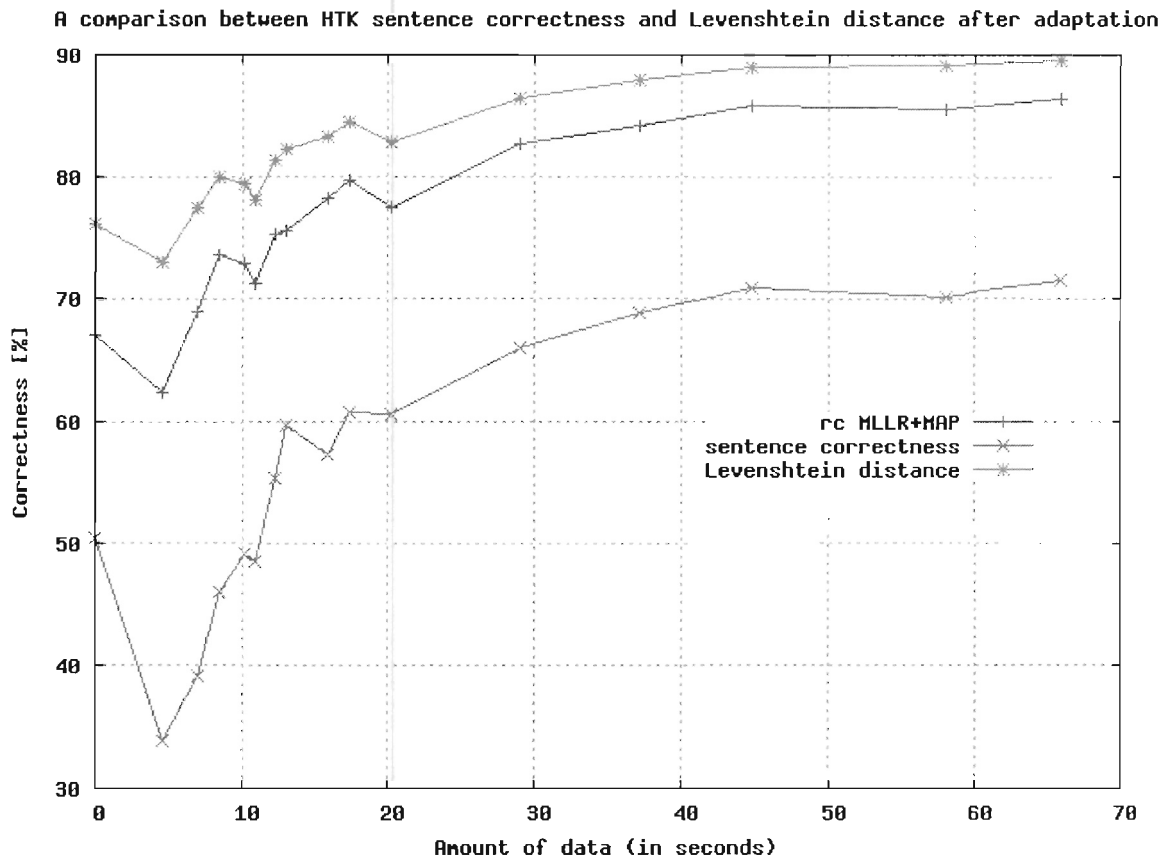


Figure 5-13: A comparison of sentence evaluation methods

Figure 5-13 illustrates the comparison between HTK sentence correctness measurement and the Levenshtein distance after MLLR+MAP adaptation. The difference between this test and the one illustrated in Figure 5-12 is that the MLLR adaptation is now done using regression classes. As a result, the word accuracy of this system is closer to 90% after 60 seconds of data than is the system depicted by Figure 5-12. However, it is important to note that the word accuracy generally increases. When it decreases, it decreases much in the same intervals as the system depicted in Figure 5-12. The sentence correctness measured by HTK also shows similar patterns as in Figure 5-12 and as a result its performances decrease at the described intervals and generally increase as described for the figure. Overall, in much the same way as with previous test, more fluctuations are observed when the amount of adaptation data used is less than 20 seconds. Fewer fluctuations are observed when more than 20 seconds of data are used.

5.3.6 A comparison of the amounts of data used

This sub-section describes the amount of adaptation data and training data used towards reducing the WER of the systems. In order to build a good speech recognition system, a large amount of data is needed in order to train the models of that system. Such a system can be trained to be SI or SD. SI systems are very common in public applications. However they do not perform as well as SD systems for an intended speaker and have a higher WER than SD systems. This situation is applicable to the public access IVR server since it is built as an SI system. The performance of the SI system can be improved either by using more speech data in training or by using speaker adaptation. In this study speaker adaptation was used for improving the systems. Table 5-5 below illustrates the comparison between the training data used in training and adaptation data used in adaptation.

Table 5-5: Amounts of speech data used

Speech data	Amount in seconds
Training data	1400.5
Adaptation data	63.36

As seen above, the amount of training data used on average for each SI system built is about 1400.5 seconds. The amount of data used in order to adapt such a system is on average 63.36 seconds. Adaptation data constitutes a portion equivalent to about 4.5% of the training data used. This shows that the adaptation data that is used for adaptation is significantly less than the amount of training data used for training. If this data is acquired from the speaker using a telephone line, then acquiring training data will be significantly more expensive than acquiring adaptation data.

It is possible to look at the amount of speech data present for every percentage WER in that particular system. This ratio of speech data to percentage WER shows the effectiveness of the speech data within the speech recognition system. Speech data will be used in order to achieve a certain level of accuracy within a system. However, this level of accuracy is also indicative of the WER yield of the data. This WER yield can be expressed in a ratio and can show the effectiveness of the speech data used in the system. It is important to calculate the ratio for the

particular contribution which the data is credited for. Therefore for the optimized SI system the ratio can be calculated as follows:

$$1400.5 \text{ sec} / (100\% - 32.94\%) = 20.89 \text{ sec} / \text{WER}\% \quad (5-1)$$

where 1400.5 seconds are the amount of speech data used to train the SI system and 32.94% is the WER of that SI system credited to the 1400.5 seconds of data. In this case, the optimization of the SI system is not taken into account in the calculation since the same amount of data is used in the optimization. This figure can be calculated for the supervised adaptation algorithms. In this study this figure is calculated after 60 seconds of adaptation data is used. The following are calculated:

$$63.32 \text{ sec} / (32.94\% - 25.58\%) = 8.6 \text{ sec} / \text{WER}\% \quad (5-2)$$

$$63.32 \text{ sec} / (32.94\% - 16.87\%) = 3.9 \text{ sec} / \text{WER}\% \quad (5-3)$$

$$63.32 \text{ sec} / (32.94\% - 13.47\%) = 3.25 \text{ sec} / \text{WER}\% \quad (5-4)$$

$$63.32 \text{ sec} / (32.94\% - 14.21\%) = 3.4 \text{ sec} / \text{WER}\% \quad (5-5)$$

where equation 5-2 represents the value computed for global MLLR adaptation, equation 5-3 represents regression class MLLR adaptation, equation 5-4 represents MLLR+MAP adaptation and 5-5 represents MAP adaptation. In comparing the values computed for adaptation and the value computed for training, it can be seen that the adaptation values are smaller than the training value. The values show the comparison of the ratio between the amounts of data used for WER reduction obtained in that system. More importantly it shows how much data is used in order to effect a WER reduction in this study. The calculated value for the training data is far higher than the value calculated for the adaptation algorithms. This means that more training data has been used for the reduction of the WER compared to adaptation data. Lower values therefore mean that that method of WER reduction is more effective than higher values. In this case, speaker adaptation proves to contribute more effectively than training data contributes to WER reduction.

It is important to realize that these calculations do not imply that speaker adaptation data is more special than training data. They show that the amount of data combined with the performance improving technique can have high improvement efficiency.

5.4 Discussion

This chapter has described the results, which have been obtained in this study. Firstly, the description of the speech database, which was used for this study, was given in sub-section 5.1. The design of the system, which is used for this study, was described in sub-section 5.2, while the settings, which are used for the study, are given. The description of the verification of the algorithms was given in sub-section 5.3.1 and the results from the evaluation of SI systems were given in sub-section 5.3.2. A description of system parameter variation was given in sub-section 5.3.3, while results from unsupervised speaker adaptation were given in sub-section 5.3.4. Sub-section 5.3.5 gave a comparison between the HTK sentence correctness measuring method and the Levenshtein edit distance, which is implemented for this study. Finally sub-section 5.3.6 gave the analysis of the amounts of data used for the study.

The verification of the algorithms shows that the algorithms work as expected. The key aspects of the verification were, SD WER, SI WER as well as the speaker adaptation WER. In comparison to other studies, certain expectations in results prevailed. Therefore this study also had similar expectations. Expectations from this study were that the SD WER would be lower than the SI WER as observed. The other expectation was that when applying adaptation to the SI system, the WER would then decrease and the WER would approach the SD WER. From these studies, it is known that SD systems work well for the intended speaker. SD systems generally model the intended speaker ideally, because they are trained only on that speaker. Conversely, a SD system does not model another speaker ideally, because they are not trained on that speaker. The system does not work well for the speaker they were not trained on. In order to test this, several tests were conducted where simple SD systems were used. A system was trained on one speaker, hence SD and it was tested with that speaker. The result was noted for future comparison. A different speaker then tests this system and this result is noted for future comparison. From expectation, the first result should be better than the second result. The results of these tests actually concur with previous studies. Speaker adaptation is used in order to improve the second result and as such, an improvement is seen. As a result, the tests show that speaker adaptation works. Next, the performance of speaker adaptation on SI systems was evaluated. From previous studies it is known that the performance of SI systems on a new speaker not used in its training is not as good as the performance of an SD system used on the speaker it was trained on. This chapter has also

shown that this is the case. The WER for an SI system being tested by a speaker is higher than for an SD system for the same speaker. The tests conducted confirm the expectations.

In the subsequent sub-section, the performance of SI systems was evaluated. For this evaluation, speaker adaptation was used to improve the systems. This sub-section builds on the expectation from the previous test. Generally the algorithms performed differently to each other. Typically MAP and MLLR+MAP performed well when larger amounts of adaptation data are used. However, when smaller amounts of adaptation are used, the performances didn't differ by much. It is important to note is that when 1 Gaussian mixture component is used for each state, the adaptation tended to degrade significantly when small amounts of adaptation data was used. However, the performance improved significantly when greater amounts of data were used.

In optimising the systems, mixture incrementing was conducted where more than 1 Gaussian mixture components were used per state as conducted in the chapter. This yielded performance increases over the original 1 Gaussian mixture component per state system that was built initially. The mixture incremented system was called the optimized SI system, while the original system was called the original SI system. Generally, using more mixture components per state models the new speaker better than the in the original system. The adaptation algorithm settings were also varied in order to gain performance. For MLLR, the number of regression classes was increased until a point the HTK maximum was reached. In general increasing the number of regression classes yields performance increases, however, it may also degrade the performance.

When varying the scaling factor for MAP, it was also possible to increase the performance as well. This was done over a range from 0 to 100. Generally in starting from 0 and increasing the scaling factor over the range, the WER increased. Since this increase in the general WER was the trend, it was not necessary to evaluate further than the range. Unsupervised adaptation was conducted on the original system. This yielded no initial improvements and actually showed a slight degradation in performance. However, performance was observed when performing unsupervised adaptation on mixture incremented systems.

HTK sentence correctness measurement was evaluated against word accuracy and the implemented Levenshtein edit distance. Generally this showed that sentence correctness in HTK didn't follow a similar performance pattern as word accuracy in most of the tests. However, the Levenshtein distance followed a similar patter. The Levenshtein distance penalizes a sentence by

the distance by which it differs from the reference sentence. The sentence correctness measurement in HTK penalizes the sentence if there are any insertions or deletions. As a result, the Levenshtein distance scores the sentence correctness more accurately than the HTK sentence correctness measurement method.

Lastly, a comparison of the speech data used and the WER reduction effectiveness was done. This showed that more data is used in training towards WER reduction. However, for speaker adaptation, it showed that less data was used towards WER reduction thereby making for an effective WER reduction strategy. This makes speaker adaptation a suitably method for the public access IVR server. It also makes it an attractive option due to the time savings.

Chapter 6

6 Conclusions

The main aim of this study was to evaluate the suitability of speaker adaptation algorithms for the public access IVR server. Previous studies have looked at speaker adaptation for different contexts like large vocabulary speech recognition, news transcription, speaker verification, and telephone conversation transcriptions [14], [29], [52], [76][77]. In this study it has been looked at in the context of IVR servers. MLLR, MAP and MLLR+MAP were implemented in the evaluation to determine their suitability. Speaker adaptation is an attempt to improve speech recognition performance of such IVR servers. Since the study's focus is on speaker adaptation, other speech recognition improvement techniques were not implemented.

140 seconds of adaptation data required for SD System error reduction

When SD systems are tested on the speaker they were trained on, they give a good performance of 6% WER on average. When tested on a speaker it wasn't trained on, they give a performance of 44% WER on average. However, after adapting the systems with about 140 seconds of data from the speaker who wasn't used for training:

- global MLLR adaptation gives a performance of 23% WER on average;
- RC MLLR adaptation gives a performance of 18% WER on average;
- MAP adaptation gives a performance of 17% WER on average; and
- MLLR+MAP adaptation with global MLLR applied first, followed by MAP gives a performance of 16% WER on average.

Consistent performance with all algorithms was achieved after using at least 30 seconds of adaptation data.

60 seconds of adaptation data sufficient for substantially improving SI systems

When the SI systems are tested on a speaker they were not trained with, they give a performance of 41.6% WER on average. After using 60 seconds of adaptation data from the speaker:

- Global MLLR adaptation gives a performance of 31.9% WER on average;
- RC MLLR adaptation gives a performance of 29.7 % WER on average;
- MAP and MLLR+MAP adaptation both give a performance of 23.7 % WER on average.

Consistent performance was achieved after using at least 20 seconds of adaptation data.

Improved performance of SI systems achieved through HTK tools

Starting with initial SI systems that give a performance of 41.6% WER on average. Using HTK Tools, the number of Gaussian mixture components per state was gradually increased by one. The following results were achieved:

- The best performance was achieved with 9 Gaussian mixture components per state, with 36.1% WER on average.
- After optimizing the number of Gaussian mixture components per for each system the performance improved to 32.9% WER on average. This formed a basis for optimized speaker adaptation tests.

Optimized speaker adaptation involved optimizing the speaker adaptation algorithm settings. The following was achieved:

- Global MLLR adaptation gave a performance of 25.6% WER on average.
- RC MLLR adaptation using 256 regression classes gave a performance of 16.8% WER on average.
- MAP adaptation using a scaling factor of 0 gave a performance of 13.5% WER on average.
- MLLR+MAP adaptation with optimal MLLR and MAP settings gave a performance of 14.2% WER on average.

Unsupervised versus supervised

Only unsupervised MLLR adaptation could be tested. When unsupervised MLLR adaptation using 32 regression classes is applied to the initial SI systems, a performance of 41.9% WER on average is achieved. However, when adapting the systems with optimized Gaussian mixture components per state, the performance obtained is 30.4% WER on average. When 256 regression classes are used for adapting the optimized systems, a performance of 30.3% WER is obtained.

Throughout these tests these performance figures have been worse than figures for supervised MLLR adaptation.

Levenshtein edit distance versus HTK sentence correctness

There is a basic structure which contains “*a clear structure that binds the specific pieces of information together*” [99]. The Levenshtein distance has been used in previous work as a loss function for minimizing error rates [89], but it has been originally designed for evaluating the performance of recognition output by [91]. While many performance evaluating criteria penalize incorrect words heavily in an utterance [48], this distance in [89] only penalizes discriminately the key differences between utterances. When measuring the distance after optimizing the SI system Gaussian mixture components per state, the performance is 64% accuracy on average. The Levenshtein distance is 74% on average. The relevant sentence correctness measured with HTK is 41%. This trend is exhibited throughout the tests with the adaptation algorithms. The Levenshtein distance measured is more closely related to the word accuracy than with sentence correctness. The Levenshtein distance is also a more predictable means to measure sentence correctness than the HTK’s sentence correctness measurement.

Based on the conclusions the following future work is recommended:

- Speaker adaptation was evaluated on the experimental database. Various settings were used in order to improve the performance. Other studies which covered a different scope than this study have used methods like Vocal Tract Normalization and Adaptive training. It would be interesting to see how much more improvements can be yielded through such techniques.
- While mixture incrementing was studied, it was applied to all the models. Generally it yielded improved performance at a certain point where the new speaker was modelled well. However, it would be of interest to find out what happens if say non-speech models are incremented separate from speech models. A segmentation-based method of incrementing would be also an interesting prospect towards performance improvement.
- Levenshtein distance was evaluated in this study. While WER is based on it, the sentence performance was not based on it and as result this study implemented it for sentence performance. Such a distance is sometimes used as a loss function in risk criterion management in speech recognition. In this study, it has proven to be a useful method for evaluating sentences than the current method. It would be interesting to implement this

system and test it for an IVR task. In fact this method would be interesting for word spotting systems as well.

Bibliography

- [1] L. Rabiner and B. Juang, "Fundamentals of speech recognition", Prentice Hall, 1993.
- [2] K.J. Power, "Speech Technology for Telecommunications", Chapman Hall, New York, 1998.
- [3] C.J. Leggetter and P.C. Woodland, "Maximum Likelihood Linear Regression for speaker adaptation of continuous density hidden Markov models." *Computer Speech and Language*, Vol. 9, pp. 171-185, 1995.
- [4] P.C. Woodland and S.J. Young, "The HTK Tied-State Continuous Speech Recogniser", In: *Proc. Eurospeech '93*, pp. 2207-2210, Berlin, 1993.
- [5] K. Weber, S. Ikbal, S. Bengio and H. Bourlard, "Robust speech recognition and feature extraction using HMM2", *Computer Speech and Language*, Vol. 17, pp 195-211, 2003.
- [6] J. McDonough, T. Schaaf and A. Waibel, "Speaker adaptation with all-pass transforms." *Speech Communication*, Vol. 42, pp. 75-91, 2003.
- [7] A. Gunawadarna, and W. Byrne, "Discounted Likelihood Linear Regression for rapid speaker adaptation", *Computer Speech and Language*, Vol. 15, pp. 15-38, 2001.
- [8] C.M Rebman, M.W.Aiken and G.C. Cegielski, "Speech Recognition in the human-computer interface", *Information & Management*, Vol. 40, pp. 509-519, 2002.
- [9] A.J. Rodger and P.C. Pendharkar, "A field study of the impact of gender and user's technical experience on the performance of voice-activated medical tracking application", *International Journal of Human-Computer Studies*, Vol. 60, pp. 529-544, 2003.
- [10] C. Niewoudt and E.C. Botha, "Cross-language use of acoustic information for automatic speech recognition", *Speech Communication*, Vol. 38, pp. 101-113, 2002.
- [11] N. Mahlanyane and D. Mashao, "Using a low-bit Speech Enhancement Adaptive Filter to improve GMS Speech Recognition Performance," In: *Proc. PRASA 2003*, pp. 75-79, Langebaan, 2003.
- [12] K. Ohtsuki., et. al., "Japanese large-vocabulary continuous-speech recognition using a newspaper corpus and broadcast news", *Speech Communication*, Vol. 28, pp. 155-166, 1998.

- [13] A. Sankar and A. Kannan, "A comprehensive study of task-specific adaptation of speech recognition models", *Speech Communication*, Vol. 42, pp.125-139, 2004.
- [14] J. Chien and H. Wang, "Telephone speech recognition based on Bayesian adaptation of hidden Markov models", *Speech Communication*, Vol. 22, pp. 369-384, 1997.
- [15] A. Hayes, "Interactive Voice Response and Kiosk Technology Research Paper", Reform and Review Division, NSW Premier's Department, ISBN: 0 7313 30 59 5, 1999.
- [16] "AnandTech: Speech Recognition –Ready for Prime Time", www.anandtech.com, Last accessed: 01 May 2006.
- [17] O.P. Noah, "Design and Evaluation of a TTS/ASR system", BSc Thesis, University of Cape Town, 2003.
- [18] Statistics South Africa, <http://www.statssa.gov.za/>, Last accessed: 08 August 2005.
- [19] Linguistic Data Consortium, www.ldc.upenn.edu, 20 October 2005.
- [20] Speech Recognition Milestones, <http://www.zdnet.com/pcmag/features/speech/sb1.html>, Last accessed: 26 July 2005,
- [21] Knight Rider, www.knightrideronline.com , Last accessed:10 October 2005.
- [22] Daimler Chrysler, www.daimlerchrysler.com, Last accessed: 02 November 2005.
- [23] F. Jelinek, *Statistical Methods for Speech Recognition*, The MIT Press, Massachusetts, 1999.
- [24] Festival, <http://www.cstr.ed.ac.uk/projects/festival/>, Last accessed: 24 February 2006.
- [25] A. Rosenberg, O. Siohan and S. Parthasarathy, "Small group speaker identification with common password phrases", *Speech Communication*, Vol. 31, pp. 131-140, 2000.
- [26] S. Lung, "Adaptive fuzzy wavelet algorithm for text-independent speaker recognition", *Pattern Recognition*, Vol 37, pp 2095-2096, 2004.
- [27] A Space Odyssey, www.kubrick2001.com, Last accessed: 24 February 2006.
- [28] M. Westphal, "The use of Cepstral Means in conversational speech recognition.", Eurospeech '93, Rhodes, 1993.
- [29] P.C. Woodland and D. Pye, "Experiments in speaker normalisation and adaptation for large vocabulary speech recognition", In *Proc: Int'l Conf. on Acoustic, Speech and Signal Processing*, pp. 1047-1050, 1997.
- [30] Microsoft Corporation, www.microsoft.com, Last Accessed: 04 November 2004.
- [31] Dragon Systems, www.dragontalk.com, Last accessed: 01 August 2004.

- [32] IBM, www.ibm.com, last accessed: 01 August 2004.
- [33] German Car Fans, www.germancarfans.com, Last accessed: 01 August 2004.
- [34] F. Stremler, *Introduction to Communication Systems*, Addison Wesley, New York, 1990.
- [35] S. Molao, D. Keysers and H. Ney, "Matching training and test distributions for robust speech recognition", *Speech Communication*, Vol. 41, pp. 579-601, 2003.
- [36] D. Kim and N. Kim, "Maximum a posteriori adaptation of HMM parameters based on speaker space projection", *Speech Communication*, Vol. 42, pp. 59-73, 2004.
- [37] C. Mokbel, et al., "Towards improving ASR robustness for PSN and GSN telephone applications", *Speech Communication*, Vol. 23, pp. 141-159, 1997.
- [38] M. Vihola, "Dissimilarity Measures for Hidden Markov Models and Their Application in Multilingual Speech Recognition", Master of Science Thesis, Tampere University of Technology, 2001.
- [39] Speech recognition and information retrieval, <http://www.nist.gov/speech/publications/darpa97/html/witbroc1/witbroc1.htm>, last visited: 18 February 2005.
- [40] J. Foote, "An overview of Audio Information Retrieval, Multimedia Systems", vol. 7, no. 1, pp. 2-11, 1999.
- [41] DARPA, "the Darpa TIMIT acoustic-phonetic continuous speech corpus, training and testing data", 1990.
- [42] P.C. Woodland, "The development of the HTK Broadcast News transcription system: An overview", *Speech Communication*, Vol. 37, pp 47-67, 2002.
- [43] A. Zafar et. al, "A simple error classification system for understanding sources of error in automatic speech recognition and human transcription", *International Journal of Medical Informatics*, Vol. 73, pp. 719-730, 2004.
- [44] J. Hakkinen, J. Suantausta, S. Riis and K. Jensen, "Assessing text-to-phoneme mapping strategies in speaker independent isolated word recognition", *Speech Communication*, Vol. 41, pp 451-467, 2003.
- [45] O. Kwon and T. Lee, "Phoneme recognition using ICA-based feature extraction and transformation", *Signal Processing*, Vol. 85, pp. 1005 –1019, 2004.
- [46] K. Palomaki, G. Brown and J. Barker, "Techniques for handling convolutional distortion with 'missing data' automatic speech recognition", *Speech Communication*, Vol. 43, pp. 123-142, 2004.

- [47] J. Barker, M. Cooke and D. Ellis, "Decoding speech in the presence of other sources", *Speech Communication*, Vol. 45, pp. 5-25, 2004.
- [48] S. Young et. al, *The HTK Book*, 2002, Internet Publication: [www//htk.eng.cam.ac.uk](http://htk.eng.cam.ac.uk), Last Visited: 07 March 2005.
- [49] O. Noah and D. Mashao, "Evaluation of speaker adaptation algorithms", *In proc: ICT 2005*, 2005.
- [50] L. Wang, X. Wang and J. Feng, "Intrapersonal subspace analysis with application to adaptive Bayesian face recognition", *Pattern Recognition*, Vol. 38, pp. 617-621, 2005.
- [51] D. Kim and N. Kim, "Rapid online adaptation using speaker space model evolution", *Speech communication*, Vol. 42, pp. 467-478, 2004.
- [52] J.E Hamaker, "MLLR: A speaker adaptation technique for LVCSR", *Lecture for a course at ISIP-Institute for Signal and Information processing*, Department of Electrical and Computer Engineering, Mississippi State University, 1999.
- [53] G. Forbes, "The Automatic Detection of Patterns in People's Movements.", MSc Thesis, University of Cape Town, 2002.
- [54] S. Lupembe , "Improving performance of GSM-Based Speech Recognizer.", MSc Thesis, University of Cape Town, 2004.
- [55] Speech Recognition using Neural Networks at the Center for Spoken Language Understanding, http://cslu.cse.ogi.edu/tutordemos/nnet_recog/recog.html, last accessed: 21 June 2005.
- [56] A. Hagen and A. Morris, "Recent advances in multi-stream HMM/ANN approach to noise robust ASR", *Computer speech & Language*, Vol. 19, pp. 3-30, 2004.
- [57] L.E. Baum and T. Petrie, "Statistical inference for probabilistic functions of finite state Markov chains," *Ann.Math.Stat.*, pp. 1554-1563, 1966.
- [58] L.E Baum, J.A. Egon, "An equality with applications to statistical estimation for probabilistic functions of a Markov process and to a model for ecology", *Bull.Amer.Meteorol.Soc.*, Vol. 73, pp. 360-363, 1967.
- [59] L.E. Baum and G.R. Sell, "Growth functions for transformations on manifolds," *Pac.J.Math*, Vol. 27, pp. 211-227, 1968.
- [60] L.E. Baum, T. Petrie, G. Soules and N. Weiss, "A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains," *Ann.Math.Stat*, Vol. 41, pp. 164-171, 1970.

- [61] L.E. Baum, "An inequality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes," *Inequalities*, Vol.3, pp. 1-8, 1972.
- [62] J.K. Baker, "The dragon system –An overview." *IEEE Trans. Acoustics, Speech, Signal Proc.*, ASSP-23. pp. 24-29, 1975.
- [63] F. Jelinek, "A fast sequential decoding algorithm using a stack," *IBM J.Res.Develop.*, Vol. 13, pp.675-685, 1969.
- [64] G. Forney, "The Viterbi Algorithm," in *Proc. IEEE*, vol. 61, pp.268-278, 1973.
- [65] S. Levinson, L. Rabiner and S. Sondhi, "An introduction to the Application of the Theory of Probabilistic Functions of a Markov Process to Automatic Speech Recognition," *Bell System Tech.J.*, Vol. 62, pp. 1035-1074, 1983.
- [66] H. Hermansky, "Perceptual linear predictive (PLP) analysis of speech," *Journ. Acoustic Soc. America*, vol. 8, no.4, pp. 1738 -1752, 1990.
- [67] B. Kingsbury, N. Morgan and S. Greenberg, "Robust speech recognition using the modulation spectrum," *Speech communication*, vol. 25, pp. 117-132, 1998.
- [68] J. Picone, "Signal Modelling Techniques in Speech Recognition," *In Proc IEEE*, Vol. 81, no. 9, pp. 1215-1247, 1993.
- [69] A. Oppenheim and R. Schafer, "Digital Signal Processing," Engelwood Cliffs, NJ, Prentice-Hall, 1975.
- [70] R. Marois and J. Ivanoff, "Capacity limits of information processing in the brain," *Trends in Cognitive Sciences*, vol. 9, pp. 296-305, 2005.
- [71] S. Lahtinen and J. Peltonen, "Adding speech recognition support to UMP tools," In *Proc. Journal of Visual Languages and Computing*, vol. 16, no. 2, pp. 85-118, 2004.
- [72] Intellecta Voice and Mobile, www.intelleca.co.za, Last accessed: 9 July 2005.
- [73] Structured Query Language, databases.about.com/od/sql, Last accessed: 9 July 2005.
- [74] MySQL Database, www.mysql.com, Last accessed: 9 July 2005.
- [75] Microsoft SQL Server, www.microsoft.com/sql/default.asp, Last accessed: 11 July 2005.
- [76] Z. Wang, T. Schultz and A. Waibel, "Comparison of acoustic model adaptation on non-native speech", In *Proc: ICASSP 2003*, pp. 540 – 543.
- [77] A. Surendran and C. Lee, "Transformation-based Bayesian prediction for adaptation of HMMs", *Speech Communication*, vol.24, pp.159 -174, 2001.

- [78] A. Jain, R. Duin and J. Mao, "Statistical pattern recognition: A review," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, pp. 4-37, 2000.
- [79] M. Skosan, "Histogram Equalization for Robust Text-independent Speaker Verification in Telephone Environments.", MSc Thesis, University of Cape Town, 2005.
- [80] B.S. Atal, "Speech technology in 2001: New research directions", In proc: Natl. Acad. Sci. USA, vol. 92, pp. 10046-10051, 2001.
- [81] S. Foo, C. Yeo, S. Hui and D. Xue, "Design of an Internet fax and voice gateway," *Microprocessors and Microsystems*, vol. 22, pp. 395 – 402, 1998.
- [82] D. Johnston, et al. "Current and experimental applications of speech technology for telecom services in Europe," *Speech communication*, vol. 23, pp. 5-16, 1997.
- [83] C. Lee et al, "On natural call routing", *Speech communication*, vol. 31, pp. 309-320, 2000.
- [84] A. Nierenberg, et al, "Suicide risk management for the sequenced treatment alternatives to relieve depression study: applied NIHM guidelines," *Journal of Psychiatric Research*, vol. 38, pp. 583-589, 2004.
- [85] Indian Language Speech Recognition,
<http://www.research.ibm.com/irl/projects/speech.html>, [Last accessed: 25 July 2005].
- [86] C. Mokbel et al., "Towards improving ASR robustness for PSN and GSM telephone applications," *Speech Communications*, vol. 23, pp. 141-159, 1997.
- [87] K. Turner, "Analysing interactive voice services," *Computer Networks*, vol. 45, pp. 665-685, 2004.
- [88] S. Lahtinen and J. Peltonen, "Adding speech recognition support to UML tools," *Journal of Visual Languages and Computing*, vol. 16, pp. 85-118, 2005.
- [89] J. Kaiser, B. Horvat and Z. Kacic, "Overall risk criterion estimation of hidden Markov model parameters," *Speech Communications*, vol. 38, pp. 383-398, 2005.
- [90] R.O. Duda and P.E. Hart, "Pattern Recognition and Scene Analysis", Wiley & Sons, New York, 1973.
- [91] V. Levenshtein, "Binary codes capable of correcting deletions, insertions and reversals", *Soviet Phys. Dokl.* 10. pp. 707-710, 1966.
- [92] J. Bellegarda, "Unsupervised, language-independent grapheme-to-phoneme conversion by latent technology", *Speech Communications* vol.46, pp. 140-152, 2005.

- [93] F. Nicolas and E. Rivals, "Hardness results for the center and median string problems under the weighted and unweighted edit distances", *Journal of Discrete Algorithms* vol. 3, pp. 390 – 415, 2005.
- [94] H. Qiu and E. Hancock, "Graph matching and clustering using spectral partitions", *Pattern Recognition* vol. 39, pp.22-34, 2006.
- [95] H. Hyrro, "Bit-parallel approximate string matching algorithms with transposition", *Journal of Discrete Algorithms* vol. 3, pp. 215-229, 2005.
- [96] H. Zhao and S. Ram, "Entity identification for heterogeneous database integration –a multiple classifier system approach and empirical evaluation", *Information Systems* vol. 30, pp. 119-132, 2005.
- [97] D. Palmer and M. Ostendorf, "Improving out-of-vocabulary name resolution", *Computer Speech and Language* vol. 19, pp. 107 – 128, 2005.
- [98] Speech-Enabled Interactive Voice Response Systems. International Engineering Consortium. [Online]. Available: http://www.iec.org/online/tutorials/speech_enabled
- [99] N. Gupta et.al., "The AT&T Spoken Language Understanding System", *IEEE Transactions on Audio, Speech and Language Processing* vol 15, pp. 213 – 222, 2006.
- [100] A. Batliner, et.al., "How to find trouble in communication", *Speech Communication*, vol. 40, pp. 117-143, 2003.
- [101] UCLA Electrical Engineering Department, <http://www.ee.ucla.edu/>, Last accessed: 11 August 2006.
- [102] interact Center, <http://www.is.cs.cmu.edu/11-751/2003/hw/hw2.html>, Last accessed: 11 August 2006.
- [103] Ster Kinekor, www.sterkinckor.co.za, Last accessed: 21 August 2006.
- [104] A.P. Dempster, A.M. Laird, D.B. Rubin, "Maximum likelihood for incomplete data", *Journal of Royal Statistical Society, Series B*, Vol. 39, pp. 1 – 38, 1977.
- [105] C. Lee, J. Gauvain, "Speaker adaptation based on MAP estimation of HMM parameters", *IEEE International Conference on Acoustics Speech and Signal Processing*, Vol. 2, pp. 558 – 561, 1993.