AGH UNIVERSITY OF SCIENCE AND TECHNOLOGY
KRAKOW

Faculty of Electrical Engineering, Automatics, Computer Science and Electronics

David Sierra Rodríguez

Text-Independent Speaker Identification

Master Science Thesis

Tutor
Prof. dr hab. inż. Mariusz Ziółko

Supervisor
Dr. Rafael González Ayestarán

Kraków 2008

INDEX

1 INTRODUCTION

The latest technology improvements in such fields as banking, communications and networking require the latest advances in security. In the last years a new kind of security has risen among the others as it is printed in subjects' bodies, it is impossible to lose, almost impossible to duplicate and univocally designates a person. These are the biometric keys. These security systems exploit some human characteristics that are different and unique in each person such as fingerprints, DNA chains, face shape and voice. In this thesis, human voice will be the security key to be used. A system that exploits the individuality of each speaker's voice and identifies or verifies speaker's identity by machine is known as an Automatic Speaker Recogniser (ASR).

A single sentence pronounced by a speaker like "Hello, how're you?" conveys a large amount of data our brain is able to analyze. Generally, one only tries to extract the message or data information from it, this is, what is the meaning of the sentence? However, a listener can get to know many other things about the speaker like his mood, age, gender, heath state, and etcetera just by listening to him. In addition, it is possible that this is not the first time the listener has heard the speaker's voice and thus he is able to recognize the speaker. In this case, the listener has performed a speaker recognition operation.

In such way, ASRs work in the same way as human being. Thus, a parallelism with the way humans recognise their fellows will be done in order to explain the procedure carried out by an ASR. The first step that allows the listener to recognise the speaker's voice is meeting this person and listening to his/her voice. Then, a memory of that voice can be kept in the brain. Now, if the speaker's voice is heard again, the listener will find it familiar and can estimate whose voice it is. In ASRs the overall procedure is quite similar. First, the system has to get to know the speakers latterly to be indentified. This is done in the enrolment stage where the speaker's identity is certified. In this stage, the speaker's voice is recorded and stored. Then this voice is used in the training stage where some models of each speaker are obtained. These models represent the memory of the speaker's voice. Finally, in the test stage, any speaker can introduce his voice in the system. The system will compare this voice with the models it has stored and decide about the identity of the speaker.

Voice is a phenomenon that is highly dependent on the speaker who produced it. Many physical aspects of speech such as the timber, tone or intensity vary a lot from a speaker to another. The same happens with other linguistic aspects as the single intonation and range of vocabulary or expressions a speaker normally uses. All these properties make voice a very powerful biometric key to be used in security systems since the physical characteristics of speech are easy to measure and compare in comparison to other biometric keys. In addition, the speech signal is quite well-known and has been deeply studied for many years so many powerful algorithms can be found to deal with this kind of signal.

In this introduction, voiceprints advantages and drawbacks will be exposed. In addition they will be compared to other biometric keys. The main applications of this technology will be also discussed. Finally a classification of the ASRs will be done and the general ideas of an Automatic Speaker Identifier (ASI) will be shown. In addition this thesis will describe the theory of speech production and perception and show the applications of this theory to the field of speaker recognition. Furthermore, some text-independent speaker identification systems will be proposed, described and tested. The results of these tests will be presented in the last section of the thesis.

1.1 Advantages and drawbacks of voiceprints and comparison to other biometric keys

Before exposing the particular properties of voiceprints and their potential usefulness, it is necessary to understand the power of biometric keys. How many times one forgets the mobile phone or credit card PIN. In addition, this number can fall in wrong hands and create a very uncomfortable situation. Furthermore, this kind of security is based in encryption algorithms that can be broken and many times have been proved to be inefficient by themselves. Biometric keys come to solve some of these problems as they are "printed" in the human body and therefore can not be lost. Duplication of these keys requires the direct participation of the owner and, depending on the kind of biometric key, this duplication sometimes is impossible. In conclusion, biometric keys are unique, they are difficult to duplicate or forge and they are always with their authorised user [33].

A good biometric key has to match certain requirements. It has to be easy to extract, measure, save and compare. Voiceprints match all these requirements since not a very expensive hardware is needed to perform all these operations. In fact, the only infrastructure needed is a microphone and a PC. A medium-quality microphone is relative cheap hardware if one compares it to a digital face, iris or finger scanner, not mentioning a DNA analyser. In addition, many new banking applications rely on the usage of telephone line. Voiceprints are the only suitable biometric technology able to operate in this environment without needing some additional hardware. To sum up, the front-end infrastructure needed in speaker recognition is very simple and in many cases it is already set up, therefore, making an ASR running generally requires a low investment.

Voice is one of the better-known kinds of signal. For many years, automatic speech recognition (the task of extracting the message contained in one segment of speech by machine) has been and already is the subject of study of many researchers. All this previous research is very useful in the field of speaker recognition since many algorithms initially created for the speech recognition task can also be used in speaker recognition. In addition, many of them can be properly modified to attend the requirements of the speaker recognition task. All these years of studies have lead to find a set of algorithms with a low computational cost and ready to work in real-time applications. Features of voice which are highly speaker-dependent can be extracted and classified using some pattern matching algorithms. Since only a few speaker model parameters have to be stored, memory requirements are not high comparing to 2D or 3D-based biometric keys as iris or face recognition.

Furthermore, there are great advantages in using voiceprints. Voice is a phenomenon that doesn't require the subject to be present. It can be recorded in some place and sent to another almost instantly or even it can be recorded in the destination using a telephone line. Finger-prints images or face-scans require a higher bandwidth.

In addition, almost every user already has a microphone or telephone whereas not many can afford a face or finger scanner.

Lastly, an important property of the voice signal is that it normally transports a message, e.g., a word, a sentence or a number code. Thus, this message can be used to increase the security level by providing the recognition system not only a segment of the subject's speech but also a password included in the message. The system can ask the subject to spell a code number or to say a given word or sentence in order to fight against attacks performed with speech recorded from the subject.

The main drawbacks of speaker recognition are that voice depends on the health condition of the subject, that voice varies along life and that the microphone or the channel used to transmit voice has a very important effect on the speech signal. A cough or a flue can result in a handicap for voice production and consequently alter the natural speech of a person. In this case an ASR will not be able to recognize an authorized speaker. It is also clear that the voice of a person in his childhood is very different from the one in the adult stage, not mentioning the eldership. Some algorithms can deal with this problem by readapting the speaker models along time. Finally, the microphone used in the training stage might be different from the one used in the testing one. This can lead to a mismatch of the models with the speech recorded with the new microphone. However, there are algorithms that are able to reduce the effects of the microphone and the transmission channel over the speech signal.

1.2 Classification of speaker recognition systems

Speaker recognition systems can be classified attending to three different characteristics. Firstly, they can be divided into speaker identification or speaker verification. Secondly, they can be text-dependent or text-independent systems. Lastly, it is possible to classify them into open-set or close-set systems. In this section, all these characteristics will be described.

1.2.1 Speaker verification vs. speaker identification

Speaker recognition systems can work in two different modes, speaker verification or speaker identification. In a speaker verification system, the user will first provide his identity to the user and then the system will check if this identity is correct by analyzing the speaker voice. For example, in the case of an ATM, the user will insert his credit card in the ATM. This credit card contains the identity of the user. If the ATM contains an ASR module, the ATM can check if the card is being used by its owner or by an impostor by asking the user to input some speech. In this case, the reader has to notice that since the user will provide his identity to the system, only a yes/no decision has to be made. The ASR only has to compare the voice input with the model attached to the identity provided by the user. This means that the ASR will perform a single comparison and then a single decision based on the result of that comparison.

On the other hand, in a speaker identification system, the user will not provide his identity to the system. Instead, the user will input his speech and the system will decide which speaker model better matches the speech input. In this case, the system has to perform *N* comparisons, being *N* the number of speakers in the system database. Each comparison will produce a likelihood score so the system will choose the identity attached to the most likely speaker model. This means that the kind of decision will be "*Speaker i*" with *i=1...N.*

It is clear that the speaker identification problem is more complex than speaker verification and therefore, the results achieved in speaker identification systems will be poorer than in speaker verification.

1.2.2 Text-independent vs. text-dependent

The second possible classification method for ASRs conveys the use of a constrained sentence or text for the recognition. In the text-dependent domain, a given sentence or word (more properly called password) will be always used for the training and the recognition. In this case a two-levelled security system is obtained. Firstly, the voice has to be produced by an authorized user and secondly, the user has to provide the proper password.

Another kind of system can be included in the group of text-dependent recognisers. These are the text-prompted systems. In this case, the system will ask the user to pronounce a given code (e.g. a sequence of numbers) in order to prevent a phishing attack. In this case an attacker should have voice registries of the authorised user including all the possible combinations the system may ask for. The larger number of key words used, the more robust system can be built.

On the other hand, text-independent systems do not care about the message contained in the speech input. They only attend to speaker-dependent attributes of speech and do not rely on the sequence of words pronounced by the speaker. The recognition and training will be based on any utterance of speech produced by the speakers.

It is clear that the text-independent case is much more flexible since it is not constrained to any kind of password. The applications of this method are not only in biometric security but in any field which requires the answer to the question "*who said this?*". This thesis is based on this kind of systems because of this reason. Text-independent security systems might look unsafe because of the absence of a password. However, this is not true since speech recognition can be combined with speaker recognition to introduce the password level. In this case, speaker recognition will be clearly separated from speech recognition in opposite to the case of text-dependent recognition. The most pure kind of ASR is a text-independent one from the point of view of this thesis author.

1.2.3 Open-set vs. closed-set

The last method for classifying speaker recognition systems discerns between the possibility any user in the world can try to enter the system or only a set of them will try to do it.

A closed-set recogniser will consider that the user who is trying to access the system belongs to a group of known users. In this case, if a speaker identification system is being used, the decision taken by the system will just be the most likely speaker to the speech input.

However, an open-set system will consider the possibility that the user who is attempting to enter in the system can be unknown. This means that there is not a model associated to this user and therefore the decision taken by the system is that the user is unknown or an impostor. There are three main methods to deal with this problem. The first one is including a model for unknown speakers obtained from a large voice database including several speakers. The second is setting a likelihood threshold so that if any of the speaker models score overpasses this threshold, the decision will be "*unknown user*". The last one is a proper combination of both methods.

It is clear that open-set systems are more complex and match better the conditions in real life where a listener does not have to be familiar with all the voices he hears every day. This is the reason why this thesis will consider this case.

1.3 Applications

Speaker recognition can be applied in many different fields since it is a biometric key security system. However it will be shown in this chapter that this is its only practical application.

Nowadays many banks, shops or other kind of business companies allow their clients to perform different operations using the telephone line. This is very comfortable for the costumer since almost any kind of shopping or bank transaction can be done from any part of the world. However here rises the problem of security. A non-authorised user can supplant an authorised user's identity very easily. In fact, in 2006 alone 8.9 million U.S. adults were victim of identity fraud. Speaker recognition technology can reduce significantly the probability that an impostor supplants a real user identity over the telephone line. In this case, an ASR can decrease the company fraud related costs as well as the fraud prevention costs and therefore increase the costumer satisfaction.

Other applications over the telephone line also include access to different telephonic services as voice mail or operations with interactive voice response systems. Instead of the non-comfortable use of passwords and PIN codes, the speaker's voice is enough to verify his identity.

Speaker recognition can also be successfully applied in cases where the speaker's presence is needed. In this case the voice acts merely substitutes a physical key. An application related to this topic can be seen in VADCs (Voice Activated Device Control). VADCs are devices such as doors, locks, etc which are activated by voice instead of a key or remote control. Access to computers, networks or credit card transactions can also be denied or granted by using voice instead of a user name and a PIN code.

The third main field of application of speaker recognition is not directly related to security systems. Speaker recognition can be used to monitor the behaviour of users or clients. Nowadays many companies use different methods to check the arrival and leaving times of their employees. Speaker recognition can improve these systems so that it is possible to know if the person who is checking in or out at the office really is who that person says he or she is. In addition, if the employees have to use the telephone in their job, an automatic remote log of time and attendance can be generated for each member of the staff. Furthermore, speaker recognition can help in law issues. Sending a person to jail has an elevated cost for the state. However, in the case of minor crimes many people are sentenced to home detention instead of prison incarceration. This leads to save money and also prevents the convict to suffer the adverse atmosphere of a prison (this is especially interesting in the case of youth). Speaker recognition has been used in this case to periodically check that the condemned person is really staying at home. No police visits are required and neither special devices.

The last group of possible applications comprises law enforcement. Speaker recognition has been used in criminal and forensic investigations to analyse recorded speech presented as a proof in different trials. It is also possible to perform real-time speaker identification over telephone to detect terrorists' conversations. Sometimes some conversations are formed by overlapped speech produced by several speakers. Speaker recognition can be used for tracking, spotting and surveillance of these conversations so that it is possible to extract the speech of each speaker.

In conclusion, it can be seen that speaker recognition can be applied in many fields and not all of them regarding security issues. It is a cheap technology that can save a lot of money when applied in certain cases.

1.4 Generic speaker identification

A speaker identification system comprehends two main stages. These are the enrolment and the testing stages. Both stages can be implemented by monolithic blocks as it can be seen in Figs 1.1 and 1.2.

Fig 1.1 represents the basic elements in the enrolment stage. The aim of this stage is to obtain a registry of the voice properties of a given speaker. This registry is called a model. In this stage, the speaker's identity is well-known by other means different from speaker recognition. Therefore, once the speaker's identity has been proved, he is asked to read out a given text or just produce some seconds of speech. This speech signal is then captured by a microphone. According to Fig 1.1, the output of the microphone is entered in an "A/D + pre-processing" block. This block performs the digitalisation of the speech signal, splits the signal into smaller frames and prepares these frames for the next step. The reader can notice that this last block is common to almost every speech processing application.

The second step is called "Feature extraction". In this step, the dimensionality of the speech frames is reduced considerably. This operation is necessary for two reasons. Firstly, the pattern matching block needs to operate with low-dimensionality vectors in order to work in real-time mode. Secondly, the feature extraction block removes unnecessary information which is being carried in the speech frames and emphasises speaker-dependent aspects of speech. Feature extraction is a fundamental block in speaker recognition and a good choice in the algorithm used is highly reflected in the final performance of the system.

The third block is the pattern matching algorithm. This algorithm can work in two different modes called training and testing. In the training mode, sequences of feature vectors produced by a known speaker are used to obtain accurate models of that speaker's single voice. Then, in the testing mode, the models previously created in the enrolment stage are used by the pattern matching algorithm to obtain a similarity measure with feature vectors produced by an a-priory unknown speaker.

The pattern matching algorithm is also a very important block in speaker recognition systems. In fact, it is the "brain" of a speaker recognition system and a good choice will have also a great repercussion in the final performance of the system.

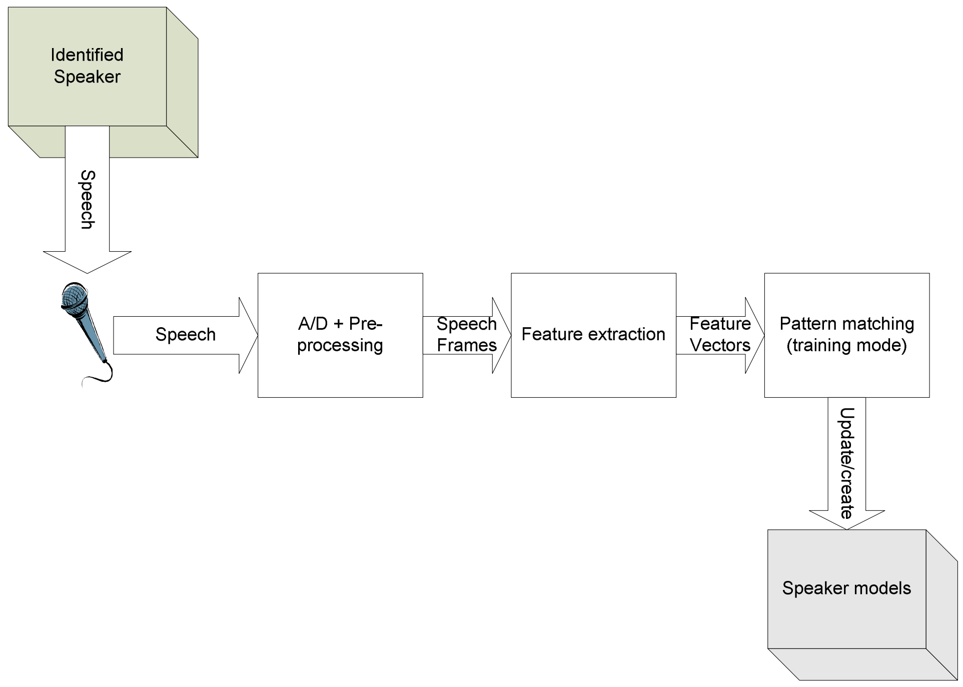


Fig 1.1 Generic identification system (enrolment stage)

The testing stage blocks shown in Fig 1.2 are very similar to the enrolment ones. The main differences are that now, the system is trying to identify a speaker so the speaker who is producing speech is a-priory unknown. The other difference can be seen in the pattern matching algorithm. Now, this algorithm will compare the input feature vectors to the speaker models stored in its database. The result of this comparison is a likelihood score for each model.

Finally, the likelihood score for each model is analysed by a decision-making block. Accordingly to these scores, the decision making algorithm will estimate who is the most likely speaker who produced the speech signal. Furthermore, in the case of an open-set algorithm, an “impostor” or “unknown speaker” decision-kind can be made.

In this chapter, a speaker identification system has been considered. A speaker verification system is similar. In fact, the enrolment stage is completely the same but a couple of things change in the testing one. First, it is only necessary to compare the input feature vectors with the model for the identity which was claimed by the speaker. Secondly, the decision kind is “Yes” or “No”. As it was said before, this thesis is based on research carried out with speaker identification systems.

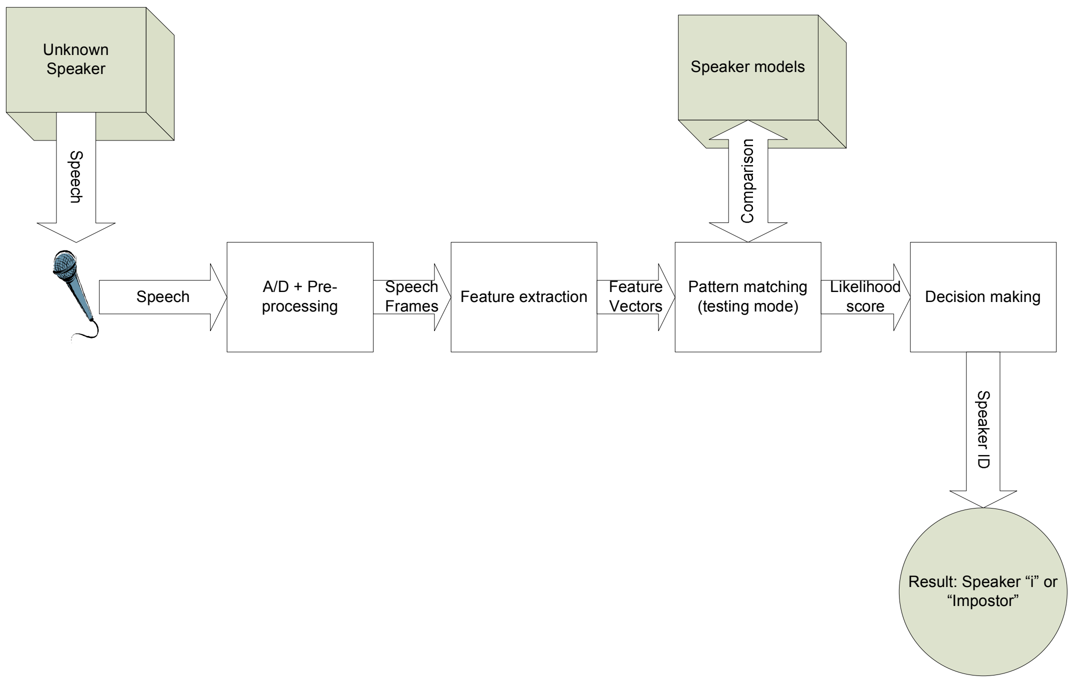


Fig 1.2 Generic identification system (testing stage)

1.5 Aim of this thesis

A general overview of automatic speaker recognition has been done in the previous sections. This description shows that the definition of speaker recognition comprises a large variety of systems. Thus, trying to fully cover all this range can be possible but futile.

In previous sections, it was exposed that the most complex and generic speaker recognition system conveys three properties. It is an open-set, text-independent speaker identification system. In this thesis three different systems which match this specification will be proposed.

There are various reasons to focus in this kind of speaker identification system. Firstly, since it is text-independent, it is highly versatile and can be applied in many different fields. If text-dependent behaviour is required, it can be adapted to work together with a speech recogniser so that specification can also be matched. Secondly, trying to obtain a system which is not ready to deal with impostors is a very limiting strategy. Nowadays, security is one of the most important aspects in modern software developing. Avoiding the inclusion an impostor detection module drastically reduces the range of applications in this kind of systems. Finally, verification can be seen as a particular case of identification. In addition, identification is a much more difficult matter since it implies comparing the input speech against a much larger number of models.

Therefore, the aim of this thesis is to obtain a speaker recogniser which can be applied in the widest range of scenarios. As it is stated in the theory described above, this is the most difficult kind of speaker recognition that can be achieved.

In the following chapter, general theory of speech production models will be described. After that, three different open-set, text-independent speaker identifications systems will be proposed. Two of them are language independent and one is language dependent. Their performance will be tested in chapter 8.

2 VOICE PRODUCTION MODELS

In the previous section it was shown that the speech signal has to be analysed and some speaker-dependent features are extracted from it before applying the pattern matching algorithm. It will be also shown in chapter 5 that these features can be obtained from different layers of the speech signal. These layers are classified from lower to higher extraction difficulty. Low-level features are those regarding physical traits or spectral information in the speech waveform. High-level features are traits learned during speaker's life. An example of a high-level feature is the semantic used by each speaker. In this thesis, only low-level features will be used. They have proved to be the most effective ones since learned traits can be easily disguised. For further information on this topic, please refer to chapter 5.

Therefore, since the physical traits derived from the speech signal will be used for the speaker recognition task, it is very important to understand properly how voice is produced and perceived by a human being. This knowledge has allowed researchers to develop digital models for these processes and in a final stage, apply these models to obtain speaker-dependent features from the speech signal.

2.1 Speech production anatomy and physiology

Speech is the result of a complex procedure carried out in the speaker's respiratory system. Figure 2.1 shows all the elements which take part in this process. All of these elements contribute somehow to the final speech signal. In fact, every block introduces some speaker-dependent information in the speech signal. Therefore, they will be individually analysed in this chapter. Much of the information here exposed has been extracted from [23].

2.1.1 Lungs

The lungs are used for the vital function of inhalation and exhalation of air. In the speech production model they are the power source that supplies energy to the rest of the blocks in the systems. Inhalation is achieved by reducing the lung air pressure. This is possible thanks to the rib cage and the diaphragm. The rib cage is expanded during this process. The diaphragm, which is placed underneath the lungs, is lowered so the lungs are expanded. This pressure lowering causes air to rush in through the vocal tract and down the trachea into the lungs.

Exhalation is opposite to inhalation. It is caused by an air pressure increase in the lungs. The volume of the chest cavity is reduced by contracting the muscles in the rib cage and lifting the diaphragm. This produces an air flow from the lungs to the larynx through the trachea.

Inhalation and exhalation always rhythmically follow the one to the other when breathing. However, during speaking short spurs of air are taken and the released steadily by controlling the muscles around the rib cage. Thus, the rhythmic breathing is overridden since expiration takes one sentence or phrase time. During this time the air pressure remains almost constantly above atmospheric pressure. However, as it will be seen later, the time-varying properties of the larynx and the vocal tract cause this constant pressure to become time-varying.

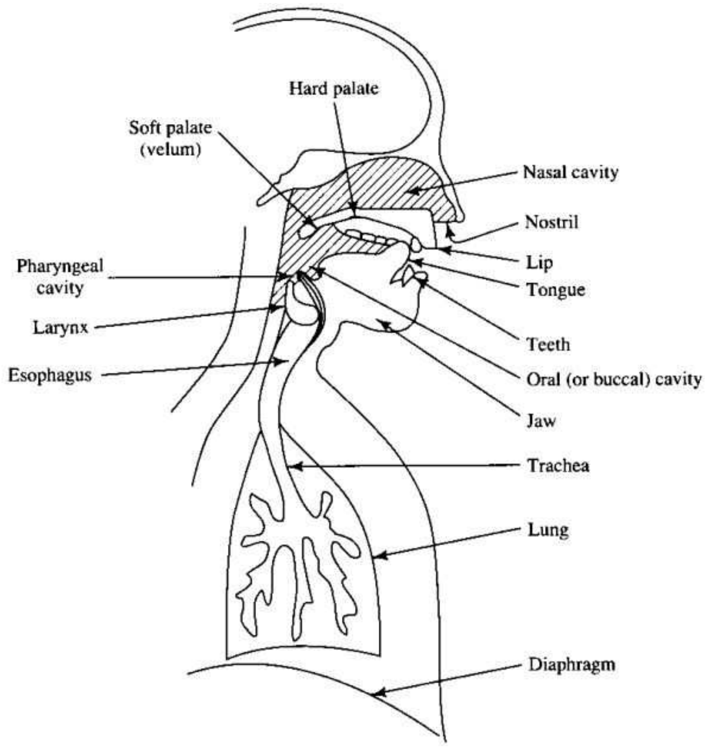


Fig 2.1 Respiratory system involved in speech production[1]

This airflow produced by the lungs has the shape of white gaussian noise. The only speaker-dependent information that is introduced by the lungs is the energy of this noise. However, this is not discriminative enough and other features have to be found.

2.1.2 Larynx

The larynx, also called the "voicebox" is a complex system of cartilages, muscles and ligaments. It has different functions such as closing the entrance to the lower respiratory system during swallowing. Since this kind of functions is not important for the speech production models, they will not be analysed in this section. From the voice production point of view, the most important parts of the larynx are the vocal folds and the glottis.

The vocal folds are two twin masses of flesh, ligament and muscle which stretch between the front and the back of the larynx. Their size varies from one person to another and in average it is around 15 mm long in men and 13 mm long in women.

[1] Source: Biometrics Information Resource. Voice Recognition. www.biometricsinfo.org

They can remain open to create unvoiced sounds or they can vibrate in order to produce voiced sounds during speech. During breathing, they remain open, allowing the air to flow into the lungs.

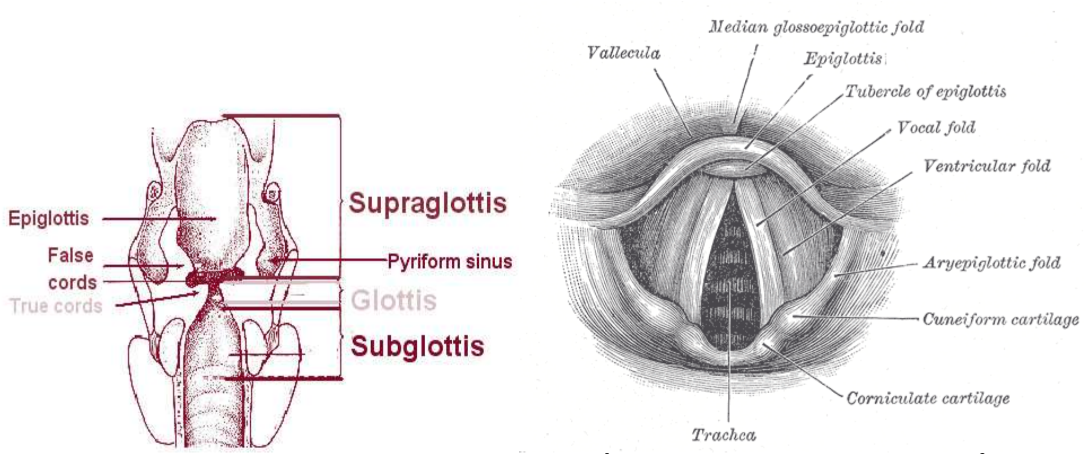


Fig 2.2 On the left, full schematic of the larynx[2]. On the right, view of the glottis[3].

Voiced sounds are characterised by the vibration of the vocal folds which open and close the airflow exit very quickly. This process is represented in Fig 2.3 and is known as the *Bernoulli's Principle* in the glottis. There are three steps in this process.

1. Initially, the vocal folds are open. Airflow is produced in the lungs and the vocal folds are immediately closed due to some air pressure effects. This is shown in step 1 in Fig. 2.3. This causes a vocal folds tension increment to hold a high air pressure as the lungs continue to pump air through the trachea as it appears in step 2 in the mentioned figure.

2. This is the voicing step and comprises steps 3, 4 and 5 in Fig 2.3. Finally, the vocal folds cannot sustain the high pressure generated and have to open (step 3). However, the pressure is reduced again (step 4) so the vocal cords can be closed again (step 5).

3. The vocal cords are in the same position as in step 2 in the figure so the whole process can be repeated again. This process is done several in a very short time period causing the vocal cords to vibrate and introduce a quasi-periodic pulse in the airflow. The frequency at which the vocal folds vibrate is called *fundamental frequency* or *pitch* (F_0).

This process creates a source signal for voiced sounds (such as vowels) which will be lately modified in the vocal tract. F_0 is higher for females than for males due to anatomical characteristics such as the length and mass of vocal folds which is lower in the case of women. In the case of children pitch is even higher. Therefore, estimation of the pitch can be a good gender or age discriminator. In addition, pitch does not remain

[2] Source: Seer's Training Web Site. Larynx. http://training.seer.cancer.gov/
[3] Source: Gray, H., Henry Gray's Anatomy of the Human Body, 20th U.S. edition.

constant during speech. Some ASR systems which use prosodic features take this pitch evolution into account despite the fact that it is relatively easy to imitate by an impostor.

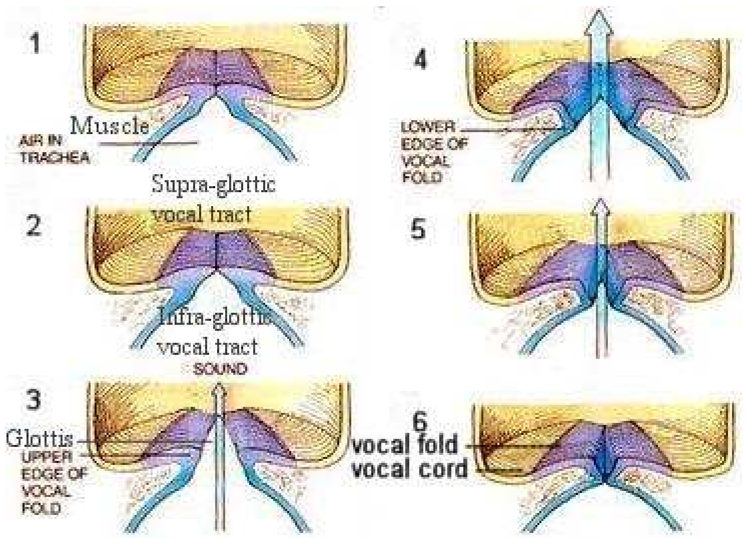


Fig 2.3 Voiced sounds production process in the glottis[4]

Some models have been created to model the airflow velocity output at the glottis. Left side of figure 2.4 represents two hypothetic glottal responses obtained from the *Liljencrants-Fant glottal model* [6]. The blue signal corresponds to a person speaking with a pressed voice. It is clear that the vocal folds are opened for a very short period. On the other hand, the red signal represents a breathy voice and the vocal folds remain open for a longer time. Both glottis responses show a pitch of 200 Hz. In the frequency domain, it can be seen that the longer the glottis remains open, the higher spectrum roll-off it shows. In addition, the spectrum contains a peak in every multiple of the fundamental frequency. Therefore, voice quality depends on the glottal pulse shape.

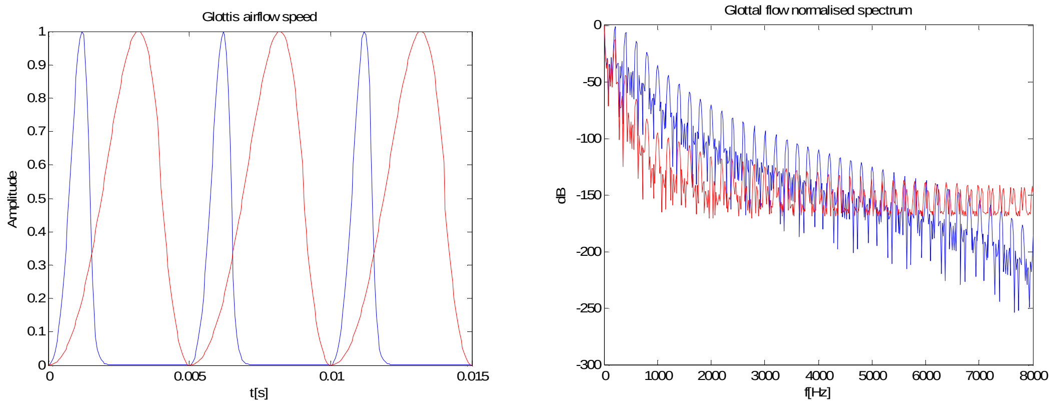


Fig 2.4 Glottal airflow speed and its normalised spectrum

[4] Source: A Review of the Universe - Structures, Evolutions, Observations, and Theories. Wave, Sound, and Music. http://universe-review.ca/.

Speaker individuality is also present in the quality of voiced. This quality is lower in the case of a breathy voice, as the glottis is not almost closed during the vocal folds vibration. It has been seen that this effect makes the glottal pulse spectrum to decrease with frequency in a faster way. However, as this glottal pulse will be later modified by the vocal tract, it is very hard to extract reliable speaker-independent information based on this issue.

2.1.3 Vocal Tract

Vocal tract is a generic term that refers to the voice production organs above the larynx. These organs are the pharyngeal, oral and nasal cavities and the velum as shown in Fig 2.5. It constitutes the main source of speaker-independent features which can be easily extracted. The length and shape of the vocal tract are highly speaker-dependent. Though, as it will be seen later, they are not fixed constants. However, their contribution to the speech signal spectrum is very important and relatively easy to measure.

The cavities themselves act as resonance boxes and spectrally colour the source airflow coming from the larynx. The frequencies at which the vocal tract resonates are called *formant frequencies* or simply formants. The velum controls the volume of air that flows into the nasal cavity. However, the vocal tract is not still during phonation. Its shape is time-varying as the movement of the tongue, lips, teeth, jaw results in a change in the vocal tract section. Each change is aimed to produce a different phoneme. When a new phoneme is to be produced, for example, the tongue has to move from the top of the oral cavity to the bottom. This movement can not be done immediately so there will be a transition time between both phonemes. In other words, one phoneme always affects the next one. This phenomenon is known as *coarticulation*. Because of this effect, frame overlapping is normally used as it will be shown in chapter 4. In addition, when a change in the position of the articulators is done, they remain in the same position for a time period of 20-40 ms. During this period the voice signal can be considered as a stationary process. This property is one of the most exploited ones in Speech Signal Processing and will also be used in speaker recognition.

All these properties make the vocal tract a very complex system since it can be regarded as a time-varying voice-box. However, it will be shown in next chapter that the most important speaker-dependent information is extracted from the vocal tract. In advance, the size a shape of each cavity varies from one speaker to another. Vowels are created when the quasi-periodic impulses generated in the glottis flow without opposition into the oral cavity. In voiced nasals the velum partially closes the conduct that leads to the oral cavity and only allows the air to flow into the nasal cavity. If the oral and nasal cavities could be approximated by a linear filter, the spectrum of a nasal or a vowel would contain much information about the nasal and oral cavities spectrum. This example is an introduction to the next section where the discrete time source-filter theory of speech production is superficially shown (for further details refer to [25]).

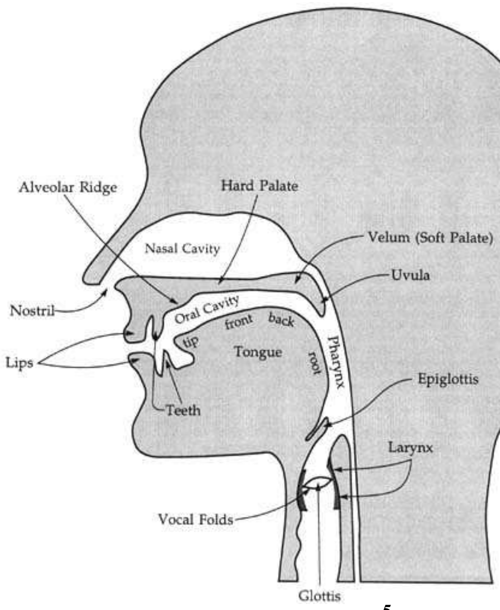


Fig 2.5 Vocal tract[5]

2.2 Acoustic Theory of Speech Production

In the previous section, the speech production process was seen under a physiological and articulator point of view. In other words, it was shown how speech sounds are produced. In addition, some important speaker-dependent properties obtained from this theory where introduced. In this section, a widely used digital model for this process will be shown [25].

In the previous section, it was shown that the voice production process was produced by a source signal (gaussian noise in the case of unvoiced sounds or a quasi-periodic pulse for voiced sounds) which modulates a time-varying filter (the vocal tract). The source-filter model for speech production exploits this approximation.

2.2.1 Source-filter model for speech production

The short study presented in the previous chapter about the speech production anatomy and physiology can be summed up with the blocks diagram in Fig 2.6. If the vocal tract is considered as a linear system, the final speech signal *S(z)* can be regarded as the result of filtering an excitation signal *U(z)* with the vocal tract impulse response *H(z)*. The excitation signal can be white gaussian noise in the case of unvoiced sounds or this noise convoluted with the quasi-periodic impulse train generated in the glottis.

The spectrum of the excitation signal is then coloured by the vocal tract. The vocal tract spectral response is characterised by the presence of certain resonant frequencies or *formants*. For voiced speech, these formants emphasize the source

[5] Source: University of Pennsylvania. Department of Linguistics. www.ling.upenn.edu/

spectrum in areas close to their location. The number of formants is unlimited. However, it is generally accepted that only the first four ones carry relevant information on the speech signal. The central frequency of each formant is numbered and is referred as *F1*, *F2*, *F3* and so on. Normally *F0* is reserved for the pitch frequency.

The real existence of the formants can be seen by using *spectrograms*. A spectrogram is a representation of a speech signal spectrum evolution along time. The intensity of the spectrum is coded using a colour scale from red (high intensity) to blue (low intensity). In Fig 2.7, the wideband spectrogram of the sentence "I love cats" is shown. The formants can be seen as small peaks in the voiced part of the speech utterance. For further information about spectrograms, see [25].

An important side effect of this theory is that the excitation source is considered independent from the vocal tract response. Some methods try to separate both components and use the information independently. The vocal tract transfer function is the most important for the speaker recognition task. It contains information about the speaker's vocal tract shape which is extremely hard to imitate.

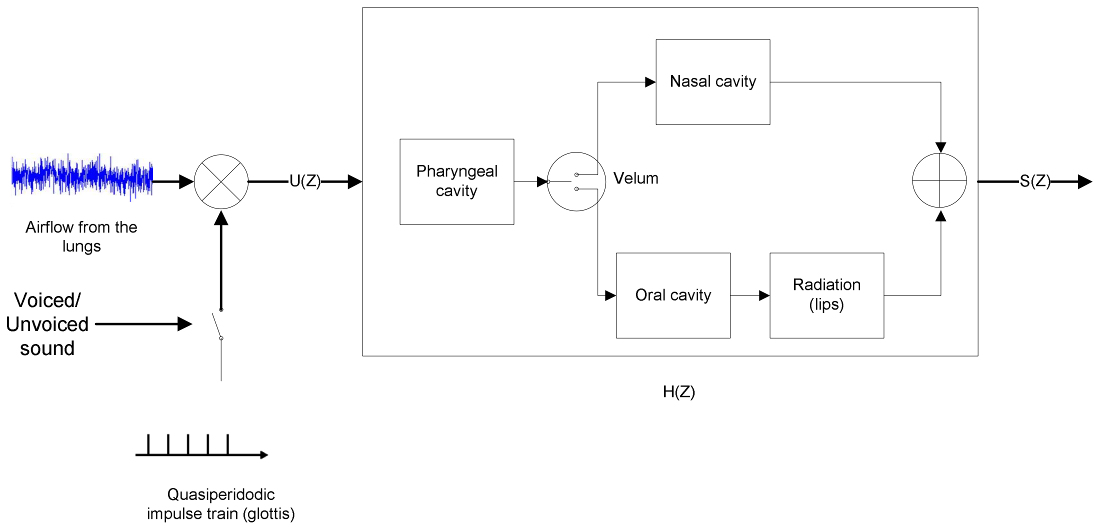


Fig 2.6 Speech production block diagram

A very important conclusion can be found in [16]. If the articulatory configurations of two speakers are assumed to be the same and the only difference is the length of the vocal tract, then the acoustic theory predicts that the formant frequencies are inversely scaled by the ratio of the speakers' vocal tract lengths. Thus, estimation of the formants location is a good speaker-discriminator. Therefore, spectral features will be used along this thesis as they contain much speaker-dependent information.

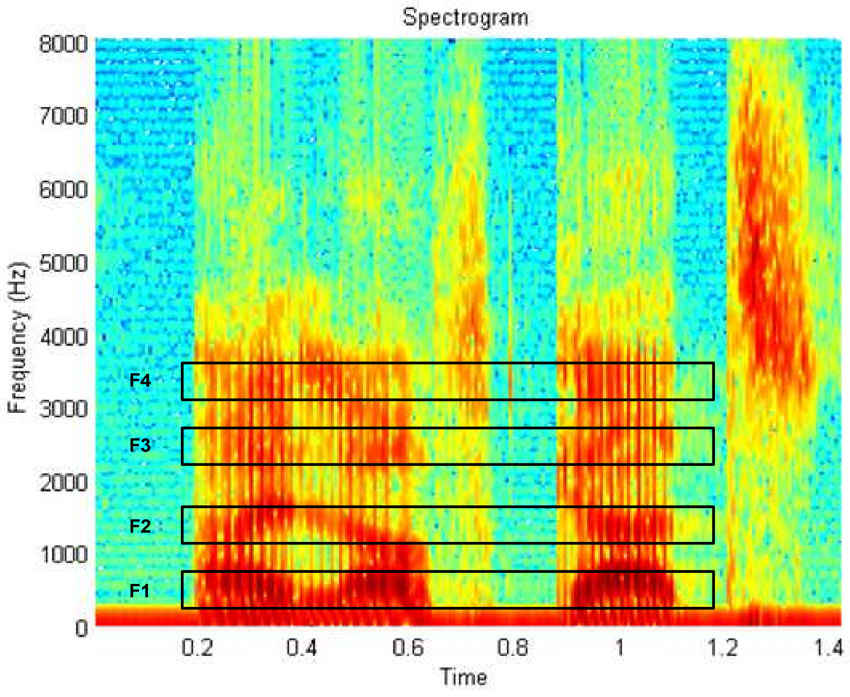


Fig 2.7 Spectrogram of the utterance "I love cats". The boxes show the voiced speech formants locations

3 AUTOMATIC SPEAKER IDENTIFICATION SYSTEM

So far, only the general theory of speaker recognition has been presented. A speaker recognition system is formed by various monolithic systems each of them dedicated to different tasks. In addition, the theory of speech production was also seen so the possible sources of speaker-dependent information which can be found in speech signals are known.

In this chapter, three text-independent open-set speaker identification systems will be presented. Two of them also have the property to be language independent whereas the last one not. In section 1.4, a generic speaker recognition system was resented. The final system here presented is a particularisation of the general one. All the subsystems will be implemented with Matlab [19] or the HTK Toolkit [15].

The main difference among the three systems is the way in which voice is classified prior to the application of the pattern matching system. Each speaker will be associated with an *N* number of models. *N* is the total number of classes in which voice is classified. This classification is done according to three different points of view. The first one divides voice into voiced and unvoiced segments. The second one clusters voice into classes using the *k-means* algorithm so feature vectors with similar components are clustered in the same class. The last one, which is language dependent, attends to a phonetic point of view. It was seen that voice production is a phenomenon which strongly depends on the articulator's activity. A nasal is not uttered in the same way as a vowel or a fricative. Therefore, it seems rational that different models for different sounds should be obtained per each speaker. In this approach, a speech recogniser will be used (this is the reason why it is language-dependent) to classify voice into different phonemes.

The three systems mentioned above are summarised in Fig. 3.1. All of them share the same blocks except for the voice classification algorithm. In section 1.4, it was also mentioned that a speaker recognition system can work in two different modes called training and testing. Therefore, in the following lines, the way these systems work in each mode will be explained.

- *Common to testing and training mode*: the speech input signal is firstly digitalised, pre-emphasised and segmented into short quasi-stationary frames (30 ms each). Then, a feature vector is extracted from each frame. This feature vector represents speech in a more proper way just by taking the speech parameters which are important for the speaker recognition task.

- *Testing mode*: The speech input in this case is a single random word or phrase produced by an a priori unknown speaker. In the feature extraction procedure, a sequence of feature vectors is obtained. This sequence is compared with a given set of speaker models using a pattern matching algorithm. This algorithm produces a likelihood score for each frame and speaker. With this information a decision making algorithm will give an estimation of who was the speaker that produced the speech signal.

- *Training mode*: The speaker identity is known a priori. This speaker produces a given length speech signal that is the same for all the speakers. The feature vectors are obtained in the same way as in the previous mode. In addition, each frame is classified using the voice classification algorithm. The speaker models are initialised so that they get some values which are supposed to be a good start. Each speaker is matched with *N* models, being *N* the number of classes into which voice is classified. Then, the pattern matching algorithm re-estimates these values using the sequence of feature vectors and the voice class type information.

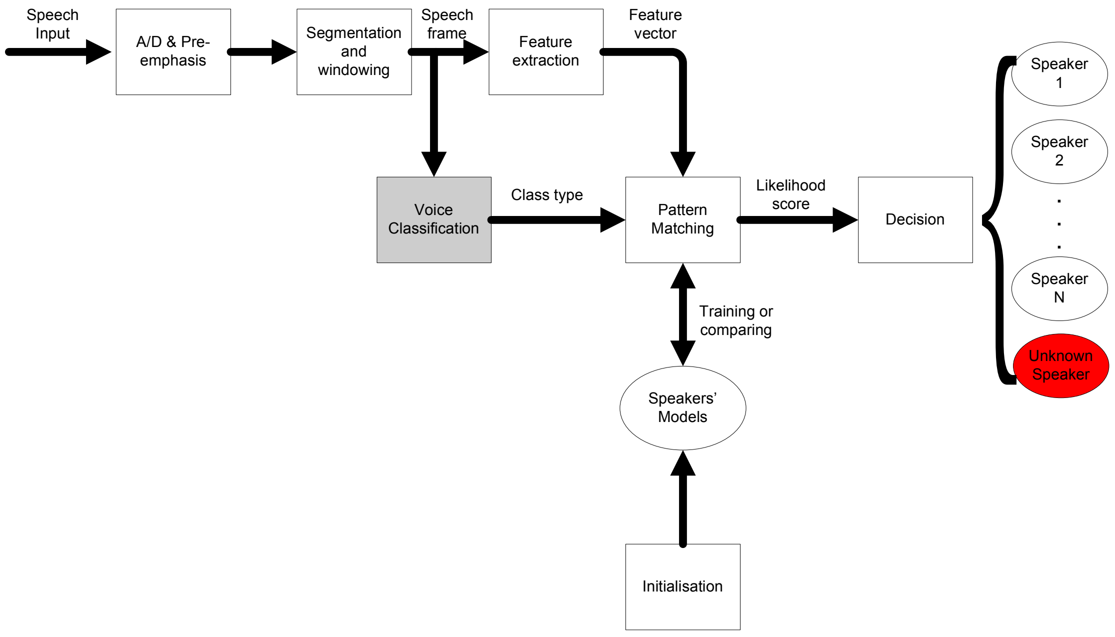


Fig 3.1 Speaker identification system

All this procedure can be divided into four main subsystems.

1. *Signal pre-processing*: includes the A/D conversion, basic filtering and segmentation into frames.

2. *Feature extraction*: obtains feature vectors from the speech signal containing highly speaker-dependent information.

3. *Pattern matching*: this subsystem includes the voice classification algorithm and the proper pattern matching algorithm. It also holds a database of speaker models and an algorithm to initialise a model prototype when a new user has to be added to the system. Its functions are to initialise and re-estimate speaker models in the training stage and to output a likelihood score for each speaker in the testing stage.

4. *Decision*: this subsystem will output the final decision taken about the estimated speaker who produced the test utterance in the testing stage. This decision is based on the likelihood scores given by the patter matching system.

The next chapters will be focused on each one of these subsystems.

4 SIGNAL PRE-PROCESSING

This system performs the basic pre-processing of the speech signal. Its basic diagram is presented in Fig. 4.1. First, the continuous speech signal *x(t)* produced by the speaker and sensed by the microphone has to be converted to the discrete domain. Secondly, the speech signal is segmented into frames. This is done to obtain quasi-stationary units of speech. Finally, a pre-emphasis filter is applied to each frame generated in the previous step. Once all this procedure has been performed, the speech frames are ready to enter the feature extraction subsystem.

In the following sections all these blocks will be described in merely descriptive way since they are widely used in speech processing and therefore well-known. If more information is required, [22] and [25] are reference books on this matter.

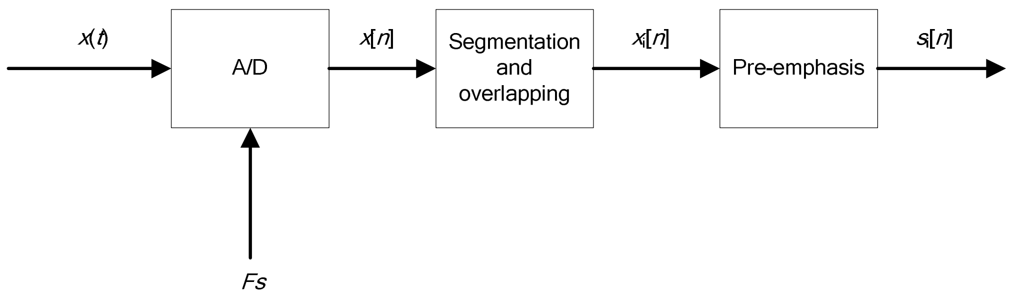


Fig 4.1 Signal pre-processing subsystem

4.1 A/D Conversion

The speech signal is a continuous air-pressure signal that can be captured by a microphone. The microphone converts this pressure-signal into a continuous electrical signal. The A/D converter target is to convert this continuous representation into the discrete domain so that it can be processed in the digital domain.

To perform this operation, the A/D module takes samples of the original signal with a frequency F_S called *sampling frequency*. According to the *Nyquist* theorem, this sampling frequency has to be at least the double of the input signal's bandwidth to avoid spectral aliasing. In speech signals this bandwidth can be approximated to 8 kHz since above this limit there is no relevant information so a sampling frequency of 16 kHz is enough. Furthermore, most of the information in the speech is located below 4 kHz. This is exploited by the telephone lines which apply a 4 kHz low-pass filtering to the voice signals. Thus telephone lines reduce significantly the speech signal bandwidth without a significant loss in the quality of speech.

The A/D converter also quantizes the amplitudes of the samples obtained into symbols or sequences of bits. This is called PCM (pulse code modulation). An example can be seen in Fig 4.2. Normally 8 to 16 bits provide sufficient resolution in the speech signal quantization.

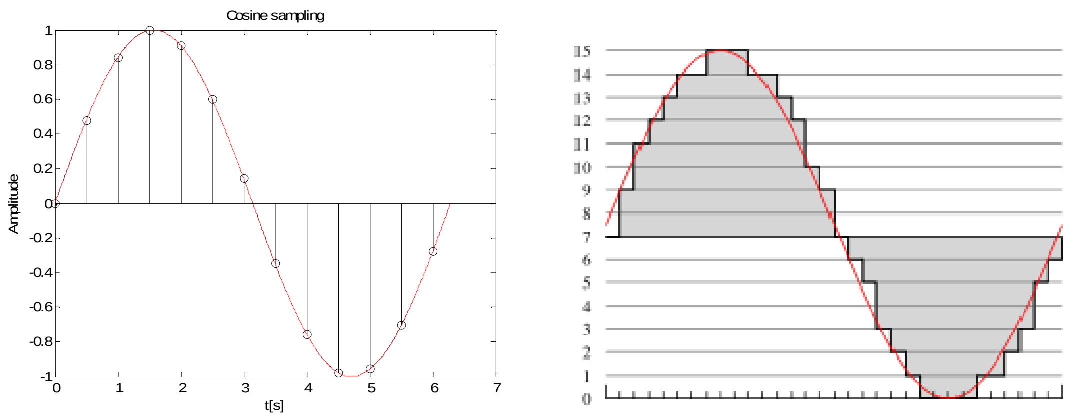


Fig 4.2 Sampling and 4 bits PCM encoding of a continuous sinusoidal signal[6]

4.2 Segmentation and overlapping

This block splits the PCM speech signal into short-length frames in order to obtain quasi-stationary speech units. The voice signal cannot be considered as a long-term stable signal as its properties vary considerably along time. However, if that signal is analysed in a very short period of time (order of milliseconds), the properties of voice do not change so drastically and it can be considered as a quasi-stationary signal. This lack of stability is produced by the movement of the articulators which vary their position to produce different phonemes. The period of time the articulators remain stable is about 80-200 ms. Segmentation is necessary to divide the speech signal into short-enough frames with quasi-stationary properties. Each of these frames will be individually analysed and used to generate a feature vector.

During speech production, the articulators change their position in order to produce the different phonemes. The transition between two phonemes involves the transition of the articulator organs from one position to another. This transition is not immediate and this is reflected in the waveform signal. Generally these transitions are problematic in speech analysis especially when a speech frame is centred in that transition. To avoid this effect, frame overlapping can be applied to the speech signal. Figs 4.3 and 4.4 show the importance of segmentation and overlapping. Two frames have been isolated. Both of them show quasi-stationary properties. One corresponds to a voiced sound and the effect of the glottis quasi-periodic pulse is clear. The other is taken from an unvoiced sound and is similar to white gaussian noise.

Segmentation and overlapping are done by windowing the speech signal using a window size of the same length as the final frames to be obtained. Once a frame is obtained, the window is delayed to obtain the following frame. Overlapping percentage will be defined by the length of the frame and the window delay. Rectangular windows

[6] Source: Wikipedia Commons. http://commons.wikimedia.org/

seem to be the most intuitively accurate. However, as it will be seen later, feature vectors will be obtained from the frame spectrum and there are other kinds of windows, as the Hamming one, which offer better spectral performance.

Fig 4.3 shows a blocks diagram regarding the segmentation and overlapping operation. First, a window with the same sized as the desired frame's size is applied to the first samples in the speech signal. Then, only the frames within the window limits are taken. The number of the last frame obtained *i* is used together with the desired overlapping level to calculate the delay which has to be applied to the window in order to generate the next frame.

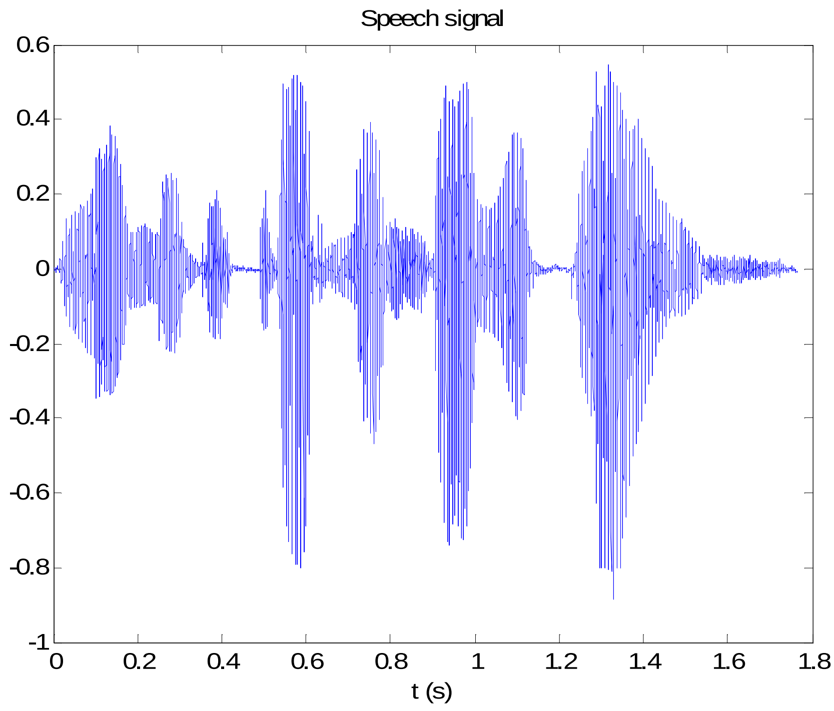


Fig 4.3 Speech signal of 1.8 s of duration. It can be observed that properties of the waveform change along time

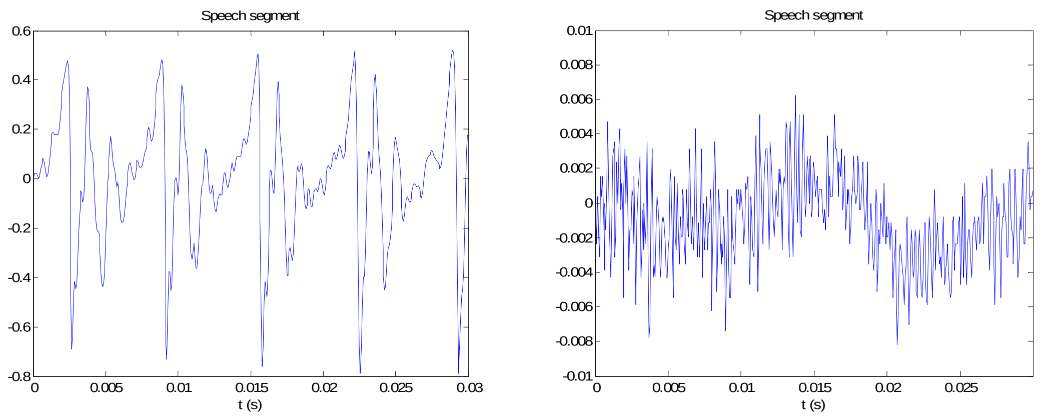


Fig 4.4 Speech frames of 30 ms each corresponding to a voiced (left) and an unvoiced (right) segment. It can be observed that both of them have stationary behaviour (the one on the left is periodic-like and the one on the right is white noise)

In the following sections, windowing will be studied from both the time and frequency point of view.

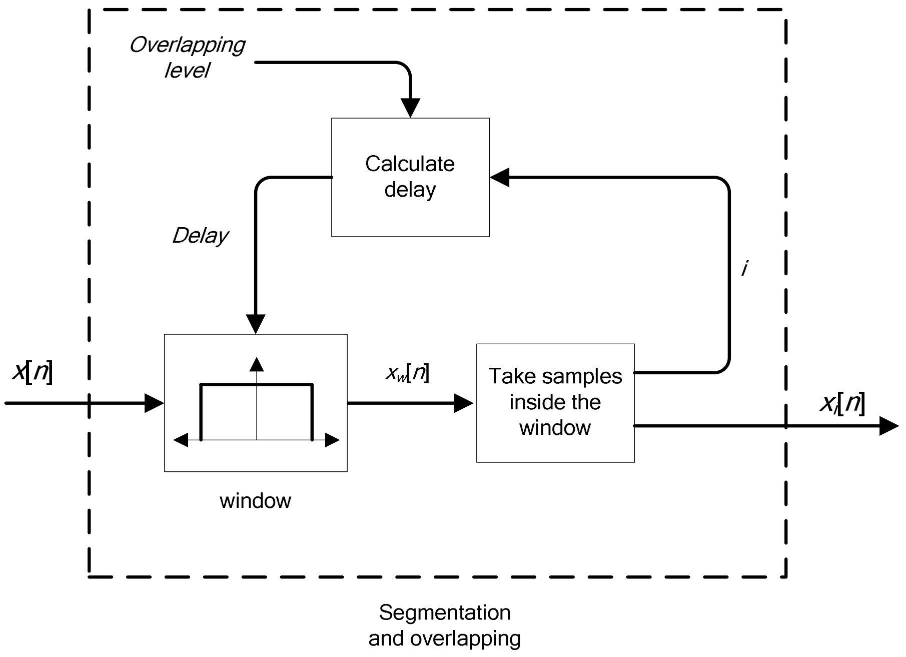


Fig 4.5 Segmentation and overlapping operation

4.2.1 Windowing

This section will offer an intuitive description about the way a temporal window modifies the spectrum of a given frame. Only a single frame will be studied, thus no window delay information is considered. The final result is relatively easy to generalise to the delayed case. However, the basic ideas about windowing can be reached without complex mathematical calculations due to this delay.

The output signal of the windowing block $x_w[n]$ can be calculated as

$$x_w[n] = x[n] \cdot w[n] \tag{4.1}$$

this is the input discrete speech signal multiplied by a discrete window function. This window matches the requirement

$$w[n] = \begin{cases} f[n] & \text{if} \quad 0 \le n \le W \\ 0 & \text{otherwise} \end{cases} \tag{4.2}$$

where W is the size of the window which determines the size of the frame and f is a function of n that declares the shape of the window.

Naturally, it can be thought that the best possible window is a rectangular one, as it produces no time-domain distortion in the speech signal. However, if a little study in the frequency domain is carried on, one can realise this is not true.

If the Fourier transform of (4.1) is taken

$$x_w[n] \xrightarrow{TF} X_w(\omega) = \frac{1}{2\pi} X(\omega) * W(\omega) = \frac{1}{2\pi} \int_{\theta=-\pi}^{\pi} X(\theta) \cdot W(\omega - \theta) \cdot d\theta \qquad (4.3)$$

the time-domain multiplication turns into a convolution in the frequency domain. This means that the spectrum of $x_w[n]$ will be the sum of the spectrum of $x[n]$ multiplied by a shifted copy of the spectrum of $w[n]$. In other words, the spectrum of $x[n]$ will be spread in means of the spectrum of $w[n]$. This is the reason why a rectangular window is not recommended as it has very high side lobes and will spread $x[n]$ spectrum along a large variety of frequencies.

There are other windows with better spectral behaviour than the rectangular one. This is the case of the Hamming window with

$$f[n] = 0.54 - 0.46 \cdot \cos\left(\frac{2\pi(n-1)}{W-1}\right) \qquad (4.4)$$

In Fig 4.6, a time and frequency domain comparison between rectangular and Hamming window is performed. Both window sizes are 21 samples. It is clear that the Hamming window has a wider main lobe but its side lobes are much softer. Therefore, the rectangular window causes a more noticeable leakage effect. The original spectral components of $x[n]$ will spread to other incorrect frequency components due to the convolution process defined in (4.3). On the other hand, as the main lobe of the Hamming window is wider than that one corresponding to the rectangular one, the rectangular window apparently seems to offer a higher spectral resolution as it spreads a wider range of frequencies that the rectangular one. However, its marked leakage effect is decisive for choosing the Hamming window.

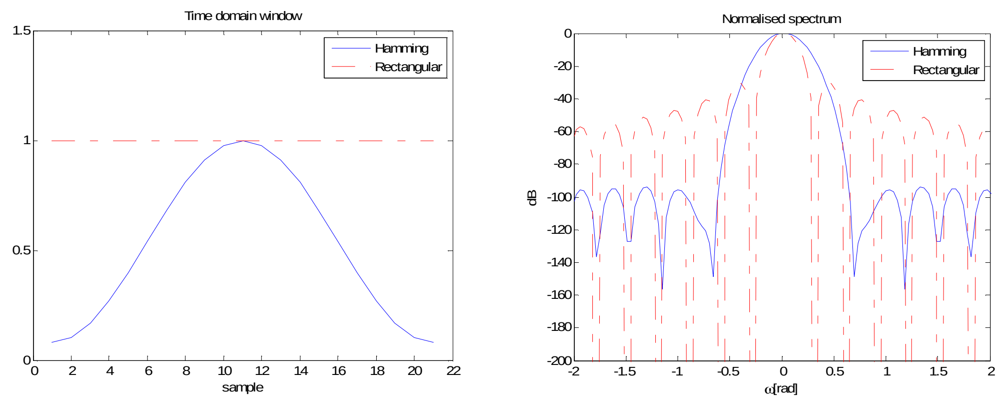


Fig 4.6 Comparison of rectangular and Hamming window

Once the Hamming window has been chosen, the next question one has to ask is: "which is the correct window size W?" This is a trade-off between time and spectral resolution. In Fig. 4.7, two Hamming windows of different length are shown. The spectral resolution (narrow main lobe) increases with the length of the window. However, increasing the size of the window involves loosing stationary characteristics of short speech frames. Choosing a window between 20 and 32 ms is a good compromise.

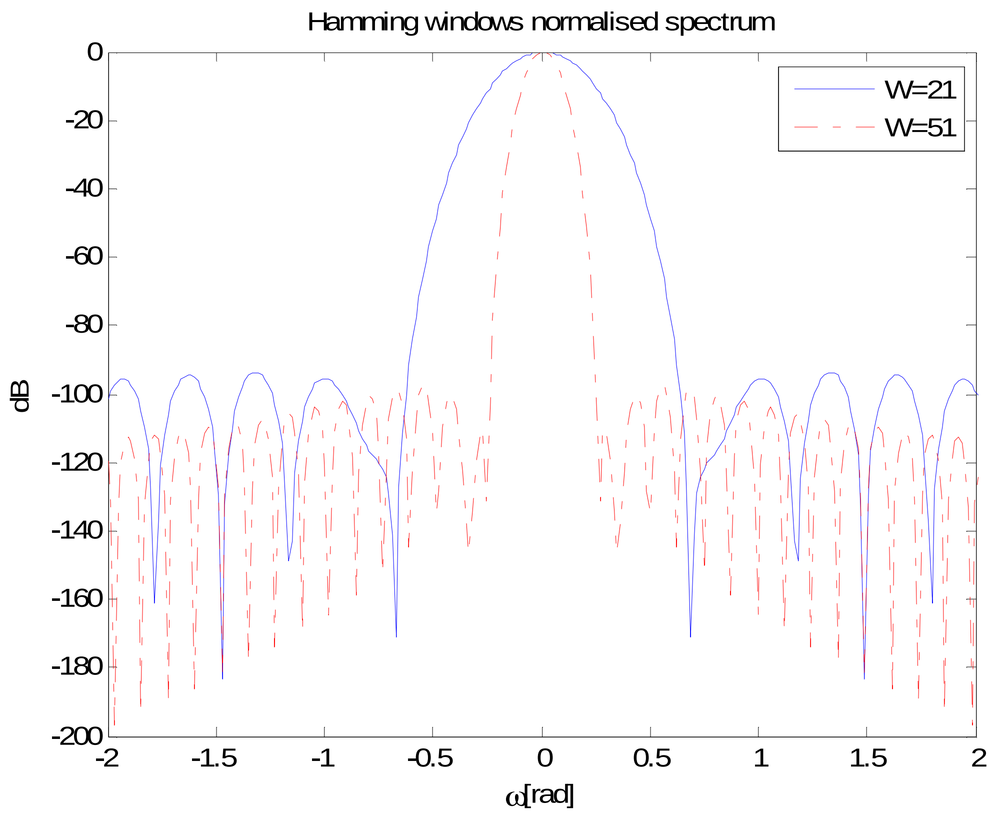


Fig 4.7 Normalised DFT spectrum of two hamming windows of sizes W=21 and W=51 respectively

4.2.2 Overlapping

The last step in the pre-processing stage is defining the overlapping of the frame sequence. The usefulness of overlapping is given by the fact that a single frame in uniform segmentation can contain a non-stationary transition between two frames. This effect can be reduced by using overlapping, as the probability that a window is centred on the middle of an inter-phoneme transition is reduced. Once again, it is necessary to find a compromise between a high correlation among subsequent frames and a lack of stationary properties in the frames.

High correlation between two frames is an undesired effect. It causes the system to compute large amounts of data with similar characteristics. Therefore, time consumption is increased without any performance increment.

4.3 Pre-emphasis

Pre-emphasis is widely used in speech enhancement. Its purpose is to lift the high frequency spectral components in the speech signal. There are various reasons to do this. Firstly, generally microphones increase the high frequencies roll-off in speech signals. Secondly, the *F3* and *F4* formants of speech are much lower than *F1* and *F2*. Therefore, it is interesting to lift these frequency components.

Voice production models are not the same for voiced or unvoiced sounds. For voiced sounds, there is a -6B/octave decay is speech radiated from the lips [25]. This is produced by the effect of the excitation source which contributes with -12dB/octave, whereas the radiation compensation of the lips produces a +6dB/octave compensation. However, these effects do not occur in the case of unvoiced sounds. Therefore, the compensation effect of pre-emphasis will cause around +12dB/octave increase in high frequency components of unvoiced segments.

Pre-emphasis is done by applying a FIR filter to the speech frame. The filter can be stated as

$$\begin{gathered} H(z) = 1 - a \cdot z^{-1} \\ 0 \le a \le 1 \end{gathered} \tag{4.5}$$

and performs the following difference operation with the input frame

$$s_i[n] = x_i[n] - a \cdot x[n-1] \tag{4.6}$$

Fig 4.8 shows a comparison of the frequency response of the filter for different values of *a*. Generally values from 0.9 to 0.98 are used since they introduce a +6dB gain in the high frequency spectrum of the signal.

Fig.4.9 shows the impact of pre-emphasis in voiced and unvoiced frames. In both cases, low frequencies are slightly lowered and the highest frequency components are boosted.

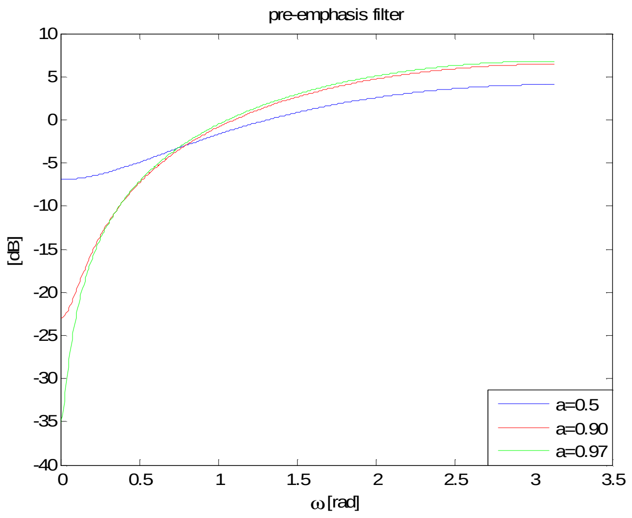


Fig 4.8 Pre-emphasis filters frequency responses for various values of *a*

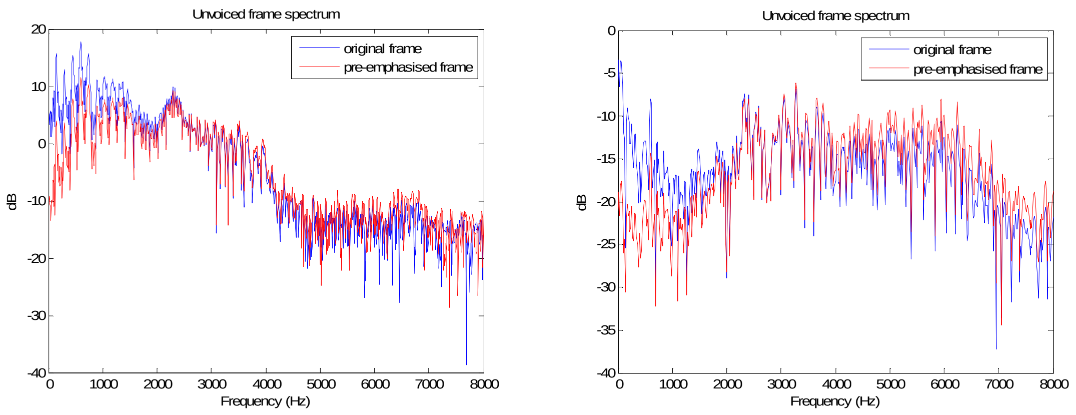


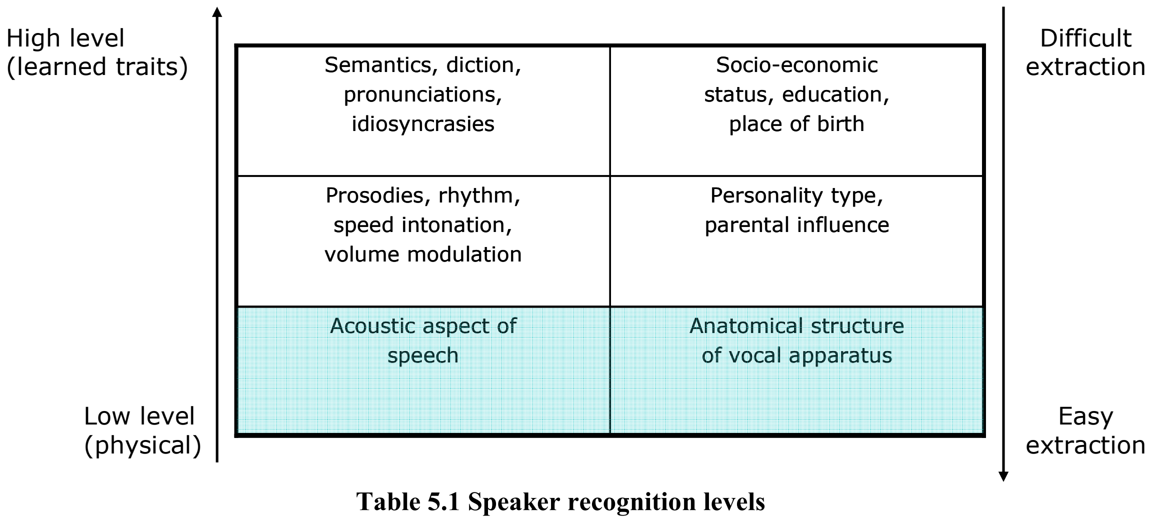
Fig 4.9 Frequency spectrum for voiced (left) and unvoiced (right) frames before and after pre-emphasis application

5 FEATURE EXTRACTION

In this step, relevant speaker-dependent information will be extracted from the speech signal. In chapter 2, some features derived from the speech production model were highlighted. Most of them are extracted from physical characteristics in the vocal apparatus of the speaker. However, much speaker-information can also be extracted from the speaker's voice signal. In [8] a short but concise description of the current state-of-the-art in speaker recognition systems can be found. In this reference a short review about the possibilities in feature extraction is also included.

Table 5.1 shows all the information which can be extracted from the speech signal. The column on the left refers to the feature itself whereas the column in the right is related to the speaker-dependent information which can be extracted from that feature. The lowest level in feature extraction has already been mentioned before. However, higher feature extraction levels are shown. These new features do not rely on the speaker's physical attributes. Though, they are based on the learned traits the speaker has acquired along his life.

Learned traits are quite harder to extract than the physical ones and have a drawback. Since they do not rely on the physical attributes of speaker's vocal apparatus, they can be imitated by an impostor. On the other hand, they are very robust against noise. In fact, when voice is produced in a noisy ambient, the speech signal is highly degraded causing a strong decrease in the performance of a physical-level-based ASR. However, humans do not seem so affected by the presence of noise when they have to recognise noisy voice. The clue is that humans use all the learned and physical information in the speech signal. Therefore, learned traits can be used to extract some useful information but recognition can not rely only on them. The most important information is present in the anatomical structure of the vocal apparatus since it can not be imitated. This thesis will only take care of physical attributes because they offer the best simplicity-discrimination rate from all the features here presented.


High level
(learned traits)
Low level
(physical)
Difficult
extraction
Easy
extraction

Semantics, diction, pronunciations, idiosyncrasies	Socio-economic status, education, place of birth
Prosodies, rhythm, speed intonation, volume modulation	Personality type, parental influence
Acoustic aspect of speech	Anatomical structure of vocal apparatus

Table 5.1 Speaker recognition levels

In the previous stage, quasi-stationary speech frames were obtained. These frames' dimensionality is too high for the pattern matching algorithm. Therefore, it is necessary to reduce this dimensionality. This reduction is performed by a surjective function f. Its aim is to perform a remapping of the speech waveform signal domain into a reduced feature vectors space as follows,

$$\begin{gathered} f : \mathbb{R}^N \longrightarrow \mathbb{R}^n \\ \mathrm{N} \gg \mathrm{n} \end{gathered} \tag{5.1}$$

This leads to a reduction in the quantity of information to be processed by the pattern matching algorithm and consequently to a reduction of time consumption.

Secondly, there is an excess of information in the speech waveform that is not interesting for the speaker recognition task. Feature extraction can remove most of this information. In addition the feature vectors emphasise those characteristics of speech which are highly speaker dependent.

There is a wide range of available feature extraction algorithms. The most interesting ones in the field concerning this paper should match the following properties [5] :

1. High inter-speaker variability.
2. Low intra-speaker variability.
3. Simple capturing and processing.
4. Robust against disguise and mimicry.
5. Robust against distortion and noise.
6. Vector components highly uncorrelated to each other.

The first two requirements are the basis of speaker recognition. High inter-speaker variability is required to obtain accurate models with a high discrimination power. On the other hand, low intra-speaker variability is necessary to avoid the fact that two utterances of the same word pronounced by the same speaker will hardly have the same waveform.

Simple capturing and processing is essential in real-time applications. Undesired effects as noise or distortion in the original waveform can be avoided by choosing feature vectors which are independent of these effects.

If the feature vector components are correlated, there is no sense in using all of them as the information they contain is being "repeated" in every component. Furthermore, as it will be exposed later, highly uncorrelated feature vectors lead to a simpler pattern matching and training algorithm. It is clear that a probabilistic model of these vectors will be characterised by very low cross-covariance values and the full covariance matrix can be approximated by a diagonal matrix with the variances of each component in the main diagonal.

Next sections will describe the different types of vectors used in this thesis. It is surprising that most of these vectors are also the basis for speech recognition. It seems to be that message and producer information travel together in the speech signal.

5.1 Mel Frequency Cepstrum Coefficient (MFCC)

MFCC are spectral features widely used in speech and speaker recognition. They are based on the speech processing carried out in the human ear and in the cepstrum of the speech signal. Not much computation is required for their calculation and generally beats other widely used parameters such as *linear predictive coding* (LPC) [16].

MFCC are generated frame by frame by the following process [1] [34]

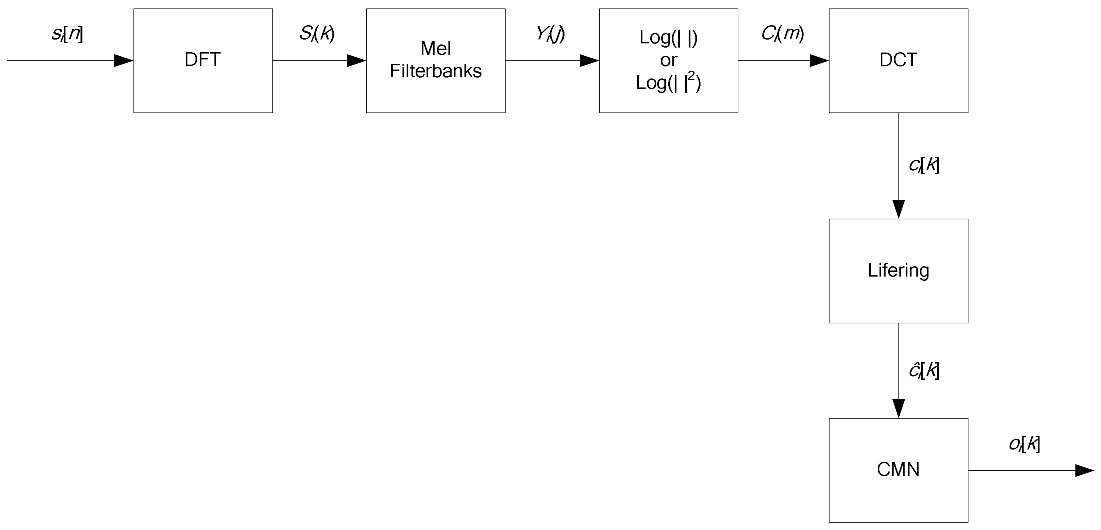


Fig 5.1 MFCC coefficients generation process

First, the *discrete Fourier transform* (DFT) is applied to the input frame. Then, a filter bank is applied to this spectrum simulating the behaviour of the human ear. The next step is compressing this weighted spectrum by the amplitude or energy logarithm. Finally, the inverse DFT (IDFT) of the log spectrum is calculated. Due to some properties of the log spectrum which will be remarked later, the IDFT can be calculated as a simple *discrete cosine transform* (DCT). In addition, some algorithms can be used to improve the MFCC performance. In this thesis *cepstral liftering* and *cepstral mean normalisation* will be used. Anyway, the following sections will describe this process in detail. More details about MFCC can be found in [2].

5.1.1 Spectral analysis (DFT)

The first step in the generation of MFCC is to calculate the DFT of the frame. Intuitively, the DFT is a discretisation of the Fourier Transform (FT) [22].

$$S_i(k) = \sum_{n=0}^{W-1} s_i[n] e^{-j\frac{2\pi}{W}kn} \tag{5.2}$$

where $k \in \mathbb{Z}$ is the discrete spectrum component index, $s_i \in \mathbb{R}^W$ is the input frame and W is the input frame components number.

It is easy to check that the DFT is a periodic signal with period W. In fact, if (5.2) is evaluated for $k = W$, the exponential part of the equation is reduced to the unity. Thus only the coefficients given by $k = 0,1,...,W-1$ are needed to represent the DFT spectrum of s_i. Furthermore, if s_i has a length of

$$W = 2^q \tag{5.3}$$

for any $q \in \mathbb{N}$, the FFT algorithm [22] can be used to reduce time complexity in the calculation of DFT from $O(n^2)$ to $O(n \log n)$ [1]. Therefore, if the length of s_i does not match (5.3), zero-padding will be applied to the signal so the FFT can be finally used.

In section 4.2.1 the importance in the window shape choice was remarked. I was concluded that a Hamming window was more accurate for spectral estimation than the rectangular one because of its frequency response. These conclusions can be ratified by analysing the DFT spectrum of a frame obtained by rectangular windowing and another by Hamming windowing. In Fig 5.2, the leakage effect caused by the rectangular window is noticeable since the pitch frequency pulses cannot be appreciated in the voiced frame case. However, this effect is reduced in Fig 5.3 where a Hamming window was used.

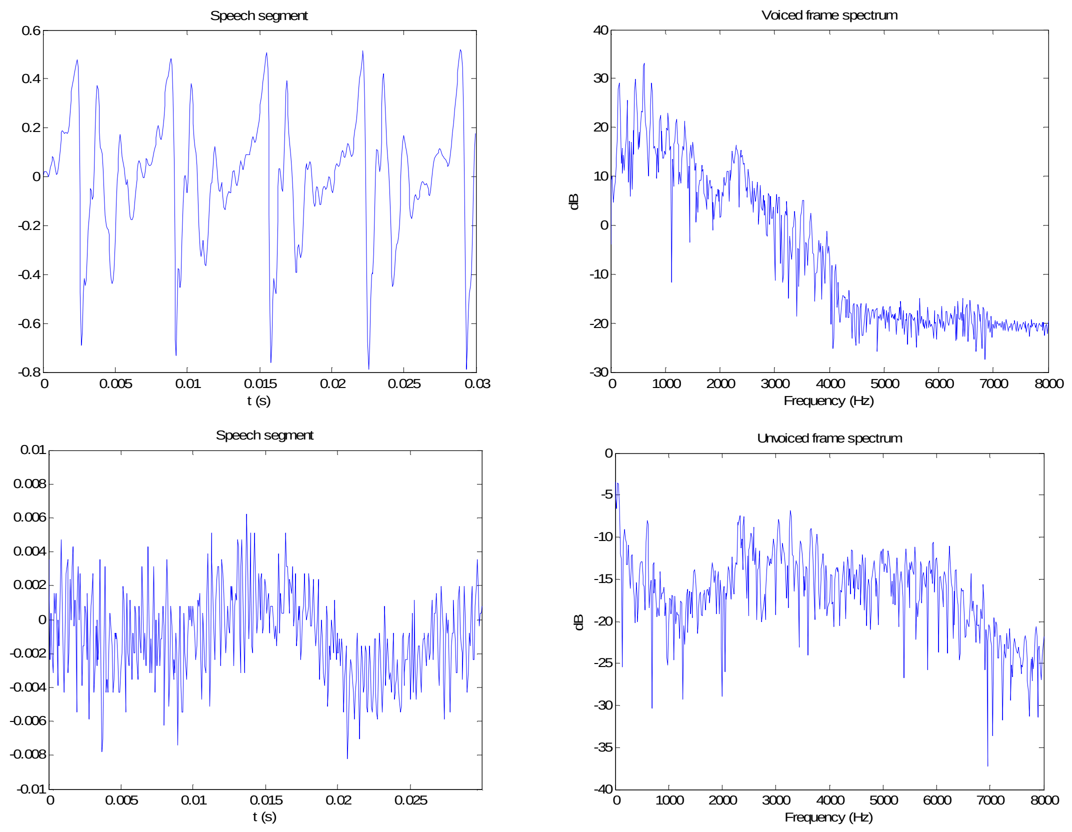


Fig 5.2 On the left, speech segments of 30 ms of duration each corresponding to a voiced frame (upper case) and an unvoiced frame (lower case) windowed by a rectangular window. On the right the corresponding spectral estimations of each frame.

5.1.2 Filter bank processing

The spectrum obtained in previous chapter is weighted by a set of filter banks in order to process the speech signal in different independent frequency subbands. This method is an attempt to simulate the behaviour of human ear system. Furthermore, the human ear resolves frequencies in a non-linear scale [34] so a not linearly spaced set of filter banks is designed.

Filter banks can be implemented in the frequency and time domain. In this case the filter banks will be directly applied to the short term DFT in (5.2). The shape and spacing of the banks is not fixed. In [35], a good comparison of different implementations of MFCC is found. It is demonstrated that the shape of the filters is not as important as if they are overlapped. The implementation used in HTK [34] uses triangular filters overlapped in frequency. These filters can be seen in Fig 5.4.

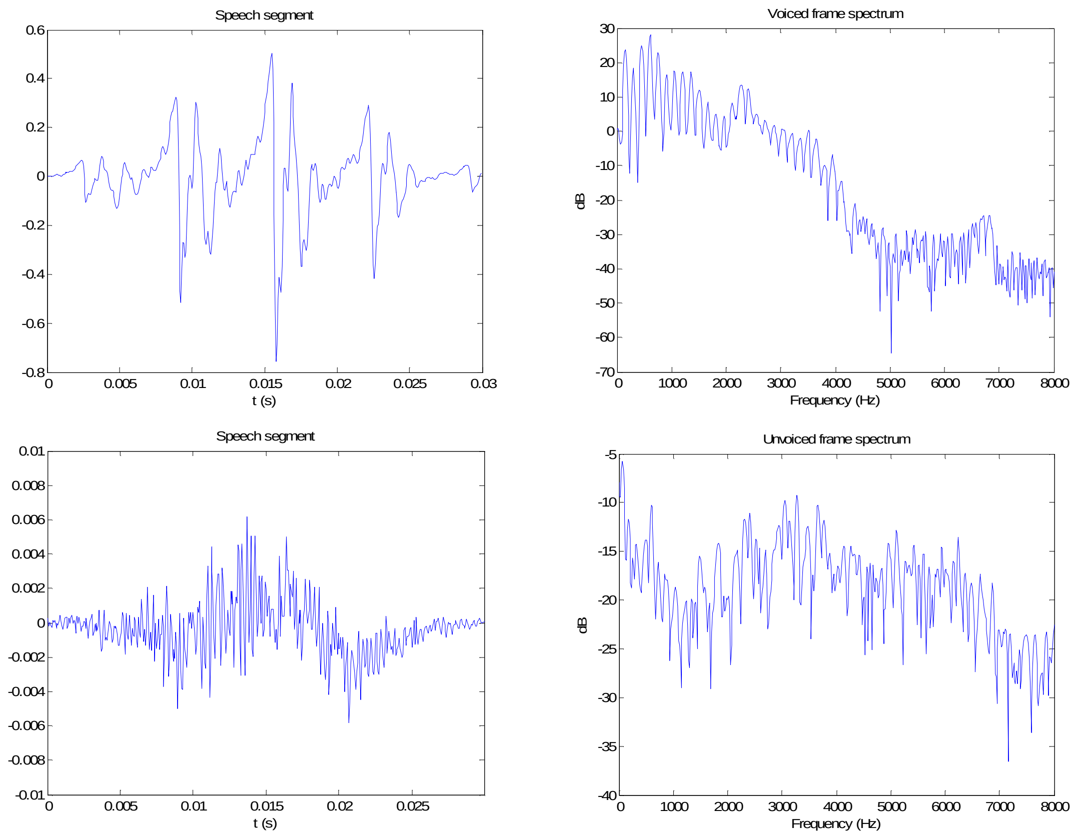


Fig 5.3 On the left, speech segments of 30 ms of duration each corresponding to a voiced frame (upper case) and an unvoiced frame (lower case) windowed by a hamming window. On the right the corresponding spectral estimations of each frame.

So every sample in the FFT is weighted by a spectral window. And the spectral response in every window is averaged so the N components in the FFT are reduced to M components called filter bank spectrum. If $F_j(k)$ is the window function or filter in the j-th frequency band, the filter banks output for the i-th frame is calculated as

$$Y_i(j) = \sum_{k=1}^{N} S_i(k) F_j(k)\,,\ j = 1 \ldots M \tag{5.4}$$

This operation converts the *N*-spaced DFT spectrum into a *M*-spaced filtered spectrum with $M < N$. Usually, *N*=512 components are taken in the FFT and between 20 and 28 *M* frequency bands are chosen.

The human ear perceptual response is non-linear. In fact, the needed increment of frequency in a tone in order to distinguish it from another tone of the same intensity increases with the frequency. Thus, the filter-banks are designed using a non-linear frequency scale. To generate the mel-frequency cepstral features, the filter banks are a set of triangular windows. Window centre are uniformly distributed from frequency component 0 to the Nyquist frequency or another interval between these two. The width of each window goes from the previous channel centre to following one. Finally, all these frequency values are warped using the mel scale which simulates the ear frequency resolution

$$\text{mel}(f) = 2595 \log_{10}(1 + \frac{f}{700}) \tag{5.5}$$

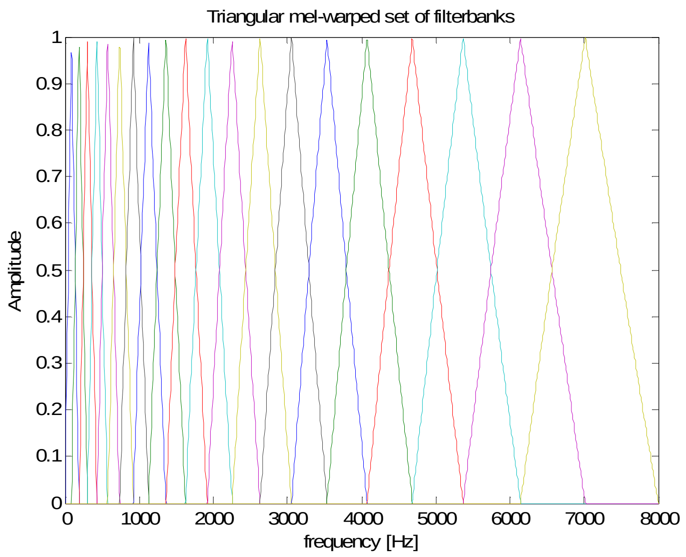


Fig 5.4 Mel-warped filter bank with 26 frequency bands.

All this process smoothes of the spectrum as it can be seen in Fig. 5.6. This process is similar to one performed in the human ear. However, it has a problem. The filter bank outputs are highly correlated. Next two sections will describe the last steps in MFCC computation which solve the problem here exposed.

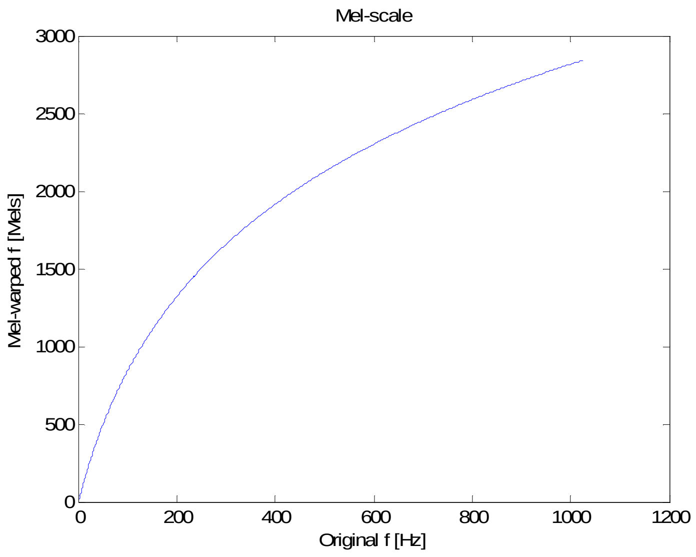


Fig 5.5 Mel frequency warping graphic

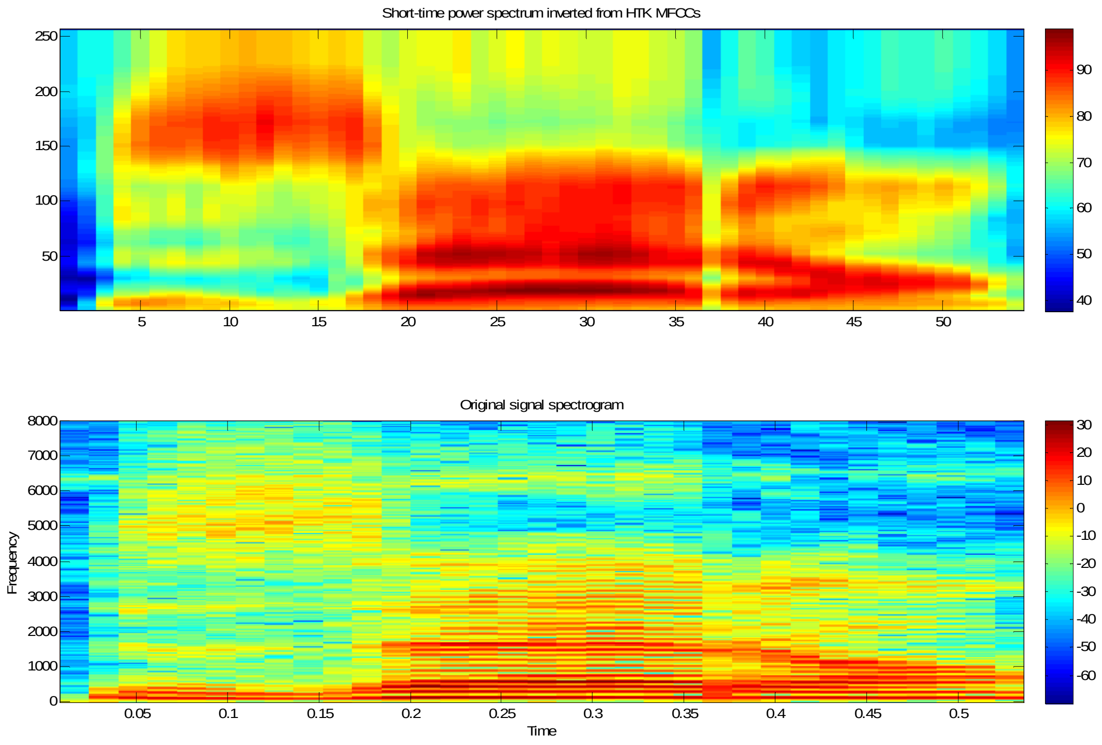


Fig 5.6 Comparison of short time power spectrum inverted from HTK MFCC implementation against the original signal spectrogram

5.1.3 Log filter bank amplitudes

The next step in the system process is to take the logarithm of the filter bank spectrum amplitude. There are two main reasons to perform this operation.

The first one attends to the homomorfic speech processing theory [25] where a deconvolution operator is applied to the speech signal. It can be considered that a short speech segment s[n] is the output of a slowly time-varying which input can be a quasi-periodic signal (case of voiced speech) or random noise (unvoiced)

$$s[n] = v[n] * h_v[n] \text{ being s[n] a voiced frame} \tag{5.6}$$

$$s[n] = u[n] * h_u[n] \text{ being s[n] an unvoiced frame} \tag{5.7}$$

where $v[n]$ is a periodic impulse train with period N and $h_v[n]$ is the impulse response of a linear system formed by the impulse responses of the vocal tract, the radiation and the glottis. In the unvoiced case, $u[n]$ is a random noise signal and $h_u[n]$ is the impulse response of a linear system formed by the impulse responses of the vocal tract and the radiation.

The radiation effects result in a high frequency emphasis and the glottal pulse shape is of finite duration. Thus a valid estimation is to consider both $h_v[n]$ and $h_u[n]$ as the vocal tract impulse response. If the case of a voiced frame is taken, by performing the DFT of (5.6) and taking the amplitude logarithm

$$S(k) = V(k) \cdot H_v(k) \Rightarrow \log\left(\left|S(k)\right|\right) = \log\left(\left|V(k)\right|\right) + \log\left(\left|H_v(k)\right|\right) \tag{5.8}$$

the convolution relationship between $v[n]$ and $h_v[n]$ in (5.6) becomes an additive relation. This simplification is very helpful as it will be shown later. In addition, the logarithm performs a dynamic compression, making feature extraction less sensitive to variations in dynamics [1]. Furthermore, by taking the amplitude of the spectrum, the phase information is removed. This phase is generally considered of not interest in spectral speech processing.

There is a way to separate the contributions of the vocal tract and the excitation. This is, to deconvolute both signals. This can be done due to the *cepstrum* properties. The cepstrum properties and details are described in [25].

The complex cepstrum of a signal is the inverse Fourier transform of the logarithm of the amplitude of the Fourier transform of the signal

$$\hat{s}[n] = \frac{1}{2\pi} \int_{-\pi}^{\pi} \log X\left(e^{j\omega}\right) e^{j\omega n} d\omega \tag{5.9}$$

with $\log X\left(e^{j\omega}\right) = \log\left|X\left(e^{j\omega}\right)\right| + j\arg\left(X\left(e^{j\omega}\right)\right)$. It has been demonstrated in [25] that the cepstrum of minimum phase signals can be computed using the real cepstrum. In addition, the real cepstrum is the even part of the cepstrum and thus, it has the same properties

$$c[n] = \frac{1}{2\pi} \int_{-\pi}^{\pi} \log \left| X\left(e^{j\omega}\right) \right| e^{j\omega n} d\omega \qquad (5.10)$$

where the logarithm of the Fourier transform amplitude is taken instead of the complex logarithm. Furthermore, by taking the DFT of the signal instead of the FT with an enough number of bins, a valid approximation is obtained.

The two main properties of the speech cepstrum can be observed in Fig. 5.7. Firstly, its low–time components decay very quickly. These low-time components correspond to the combined slow-time varying response of the glottis, vocal tract and radiation. The peaks at higher times correspond to the pitch period in voiced frames. These properties are very interesting as the vocal tract response can be estimated by only taking the low-order cepstrum coefficients. In addition, the cepstrum can be used to implement a pitch tracker.

In conclusion, if the high-order components of the cepstrum are removed, the excitation part of the cepstrum disappears and all the information it carries is just related to the vocal tract. These components can be removed just by using a time-domain window. Taking the logarithm of the amplitude (or energy) of the filter bank output is an approximation to calculating the real cepstrum of the input speech signal. Thus, if only the low-order filter bank outputs are retained, the vocal tract spectrum is estimated whereas the excitation, which is not a good speaker-dependent feature, is discarded.

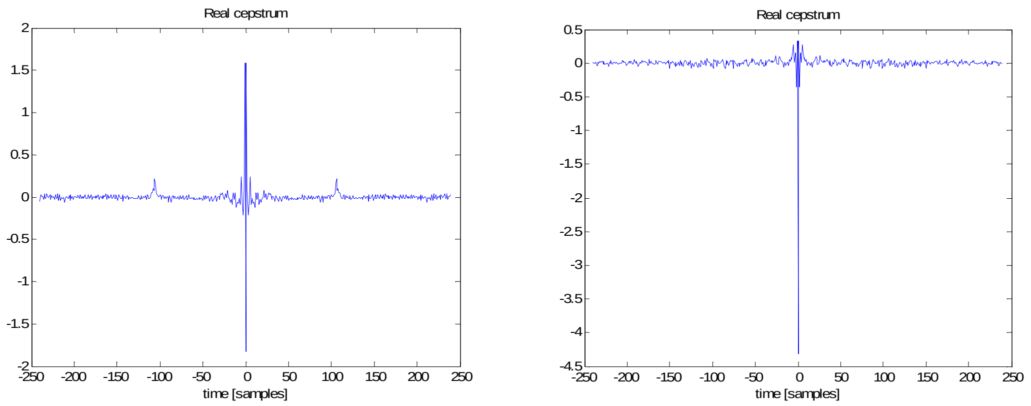


Fig 5.7 Real cepstrum of a voiced 30 ms frame (left) and an unvoiced one (right)

5.1.4 Inverse Discrete Fourier Transform (IDFT)

The last step to compute the MFCC features is to perform the IDFT of the logarithm of the magnitude of the filter bank output. The result of this operation is an approximation to the real cepstrum of the signal s[n].

The logarithm of the amplitude spectrum is a real and symmetric signal. Then, the IDFT will be only a sum of the product of each log spectrum component by the corresponding cosine function instead of a complex exponential, this is, a Discrete Cosine Transform [34]. The final MFCC can be computed as

$$c_i[k] = \sqrt{\frac{2}{M}} \sum_{j=1}^{M} \log\left|Y_i(j)\right| \cos\left(\frac{\pi k}{M}\left(j - \frac{1}{2}\right)\right), \quad (5.11)$$

$$m = 1 \ldots M,$$
$$k = 1 \ldots m,$$
$$m \leq M$$

where $c_i[k]$ is the k-th MFCC of the i-th frame. The total number of coefficients m can not be higher than the total number of frequency bands in the filter bank.

The use of a DCT has great advantages since its components are highly uncorrelated. Thus, if a Gaussian model is used to represent the MFCC coefficients, the full covariance matrix can be successfully approximated by a diagonal variances matrix. This considerably reduces the computational cost of the pattern matching algorithm.

The zero-order coefficient in (5.11) is approximately equivalent to the log energy of the frame.

5.1.5 MFCC liftering

The output of the high frequency filter banks is quite low. Therefore, high-order coefficients will have a value to small. MFCC liftering can be applied so the high order coefficients are boosted. This process allows the pattern matching algorithm to perform more accurate comparisons. MFCC liftering is done by applying the following expression to the MFCC coefficients [34]

$$c_i'[k] = \left(1 + \frac{L}{2} \sin \frac{\pi k}{L}\right) c_i[k] \quad (5.12)$$

5.1.6 Cepstral mean normalisation (CMN)

The performance of MFCC can be also improved by using some speech enhancement algorithms. The microphone and the transmission channel used in the enrolment stage are usually different to the ones present in the testing stage. Therefore, MFCC coefficients for the same utterance will not be the same in both stages. However, this can be solved by applying CMN.

The effect of the microphone and transmission channel is multiplicative in the frequency domain. This multiplication is converted to a sum after the logarithm has been taken in (5.11). Therefore, if the mean of the cepstral coefficients is subtracted, this effect can be compensated.

CMN is done to compensate long-term effects. Thus, the average of each MFCC component is calculated along all the training examples so a cepstral mean vector is obtained. Finally, this vector is subtracted from each MFCC vector in the training and testing set. Thus the final coefficients are obtained. In Fig. 5.1 the final coefficients for frame *i* are expressed as $o_i[k]$. The observation vector for the frame *i* will be called

$$O_i = o_i[1]...o_i[m] \tag{5.13}$$

being $o_i[k]$ the *k*-th MFCC coefficient of the *i*-th frame and *m* the number of MFCCs extracted.

The target of this notation is to remark that in the following chapter, these feature vectors will be considered as observation vectors.

5.1.7 MFCC first and second derivative coefficients

The basic MFCCs can be upgraded by adding the first and second derivative coefficients to the MFCCs set. This operation increases the dimension of the feature vectors resulting in a higher demand of training data. The physical interpretation of these coefficients is not clear. However a valid approximation is to consider that they reflect dynamic information in the speech production procedure.

First derivative coefficients, also known as *delta* coefficients are obtained via the regressive formula for all *k*

$$d_i[k] = \frac{\sum_{j=1}^{2} j\left(c_{i-j}[k] - c_{i+j}[k]\right)}{2\sum_{j=1}^{2} j^2} \tag{5.14}$$

There is a problem in this expression in the beginning and end of speech since no previous data is available. In this case, the first or last vector is replicated as it is stated in [34].

Second derivatives are obtained by applying (5.14) to the delta cofficients.

6 PATTERN MATCHING

The pattern matching subsystem is the kernel of the ASR. In comparison to speaker recognition performed by humans, it takes the role of the brain. First, it keeps a register of the characteristics of a given input voice like human brain does. Secondly, during the testing stage, it uses this memory registers to compare a test utterance. The result, as in human SR, depends on how familiar to the system the voice is. This is what the likelihood score of the input signal represents.

The pattern matching task is performing the identification procedure. The feature vectors obtained in the previous chapter will be used to either generate a proper model of each speaker or to compare the speech input to the stored speakers' models and calculate a likelihood score. Thus, pattern matching involves two important functionalities which are training (in the enrolling stage) and testing (in the testing stage).

There is a wide range of possible choices in the pattern matching algorithm. This choice depends on some variables such as the purpose of the system and the available amount of training data.

Pattern matching algorithms can be classified into two main classes [5]. These are *template models* and *stochastic models*. The use of one or another kind of pattern matching algorithm depends on the context and the purpose of the ASR. In the following lines a brief but comprehensive overview of these pattern matching algorithms will be made in order to justify the selection of some of them.

a. Template models

This is the simplest kind of pattern matching algorithm. As it name suggests, the model for each speaker is a template $\overline{O}$ of the input speech feature vectors $\{O_i\}_{i=1\ldots F}$ used for training. Please note that F is the number of frames per speaker used in the training stage. The likelihood score for each frame is calculated by a distance function $d\left(\overline{O}, O_i\right)$ between the observation vector and the template vector.

As an example, in the simplest case, the template is the statistical mean of each feature vector component calculated along all the training frames. This is called a *centroid*. The simplest distance function is the Euclidean one. Therefore, the closer an input vector is to the template model, the higher likelihood score is obtained for a given template.

This kind of algorithms achieves a good performance in text-dependent speaker verification. If it were applied to text-independent speaker verification, one single template would not be capable of modelling all the acoustic information a speaker is able to generate. Too many templates per speaker would be needed and it would derive in a performance decrease. However, they have a great performance in text-dependent context where only templates regarding the prompted passwords are needed. Furthermore, if the aim of the system is to perform verification instead of identification, only one template has to be compared. Hence template models are generally used in text-dependent speaker verification.

Dynamic time warping (DTW), *vector quantization* (VQ) and *nearest neighbours* (NN) are some examples of algorithms based on template models. Since the aim of this thesis is building a text-independent speaker identification system, they are not considered to be in the scope of this thesis. However, it will be seen later that some algorithms related to VQ can be useful. For further details check out [5].

b. Stochastic models

Now the model is expressed in terms of probabilities and probability density functions (pdf's). Each observation vector corresponding to a given frame is considered to be random. Thus, each utterance produced by a speaker can be considered as a random sequence of feature vectors. Stochastic models attempt to create accurate models for these sequences attending to the random sequence statistics such as its mean, variance, or probability distribution. The shape of the feature vectors probability distribution corresponding to a given speaker differs from the other speakers' one. Therefore, the aim of stochastic models is to calculate the likelihood score of an utterance for each speaker model. In stochastic models this is a synonym of calculating the probability than a given utterance was generated by a given speaker model.

The two main stochastic pattern matching algorithms are *gaussian mixture models* (GMM) and *hidden Markov models* (HMM). The difference between them is that GMM takes into account a single feature vector corresponding to a single frame whereas HMM can model quite accurately a sequence of feature vectors. Therefore, it is possible to intuitively think that GMM can suit very well the text-independent task whereas HMM are good in text-dependent context. Furthermore, HMM are widely used in speech recognition due to their good sequential modelling. However, this thesis will show how short sequential information in the speech production process can be interesting for text-independent speaker recognition.

The pattern matching subsystem used in this thesis can be seen in Fig.6.1. Firstly, each speech frame is classified into a different number of classes. In chapter 3, it was said that three different systems are to be implemented. The only difference among them relies on the classification system used. The first system will classify speech frames into voiced or unvoiced, the second one will use vector quantization in order to cluster feature vectors attending to their similarity and the last one will use phonetic information.

Secondly, in the training stage, the pattern matching algorithm will use all the training feature vectors to build speaker models. One model per speaker and per voice class will be created. The process for creating this model consists on formerly defining the initial models architecture and values and then re-estimating them.

Finally, in the testing stage, feature vectors will be compared to the previously trained models and a likelihood score (the probability that a given utterance was generated by a speaker model) will be outputted for each speaker.

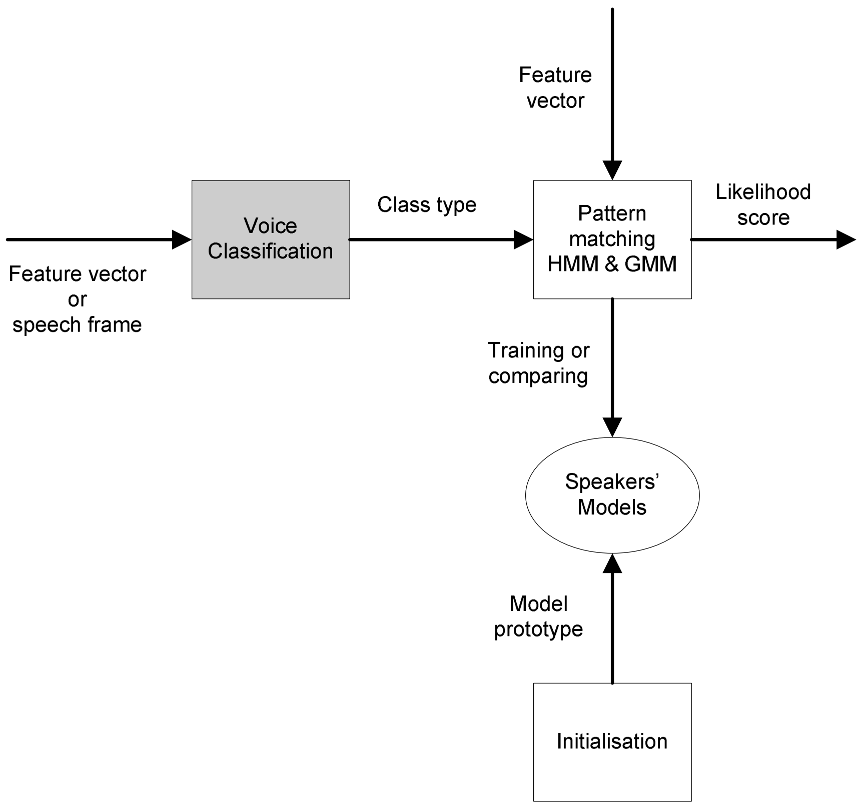


Fig 6.1 Pattern matching subsystem

The pattern matching algorithm used in this thesis is oriented to operate in text-independent context. Thus, stochastic models will be used. Concretely, the approach will be a combination of HMMs and GMMs. It may sound contradictory that HMMs are used in this context. However, some speaker-dependent sequential information which is interesting for text-independent speaker identification can be extracted from the random feature vectors sequence.

Following sections will describe all the blocks used in the pattern matching subsystem. The voice classification block will include the description of the three different systems implemented.

6.1 Voice classification

This section will describe the three different speech classification methods used in this thesis. This is a critical choice since one model for each voice class and for each speaker will be latterly obtained. Firstly, a single voiced/unvoiced decision algorithm will be implemented. Secondly, a classification algorithm based on vector quantization will be used. Finally, voice will be classified attending to acoustic or phonetic characteristics.

Model parameters will be obtained by using the HTK Toolkit for training HMMs. It will be shown later that HTK needs the input training files to be properly labelled. Thus, the task of the voice classification algorithm is labelling all the training files with the time locations of voiced and unvoiced sounds. The performance of each of these methods is tested and compared in chapter 8.

This classification will be performed only in the training stage. Its purpose is to classify voice into different groups so one model per class and per speaker can be obtained. These models which will be latterly obtained are discriminative enough so the use of the classification algorithm in the testing stage is not necessary. This is a great advantage for implementing real-time recognition systems since it not necessary to waste time in the classification stage during recognition. In addition, it will be possible to use more complex classification algorithms in the training stage since time consumption is not such a critical parameter as in the testing stage.

6.1.1 Voiced/unvoiced decision

It was shown in chapters 2 and 5 that the spectrum of the speech signal varies strongly depending on the way this signal was generated. In fact, according to the source-filter model for speech production, speech is the result of applying an excitation signal to the filter conformed by the vocal tract. In the case of voiced sounds, this signal is a quasi-periodic train pulse whereas unvoiced sounds are produced by the action of the turbulent airflow coming from the lungs. The result is that the spectrum of voiced sounds is totally different from unvoiced sounds' spectrum. Hence, it seems reasonable that models for voiced and unvoiced speech should be quite different.

There are many voiced/unvoiced (V/U) decision algorithms. In many of them, this decision is a side effect of a pitch detection algorithm. The advantage in using a pitch detection algorithm for V/U decision is that they are robust against noise and quite accurate. However, pitch estimation requires a relatively high time consumption. Since no pitch information is required, pitch estimators waste too much time performing a non-needed operation. That is why a faster algorithm is required.

Another solution is taking advantage of some speech signal properties. In fact, if one inspects Fig. 6.2 where the Polish word for number four (*"cztery"*) was pronounced, it is clear that voiced sounds (remarked in blue) are highly periodic whereas unvoiced sounds (remarked in red) are most likely noise. This property will be exploited by calculating the *zero crossings* number of each frame as it will be shown in next section.

In conclusion, a fast algorithm based on the speech frame *zero crossings* will be used. Its performance will be compared against a robust pitch estimator provided by the VoiceBox Toolbox [4]. This pitch estimator is out of the scope of this thesis. A detailed explanation of this algorithm can be found in [31]. Next section will describe the algorithm based on zero crossings used for voiced/unvoiced decision.

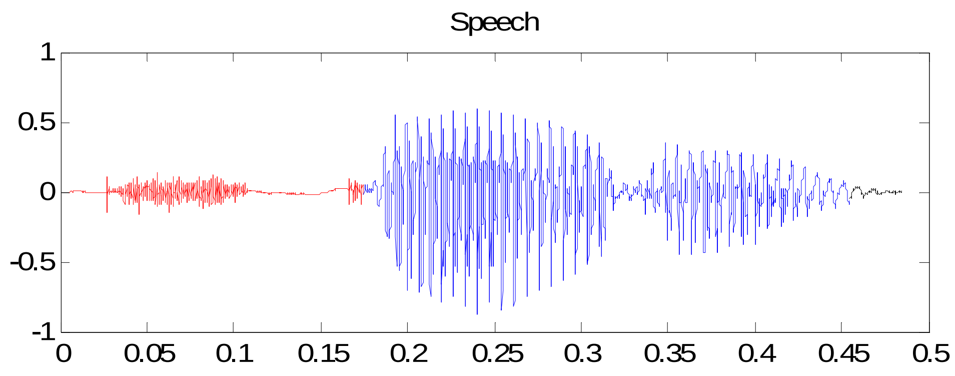


Fig 6.2 Speech signal corresponding to the pronunciation of "cztery" ("four" in Polish)

6.1.1.1 Zero crossings

The number of zero crossings of a signal is defined as its number of transitions from the positive to the negative region [25]. The zero crossings number of a speech frame can be mathematically expressed as

$$Z_i = \sum_{n=1}^{W} \left| \operatorname{sgn}\left(x_i[n]\right) - \operatorname{sgn}\left(x_i[n-1]\right) \right|. \tag{6.1}$$

This measure is sometimes used as a rough pitch estimator since the following approximation can be done

$$Z_i = \frac{2F_0}{F_s} \tag{6.2}$$

where F_0 is the pitch frequency and F_s is the sampling frequency of the speech input.

In Fig. 6.3, the result of calculating the zero crossings number for the same utterance as in Fig. 6.2 is represented. Since unvoiced sounds are similar to noise and therefore a high frequency signal, the zero crossing rate is very high. On the other hand, the zero crossings rate is quite constant for voiced sounds and lower than for unvoiced sounds. Finally, the rate for silence is normally very low.

Thus, a reasonable method for classifying speech into voiced or unvoiced frames is to calculate the zero-crossings rate and compare this value against two given thresholds as follows

$$decision = \begin{cases} 0 & \text{if } a < Z_i < b \quad (voiced) \\ 1 & \text{otherwise} \quad (unvoiced) \end{cases} \tag{6.3}$$

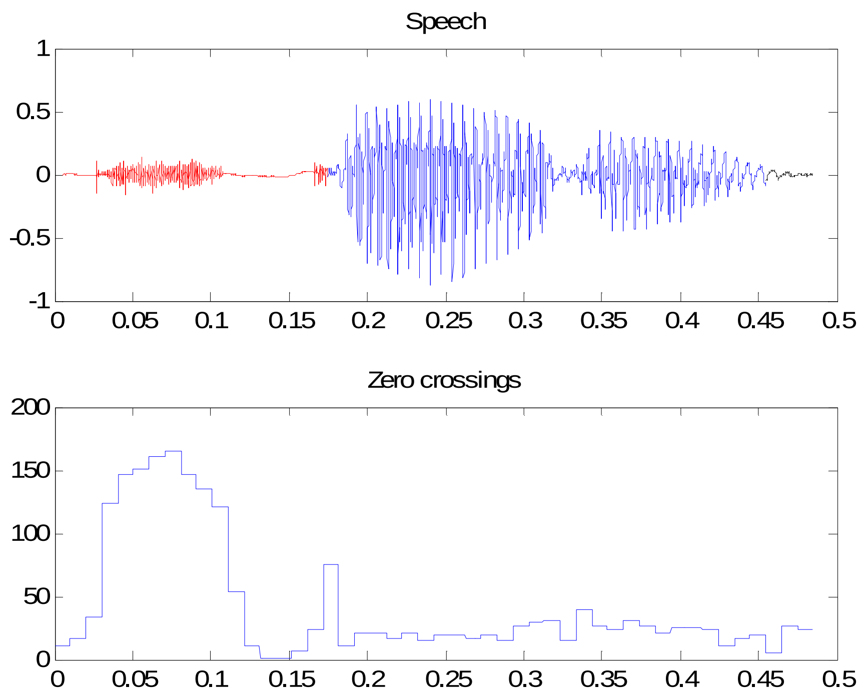


Fig 6.3 Zero crossings number corresponding to the pronunciation of cztery (four in Polish).

where *a* and *b* are the lower and higher thresholds respectively. The value of *a* and *b* can be obtained empirically. This decision is not good enough yet, since some isolated frames are incorrectly classified. For example, in Fig. 6.4, the plot in the middle represents a decision based on these thresholds. It can be noticed that some isolated frames are being incorrectly classified. This can be solved by applying a median filter in order to smooth the decision.

Finally, a median filter of length 3 is applied to the decision obtained in (6.3) so its result is smoothed. The result can be seen on the bottom plot of Fig. 6.4.

The V/U decision here implemented is not robust against noise and neither as accurate as that one based on pitch estimation. However, it is much less time-consuming and its lack of accuracy is not critical in the system performance as it will be shown in chapter 8.

6.1.2 Vector quantization (VQ)

There is a wide range of sounds a speaker is able to produce. In consequence, it seems that dividing voice just into voiced or unvoiced frames can be an inaccurate approach. Thus, the aim of vector quantization is to divide voice into some more classes than the apparently inaccurate V/U decision. Whereas V/U decision operates in the time domain, vector quantization will directly work with feature vectors. The target is to cluster each feature vector into a class type so all the feature vectors belonging to a given class are close in terms of a given distance function. A good definition of clustering is also found in [18] as "the process of organising a set of data into groups in such a way that observations within a group are more similar to each other than they are to observations belonging to a different cluster".

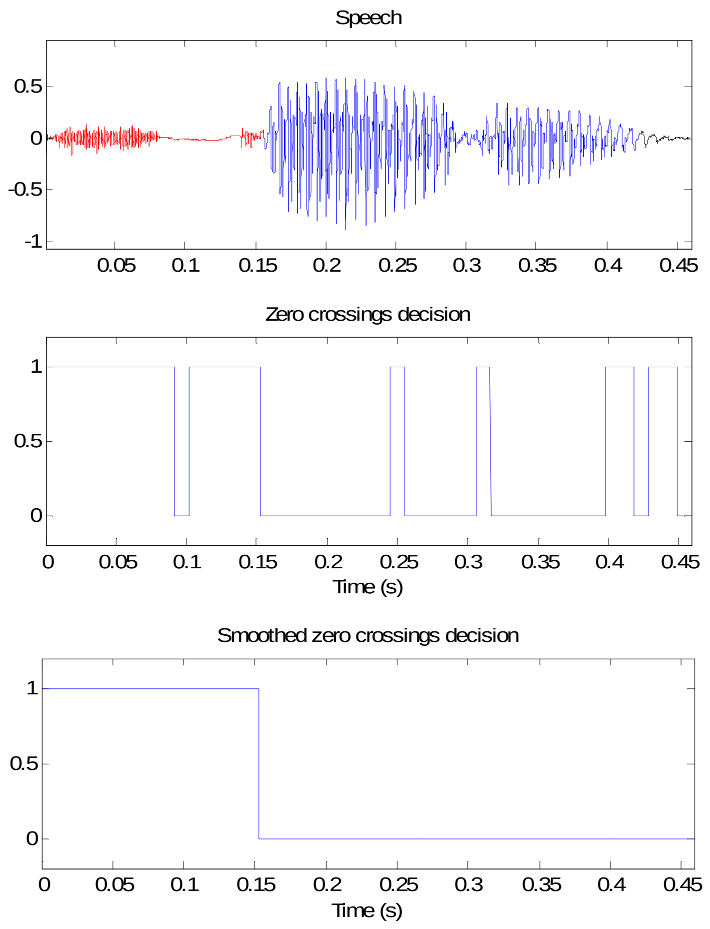


Fig 6.4 Zero-crossings-based decision corresponding to the pronunciation of cztery (four in Polish)

According to the last paragraph, VQ will work with *M*-dimensional vectors. However, in order to perform an intuitive approach to VQ, all the examples here exposed will consider the case *M*=2 (bi-dimensional). Thus, in Fig. 6.5, the first two MFCCs observed for a given speaker utterance are represented. It can be noticed that feature vectors cluster into at least two different classes since they tend to group in two space regions marked with coloured ellipses.

This class partition can be done by using the *k*-means algorithm. The goal of this algorithm is to cluster the observation vectors into *k* groups such that the within-group sum-of-distances is minimized [18]. These distances will be calculated with relation to a given point of reference or *centroid*. Each class will be characterised by its centroid. *k*-means uses an iterative algorithm to re-estimate these centroids and each feature vector will be labelled with the class name whose centroid is closer to the given feature vector. For example, in Fig. 6.6 the final classification and centroids obtained after applying the *k*-means algorithm with Euclidian distance are represented. It is observed that the classification is accurate. The centroids are located in the central region of each cluster. Next section will describe the *k*-means iterative algorithm.

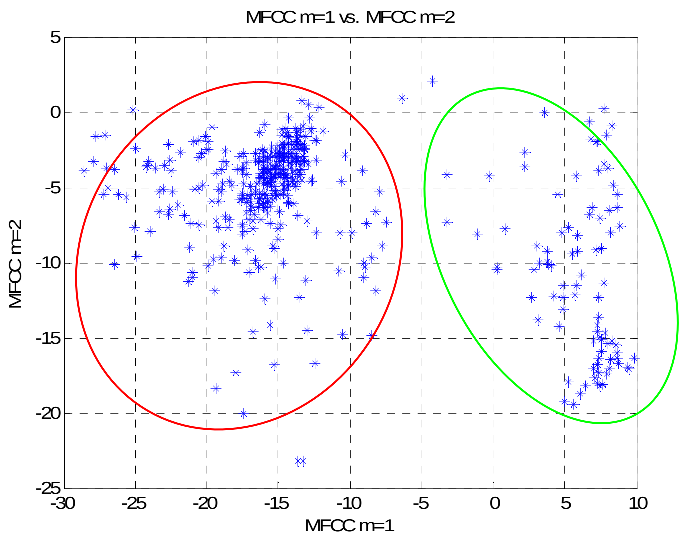


Fig 6.5 First and second MFCCs for a given speaker utterance

6.1.2.1 *k*-Means algorithm

The basic *k*-means algorithm can be found in [18]. The iterative procedure for clustering and calculating the centroids is the following.

1. Set the *k* number of clusters.
2. Determine initial cluster centroids.
3. Calculate distance between each observation vector and each cluster.
4. Assign every observation vector to the closest cluster.
5. Calculate the new centroid (i.e. the mean vector) of every cluster using the observation vectors that were just grouped there.
6. Repeat steps 3 through 5 until no more changes are made.

Steps 1 and 2 are critical to obtain a reliable partition. The larger *k* is used, the better discrimination can be obtained. However, increasing *k* leads to higher demand of training data. Therefore, a trade-off between these two issues has to be found. The initial chosen clusters have also a critical impact on the final result. Indeed, choosing a bad set of initial centroids can result in a time consumption increase (i.e. more iterations needed) or an inaccurate classification. Normally, these centroids are randomly extracted from the observation vectors' set. However, when there is an a-priori knowledge of what the final clusters should look like, the initial centroids can be set by the user.

In this thesis the distance function used is Euclidean and the centroids are calculated using as the mean vector of each cluster for each iteration.

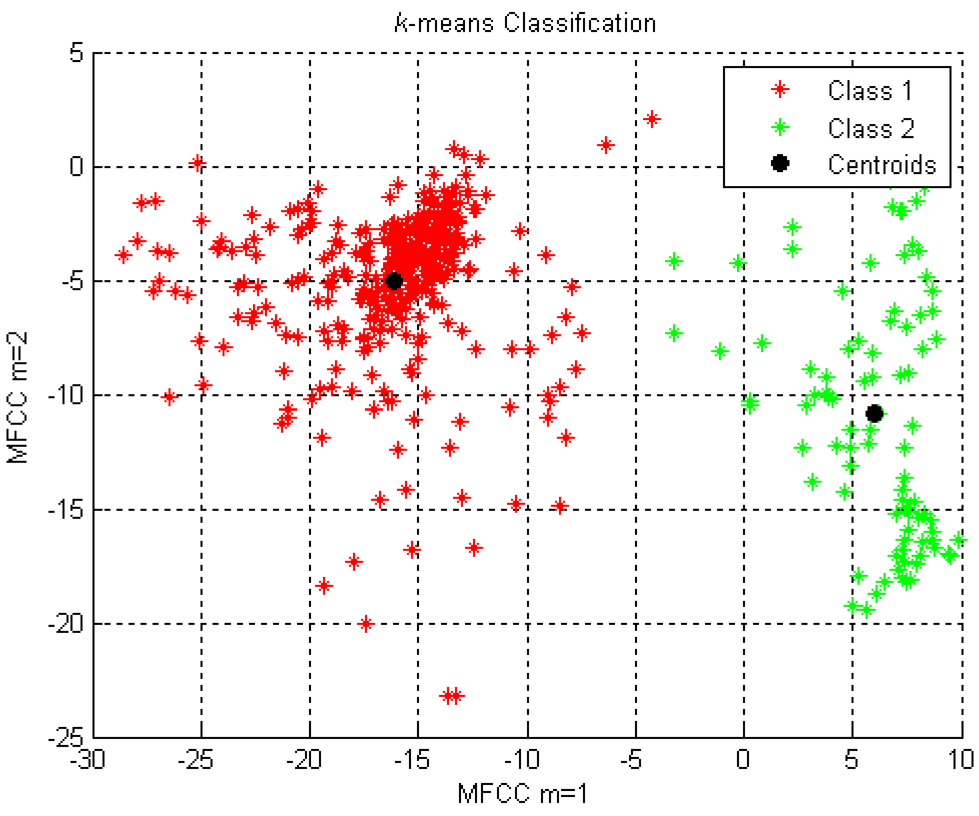


Fig 6.6 *k*-means clustering of a given speaker utterance into *k*=2 classes

6.1.3 Phonetic classification

The last classification method is based on the acoustics or phonetics of the speech input. Each speaker produces a wide variety of sounds by moving the articulator organs. Attending to the articulator organs involved in the generation of a given sound, this one can be classified into different groups. Once more, the number of classes to be used has to be large enough to obtain a good discrimination but short enough in order to avoid a deficit of training data.

This classification will be done using a speech recogniser trained for Polish speech. Therefore, this method will be optimised for Polish language and will be language-dependent. The speech recogniser is based on Hidden Markov Models. Since it is out of the scope of this thesis, its details will not be exposed here. The reader is referred to [36] for a wide exposition on the topic.

A phoneme is a realisation of an articulated sound. These phonemes can be classified attending to three different criterions.

1. *Action of the vocal folds.*
   A phoneme is considered as voiced if the vocal folds vibrate during its production process. Otherwise it is considered unvoiced. This classification was already used in section 6.1.1.

2. *Place of articulation*.
   Depending on the place where the articulators produce an obstruction in the vocal tract. E.g. bilabials (between the lips), dentals (between the teeth), etc.

3. *Manner of articulation*.
   The way the phoneme is produced in means of the organs used in its production and the way they are used. E.g. nasals (airflow goes through the nasal cavity), vowels (glottal pulse flows freely through the vocal tract), fricatives (airflow has to pass through a narrow channel produced by two articulators placed together), etc.

According to these criterions, each phoneme can present 3 main properties, one per classification criterion. Therefore, if phonemes are classified in groups which share the same three properties, the classification would be too discriminative (too many classes and too less phonemes) and too much training data would be required for an accurate training. Thus, this linguistic point of view will be discarded and a less strict approach will be taken.

Therefore, the classification proposed in this thesis is less restrictive and attempts to classify voice into the groups contained in Table 6.1. This classification is done since the organs involved in producing each of those sounds are similar and the waveform properties share certain attributes. For example, in vowels, the periodic train pulse coming form the glottis does not find any opposition before exiting the articulatory system. Voiced consonants are formed by nasals, lateral sounds and vibrating phonemes. Voiced fricatives are obtained when the glottis train pulse goes through a narrow channel produced by two articulators placed together. Unvoiced fricatives are produced when the airflow is not modulated by the vocal folds. Finally plosives are produced by closing the airflow exit by joining two articulators so great air pressure is accumulated, and releasing them violently so a short sound is produced. Finally a model for removing silence gaps is produced.

Sound Type
Vowels
Voiced consonants
Voiced fricatives
Unvoiced fricatives
Plosives
Silence

Table 6.1 Sounds classification

The speech recogniser will work in phoneme recognition mode. The exact phonetic recognition is not needed since no strict decision is required. Only a good acoustic description of each frame is needed. This can be obtained by training the phoneme recogniser for only one speaker (this speaker does not need to be enrolled in the system) by prompting a given text. If this fixed text is used later for training the speaker recogniser the performance of the phoneme recogniser will be enough to grant a good classification.

This phoneme recogniser will also consider MFCCs as its feature vectors. 12 coefficients will be obtained from an analysis based on 26 frequency bands. Delta parameters will be appended to the 12 basic MFCCs. HMMs will be used to obtain monophone models. In a last step-tying will be used to obtain triphone context-dependent models [36], [34]. This context-dependency makes the system language-dependent.

This method will be tested for a Polish language voice corpus since it has been optimised for it. Later it will be tested with an English language voice corpus to check if the acoustic classification can be generalised to a multi-lingual context.

6.2 Gaussian Mixture Models (GMMs) and Hidden Markov Models (HMMs)

Once voice has been classified using one the methods above, the next step is to obtain a model for each speaker and each voice class. This is the most complex and, at the same time, most important part of a speaker recognition system.

As it was said before, HMMs and GMMs belong to the group of stochastic pattern matching algorithms. In this kind of algorithms voice will be considered as a stochastic process and the models obtained will rely on probabilistic properties of the feature vectors extracted in chapter 5.

GMMs and HMMs are totally independent from each other. In fact, some good ASRs have obtained using only GMMs as it can be checked in [27], whereas their combined used is normally attained to ASRs that rely on speech recognisers [3], [21].

It was mentioned that voice will be considered as a random stochastic process. Therefore, a speaker can be considered as a source that outputs random stochastic vectors. This source will produce observation vectors according to an M-dimensional probability density function (*pdf*). If this *pdf* were known, it could be possible to estimate the speaker who produced a given utterance by using the Bayes theorem as it will be seen later. GMMs excel in this task since they can be used to model any *pdf* shape given a set of training feature vectors. Thus, a set of observation vectors can be used by GMMs to accurately estimate the *pdf* of a stochastic process. Accuracy of GMMs will increase with the amount of available training data. Therefore, a GMM can be considered as a generator of independent feature vectors.

However, since speech is a sequential process, a feature vector is not independent of its predecessor. In fact, the transition from a phoneme to another is not immediate and requires the articulator organs to move from one state to another. HMMs can used to model these transitions and can model the feature vectors evolution along time. This is the dynamics of speech. Using GMMs together with HMMs can offer a good short-term and medium-term modelling of a speaker's speech properties.

In conclusion, the aim of GMMs and HMMs is to find the probability that a speaker i produced a given observation vector O. This can be stated as

$$P\left(SP_i \mid O\right). \tag{6.4}$$

Thus the identification process is to find the argument *i* that maximises (6.4)

$$\arg\max_i \left\{P\left(SP_i \mid O\right)\right\}. \tag{6.5}$$

The Bayes theorem can be applied to (6.5) as that probability can be calculated obtaining

$$\arg\max_i \left\{P\left(SP_i \mid O\right)\right\} = \arg\max_i \left\{\frac{P\left(SP_i \mid O\right)\cdot P(SP_i)}{P\left(O\right)}\right\}. \tag{6.6}$$

The probability $P(SP_i)$ that a speaker produces an utterance is generally the same for all the speakers so it has no impact in (6.6). In addition, if it is considered that all the observation vectors can occur with the same frequency, $P\left(O\right)$ is neither a determinant factor and can also be removed. $P\left(O\right)$ could be interesting in the case of a speaker recognition system where high level speech features are used, however, this is not the case. Thus HMMs and GMMs are parametric stochastic models that attempt to estimate (6.6).

Next sections in this chapter will describe how GMMs and HMMs can be trained, and used for estimating the probability that a speaker produced a given utterance.

6.2.1 Gaussian Mixture Models (GMMs)

Choosing the correct *pdf* shape (gaussian, exponential, logarithmic, etc.) for a process can be a tedious matter. However GMMs are a very powerful *pdf* estimator that can be used to model almost every known *pdf* kind. In speaker recognition systems they are used to obtain the *pdf* with which each speaker generates his feature vectors. Once this *pdf* is calculated, (6.6) can be estimated. However, in this thesis GMMs will be used as a complementary tool for HMMs and its aim will be use to model the feature vectors output distribution.

GMMs consist of a sum of multivariate gaussian functions where each of them is weighted by a mixture coefficient. If $b\left(o\right)$ is the probability distribution of the feature vectors generation process, this *pdf* can be expressed as

$$b\left(o\right) = \sum_{m=1}^{G} \left(c_m \frac{1}{\sqrt{\left(2\pi\right)^M \left|\Sigma_m\right|}} e^{-\frac{1}{2}\left(o-\mu_m\right)'\Sigma_m^{-1}\left(o-\mu_m\right)} \right), \tag{6.7}$$

where each c_m is the *m*-th mixture coefficient, *M* is the dimension of each feature vector, Σ_m is the covariance matrix of the *m*-th mixture and μ_m is the mean of the *m*-th mixture mean vector. Finally, G is the number of mixtures used. Thus, to define a GMM it is necessary to find the optimum values of each c_m, μ_m and Σ_m. . The number of gaussians G is also crucial to obtain an accurate model. It is clear that the mixture coefficients are constrained to sum the unity in order to match *pdf*'s specifications E.g., in Fig. 6.7 a 2-dimensional GMM with 3 mixtures is represented.

All the parameters named in the previous paragraph can be accurately estimated via the *expectation-maximization* (EM) algorithm. This algorithm will be explained in the HMMs section, since it will also be used there. In fact, in this thesis, GMMs parameters estimation will be done during HMMs parameters estimation. If the reader is interested in the EM algorithm for GMMs, reference [20] offers its complete formulae. In [14], a fully detailed description of the EM algorithm and its applications can be found.

Thus, to correctly model a source of observation vectors, the parameters c_m, μ_m and Σ_m must be calculated. In the previous chapter, it was outlined that the feature vector components are highly uncorrelated. Because of this, the covariance matrix Σ_m will have very low values outside its main diagonal. This means that very low cross-covariance values are present. Thus, this matrix can be simplified as a diagonal matrix where all its components are zero except for all those belonging to the main diagonal, this is, the autocovariances.

As it will be seen later, this approximation will reduce the HMMs and GMMs training algorithm complexity.

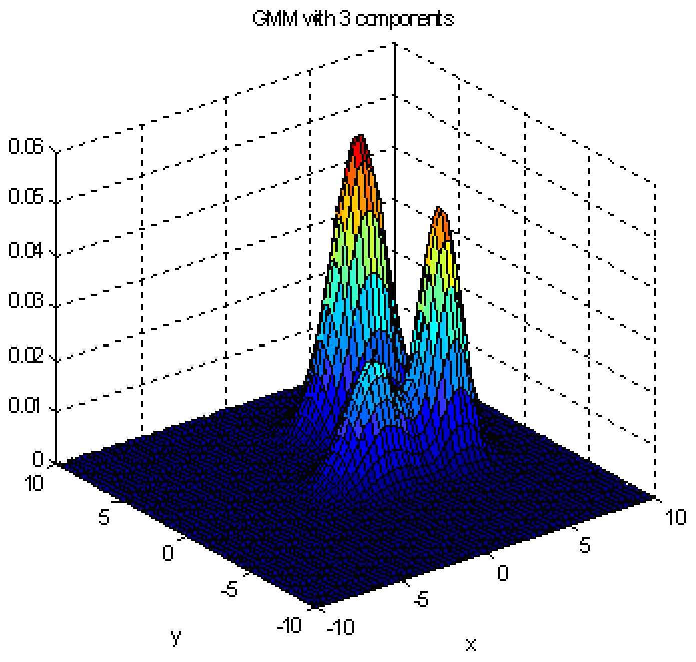


Fig 6.7 GMM pdf with 3 components and dimension 2

6.2.2 Hidden Markov Models

Hidden Markov Models (HMMs) are a powerful tool to model sequences of stochastic data. The speech signal matches this definition, as each speaker produces a temporal sequence of sounds. GMMs are good for modelling the static properties of speech. However, they do not offer any way to model the changing nature of speech. The *pdf* obtained by a GMM is used to model all the feature vectors produced by a given speaker independently of the previous vector generated. HMMs come to solve this problem. GMMs can be thought as stochastic generators of feature vectors since HMMs produce sequences of feature vectors according to a probabilistic finite states machine.

As GMMs do, HMMs are not only used in the speaker recognition context. They are widely used in speech recognition. Out of the speech processing scope, they can be used to model any kind of sequential processes.

In this section, the general theory of HMMs will be explained for the particular case of speaker recognition. For a wider point of view about all the possibilities in speech processing, the reader is referred to [24].

6.2.2.1 General principles of HMMs, notation and particularisation to speech processing

The general structure of a HMM can be observed in Fig. 6.8. It consists of a set of states S_i and a set of transition probabilities from one state to another a_{ij}. The idea is that the system will be flowing from one state to another and every time a state is reached, an observation vector will be generated in each state.

That general structure represented in Fig. 6.8 permits transitions to states which had been previously reached. Since the nature of speech is sequential and changes along time, it is reasonable that transitions to previous states should be forbidden. That is why when HMMs are used in speech processing, *left-to-right* HMMs are used.

In Fig. 6.9 the structure of a *left-to-right* HMM is presented. It does not permit any transition to past states. In addition, the output observations vector *O* produced in each state with a probability b_i is represented. All the theory which will be now described will be based on this *left-to-right* HMM.

The feature vectors of the speech signal are considered as the output of the finite state machine or chain presented in Fig. 6.9. The chain will be studied in certain time instants $t = 1, 2...T$. Every time there is a time instant change (i.e. from $t = 1$ to $t = 2$), the system will transit from the current state S_i to another state S_j with a discrete probability a_{ij}. In addition, every time a new state is entered or re-entered (loops are allowed) an observation vector o_t is generated with a probability density $b_j(o_t)$. Thus, the feature vectors are generated in sequential order by the HMM. The goal of the

HMM training algorithms is to find the optimum parameters of this finite state machine that best match a speaker's speech production process.

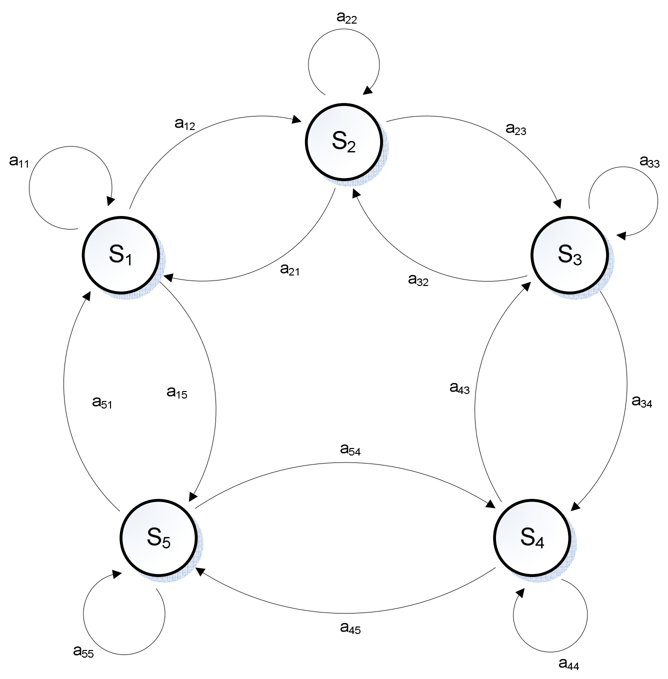


Fig 6.8 Generic Markov chain with five states

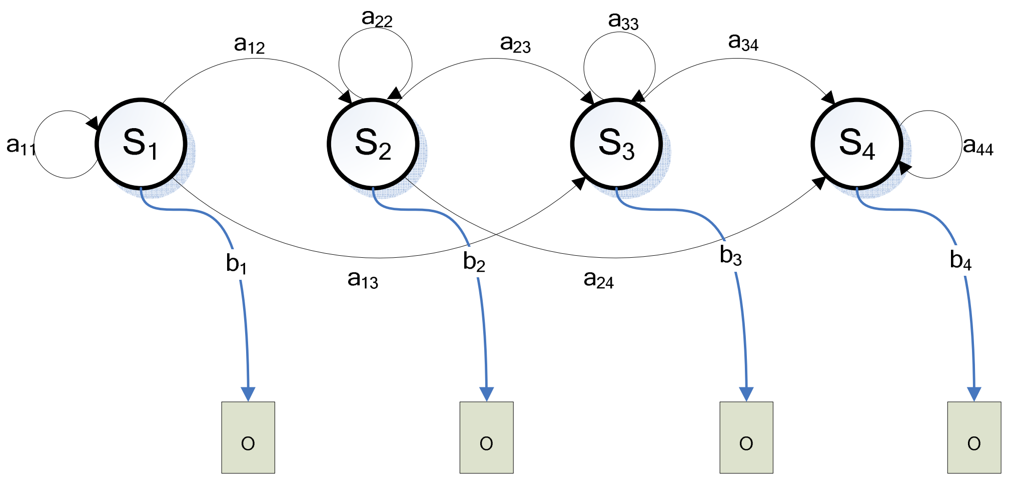


Fig 6.9 Left-right four-state HMM

In order to clarify the main ideas of HMMs, the notation used in this section will be presented before any further mathematical development.

- $\{S_i\}_{i=1\ldots N}$ *set of states.*
  The total number of states in the chain is *N*. In general purpose HMMs, each state generally corresponds to a physical event. In speaker recognition, each state

represents the duration of a frame. The chain will remain in that state during one frame length. Once that time is reached, a transition to another state occurs and another feature vector is generated.

- $O = o_1 ... o_T$ *sequence of observations*

  O is a sequence of observation vectors corresponding to an utterance produced by a speaker. Each feature vector o_i is formed by M components. The total number of observations depends on the number of time slots in which the chain is studied. The total number of time slots is *T*. Since in each time slot there is a state transition, the total number of feature vectors per sequence is *T*.

- a_{ij} *transition from state S_i to state S_j discrete probability.*

  It is important to note that this probability is fixed and does not change along the different time instants described. Formally, the probability a_{ij} is given by

$$a_{ij} = P\left(q_{t+1} = S_j \mid q_t = S_i\right) \quad 1 \le i, j \le N \tag{6.8}$$

  The complete probabilistic description of the model requires the specification of the current state at time t, as well as all the states previously reached. However, in this thesis first order Markov chains will be used. In this case the full-memory system is truncated to only the current and previous states [24]

$$P\left(q_t = S_j \mid q_{t-1} = S_i, q_{t-2} = S_k ...\right) = P\left(q_t = S_j \mid q_{t-1} = S_i\right) \tag{6.9}$$

  The transition probabilities are usually given by a transition probabilities matrix $A = \left\{a_{ij}\right\}_{i,j=1...N}$. The zero components in this matrix represent forbidden transitions. Therefore, the fundamental property of a left-right HMM is that all the components of the matrix A are constrained to

$$a_{ij} = 0, \qquad j < i \tag{6.10}$$

- $b_j\left(o_t\right)$ *probability distribution that a vector observation o_t is outputted when a state S_j is reached.*

  This distribution can be discrete or continuous. This part of HMMs model the static properties of speech. This is, the *pdf* that generates a given feature vector observation. In the previous section, it was described how GMMs are used for this matter. Thus, the output probability distribution at each state will be defined by a GMM. This set of distributions is denoted as $B = \left\{b_j\left(o_t\right)\right\}_{j=1...N}$ where

$$b_j\left(o_t\right) = P\left(\mathrm{o}_t \text{ at } t \mid q_t = S_j\right) \quad 1 \le \mathrm{j} \le \mathrm{N}. \tag{6.11}$$

- $t = 1...T$ current time instant

As it was said before, time is divided into T time slots. Every time a new slot is reached, a state transition occurs and a new feature vector is generated according to the probability distribution given in (6.11).

Then, a HMM is fully described by the transition probabilities matrix A and the output probability distribution set B. In some literature as [24] the initial state probability is also considered, however, in *left-to-right* HMMs the first state is always S_i. This model called $\lambda = (A, B)$ is a valid generator of observation vectors if the parameters A, and B are optimum. In this case, the model λ can be used to generate an observation sequence $O = o_1 ... o_T$ with the following algorithm [24].

1. Set $t = 1$.
2. Generate an observation vector o_t according to the output probability distribution in state S_i , $b_i(o_t)$.
3. Transit to a new state $q_{t+1} = S_j$ according to the state transition probability distribution for state S_i $\{a_{ij}\}_{j=1...N}$.
4. Set $t = t+1$
    a. If $t < T$: return to step 3.
    b. Otherwise: end of procedure.

Thus, a HMM is a generator of feature vectors. If A and B are optimum, the output of this generator will be very close to the real output of a speaker from a probabilistic point of view. Thus, it arises as a good model for a speaker's particular voice. Therefore, some algorithms have to be found in order to obtain the parameters A and B that best match a speaker individual voice properties. Once these values are found, the model can be used to obtain a probability score of if a given utterance or observation sequence was generated by a given speaker.

Thus, as an example, given an observation sequence $O = o_1 o_2 ... o_T$ and the states sequence $X = q_1 ... q_T$, the probability that O was generated by a model λ moving through the sequence X can be calculated as

$$P(O, X \mid SP) = P(O, X \mid \lambda) = \prod_{t=1}^{T} a_{q_{t-1} q_t} b_{q_t}(o_t) \tag{6.12}$$

However, in practice only the observation sequence O is known, while the state sequence X is the unknown or hidden part of the problem. The name Hidden Markov Model comes from this issue. There are some algorithms which try to find the correct state sequence to generate O and will be exposed later.

Next section will describe the three basic problems of HMMs and the possibilities that the solution to these problems offer in the speaker recognition area.

6.2.2.2 The three basic problems of HMMs

There are three basic problems in HMM modelling. The algorithms designed to deal with these problems are a powerful tool to face speech and speaker recognition problems. In [24], these three problems are defined as follows.

- *Problem 1*: Given an observation sequence $O = o_1...o_T$ and a model $\lambda = (A, B)$, how can $P(O|\lambda)$, the probability of the observation sequence, given the model be efficiently computed?

  This is the evaluation problem. Given a model and a sequence of observations, how can the probability that the observed sequence was generated by the model be computed? If the problem is solved for all the available speaker models, a decision based on the highest likelihood score can be taken. In conclusion, solution to problem 1 finds the model which best matches the observation vectors.

- Problem 2: Given the observation sequence $O = o_1...o_T$, and the model $\lambda = (A, B, \pi)$, how can be chosen the optimal state sequence $Q = q_1...q_T$?

  It has been shown before that the states sequence is the hidden part of the model. There is no solution to this problem since there is not a correct state sequence to be found. However, an optimality criterion can be adopted to find the best possible solution. The solution to this problem is very useful in continuous speech recognition systems and also in speaker recognition as it can reduce the computational cost of problem 1.

- Problem 3: How can the model parameters $\lambda = (A, B)$ be adjusted to maximise $P(O|\lambda)$?

  This is the most important problem since it attempts to estimate the model parameters which best model a given speaker. The parameters are obtained at the enrolment stage, where an identified speaker produces a training voice utterance. This utterance is a HMM observation sequence. A training algorithm will calculate the model parameters which best match this observation.

Next sections will describe how these problems can be solved and the mathematical expressions of all the algorithms involved.

6.2.2.3 Solution to Problem 1

Given an observation sequence $O = o_1...o_T$ and all the parameters of the model $\lambda = (A, B)$ belonging to a given speaker, the problem is to compute the probability that the given model produced the observation sequence. In (6.12), it was shown that if the states sequence is known, it is easy to calculate this probability. However, since the states sequence is unknown, one valid approach is to calculate that probability considering all the possible state sequences $Q = q_1...q_T$ of length *T*. This means that to obtain $P(O \mid \lambda)$, the probability $P(O,Q \mid \lambda)$ has to be computed for every possible state sequence Q and the sum of all these probabilities will provide $P(O \mid \lambda)$

$$P(O \mid \lambda) = \sum_{Q} P(O,Q \mid \lambda) = \sum_{Q} \prod_{t=1}^{T} a_{q_{t-1}q_t} b_{q_t}(o_t) \tag{6.13}$$

The computational cost of this algorithm is quite elevated since it is $O(2T \cdot N^T)$ [24]. The explanation is simple: the number of possible state sequences is N^T since for every time slot *t*, there are *N* possible states to be reached and, in total, there are *T* time slots. In addition, the sum in (6.13) requires around 2*T* calculations per term. This computational cost is too high for a real-time application. A faster algorithm is required to compute (6.13). In the next section, the *Forward-Backward Procedure* will be exposed as a method to reduce such algorithmic complexity.

6.2.2.3.1 Forward-Backward Procedure

A recursive algorithm to calculate (6.13) will be proposed in this section. In practise, only one of the forward or backward parts of this algorithm are required to compute (6.13). Anyway, both will be explained since both parts of the algorithm will take part in the resolution of problem 3.

- Forward part:

This part of the algorithm is based on the probability $\alpha_t(i)$ defined as

$$\alpha_t(i) = P(o_1...o_t, q_t = S_i \mid \lambda), \tag{6.14}$$

the probability that the partial observation sequence $o_1...o_t$ was observed and the current state at time *t* is S_i, given the model λ. All the procedures described below are done considering a general HMM. They will be latterly particularised for the case of a *left-to-right* HMM.

$\alpha_1(i)$ can be computed as follows considering the case of a general HMM where π_i is the probability that the first state of the chain is S_i.

$$\left.\begin{array}{l} \alpha_1(1) = \pi_1 b_1(o_1) \\ \alpha_1(2) = \pi_2 b_2(o_1) \\ \vdots \\ \alpha_1(N) = \pi_N b_N(o_1) \end{array}\right\} \alpha_1(i) = \pi_i b_i(o_1)$$

Then, in order to calculate $\alpha_2(i)$, it must be clear that to enter state S_i at time *t*=2, the previous state at time *t*=1 can be anyone of the states set. Thus,

$$\left.\begin{array}{l}\alpha_2(1)=\left[a_{11}\pi_1 b_1(o_1)+a_{21}\pi_2 b_2(o_1)+\ldots+a_{N1}\pi_N b_N(o_1)\right]b_1(o_2)\\ \alpha_2(2)=\left[a_{12}\pi_1 b_1(o_1)+a_{22}\pi_2 b_2(o_1)+\ldots+a_{N2}\pi_N b_N(o_1)\right]b_2(o_2)\\ \vdots\\ \alpha_2(N)=\left[a_{1N}\pi_1 b_1(o_1)+a_{2N}\pi_2 b_2(o_1)+\ldots+a_{NN}\pi_N b_N(o_1)\right]b_2(o_2)\end{array}\right\}$$

so,

$$\left.\begin{array}{l}\alpha_2(1)=\left[\sum_{i=1}^{N}\pi_i b_i(o_1)a_{i1}\right]b_1(o_2)=\left[\sum_{i=1}^{N}\alpha_1(i)a_{i1}\right]b_1(o_2)\\ \alpha_2(2)=\left[\sum_{i=1}^{N}\pi_i b_i(o_1)a_{i2}\right]b_2(o_2)=\left[\sum_{i=1}^{N}\alpha_2(i)a_{i2}\right]b_2(o_2)\\ \vdots\\ \alpha_2(N)=\left[\sum_{i=1}^{N}\pi_i b_i(o_1)a_{iN}\right]b_N(o_2)=\left[\sum_{i=1}^{N}\alpha_2(i)a_{iN}\right]b_N(o_2)\end{array}\right\}\Rightarrow\alpha_2(j)=\left[\sum_{i=1}^{N}\alpha_1(i)a_{ij}\right]b_j(o_2)$$

then, a recursive algorithm to compute (6.13) can be inducted from the previous expression.

1) Initialisation.

$$\alpha_1(i)=\pi_i b_i(o_1) \tag{6.15}$$

The algorithm is initialised with the probability of the first observation generated being S_i the first state of the sequence.

2) Induction.

$$\alpha_{t+1}(j)=\left(\sum_{j=1}^{N}\alpha_t(i)a_{ij}\right)b_j(o_{t+1}),\ 1\le \mathrm{t}\le \mathrm{T\text{-}1} \qquad 1\le \mathrm{j}\le \mathrm{N} \tag{6.16}$$

α is calculated for all the states and all the time instants and its final values are stored. The sum in (6.16) reflects that it is possible to reach the state S_j from any of the *N* states. A graphical explanation of this process can be found in [24]. In Fig. 6.10 it is shown how the state S_j can be reached in time *t*+1 from any of the other *N* states in time *t*.

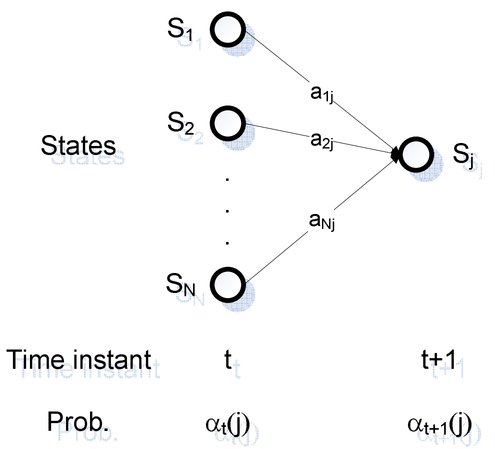


Fig 6.10 Table of possible transitions to state S_j with the probability weights of those transitions required to compute α_{t+1}.

3) Termination:

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i) \qquad (6.17)$$

According to the definition of α in (6.14), $\alpha_T(i) = P(o_1 ... o_T, q_T = S_i | \lambda)$, therefore, the probability $P(O|\lambda)$ is the sum of the forward probabilities.

Now, the computation cost has reduced from $O(2T \cdot N^T)$ to $O(N^2 T)$. The result of the operation is the same. However, the number of operations required is drastically reduced. In addition, if the case of a *left-to-right* HMM is considered,

$$\pi_i = \begin{cases} 1 & \text{if } i = 1 \\ 0 & \text{otherwise} \end{cases} \qquad (6.18)$$

Thus, in time $t = 1$

$$\alpha_1(j) = \begin{cases} b_j(o_1) & \text{if } j = 1 \\ 0 & \text{otherwise} \end{cases} \qquad (6.19)$$

Finally, in time slot *t*=2, only transitions coming from state S_1 are allowed. Thus,

$$\alpha_2(j) = \begin{cases} a_{1j} b_1(o_1) b_j(o_2) & \text{if } 1 \le j \le 2 \\ 0 & \text{otherwise} \end{cases} \qquad (6.20)$$

This reduces also the time complexity algorithm. It will be seen that the case of *left-to-right* HMMs is greatly interesting to solve problem 2. This is to find the hidden correct sequence of states in the chain that produced a given observation. In fact, this reduction is intuitive since the possible number of states that can be reached from another one is drastically reduced.

- Backward part:

Problem 1 can be solved by the forward part of the forward-backward procedure or, also, by the backward part. This algorithm is completely analogue to the forward one except that the recursive algorithm starts at the end of the time sequence. Therefore, the basis probability of the algorithm is $\beta_t(i)$ and it is defined as

$$\beta_t(i) = P(o_{t+1}...o_T \mid q_t = S_i, \lambda) \tag{6.21}$$

This is the probability of the final partial observation sequence from $o_t...o_T$ and being in state S_i at time t, given the model λ. The induction to obtain the following recursive algorithm is similar to that one used to get the forward procedure and will not be described.

1) Initialisation

$$\beta_T(i) = 1, \quad 1 \le \mathrm{i} \le \mathrm{N} \tag{6.22}$$

2) Induction

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(o_{t+1}) \beta_{t+1}(j), \quad T\text{-}1 \ge t \ge 1 \qquad 1 \le i \le N \tag{6.23}$$

3) Termination

$$P(O \mid \lambda) = \sum_{i=1}^{N} \beta_1(i) \tag{6.24}$$

6.2.2.4 Solution to problem 2

This problem has not an analytically exact solution. In this case, a somehow optimum solution has to be found. The problem of finding the correct states sequence is reduced to the problem of defining what the optimal state sequence is. The most commonly used criterion is to choose the states q_t which are individually most likely [24]. This optimality criterion maximises the expected number of correct individual states.

It was shown in the previous section how solution to problem 1 of HMMs can be used to estimate the probability that a given observation sequence was produced by a given speaker model. However, the forward-backward procedure will not be used for that. Instead, Viterbi decoding will be used to solve problem 2 and find the optimum sequence of states that generated a given observation vector. The reason is that this approach generalises easily to the continuous speech case.

The solution to this problem according to the chosen definition of correct state sequence requires the inclusion of the probability

$$\gamma_t(i) = P(q_t = S_i \mid O, \lambda) \tag{6.25}$$

which is the probability of being in state S_i at time *t* given the observation sequence *O* and the model parameters λ.

This probability can be expressed in terms of the forward and backward probabilities

$$P(q_t = S_i \mid O, \lambda) = \frac{P(q_t = S_i, O \mid \lambda)}{P(O \mid \lambda)} \tag{6.26}$$

$$\left.\begin{aligned} \alpha_t(i) &= P(o_1 \ldots o_t, q_t = S_i \mid \lambda) \\ \beta_t(i) &= P(o_{t+1} \ldots o_T \mid q_t = S_i, \lambda) \end{aligned}\right\} \gamma_t(i) = \frac{\alpha_t(i)\beta_{t+1}(i)}{\sum_{i=1}^{N} \alpha_t(i)\beta_t(i)} \tag{6.27}$$

Using (6.25) it is possible to find the most likely state q_t at time t by solving

$$q_t = \underset{1 \le i \le N}{\arg\max}\left(\gamma_t(i)\right), \qquad 1 \le t \le T \tag{6.28}$$

The solution given by (6.28) can be problematic since the resulting state sequence might be incorrect. For example, (6.28) could include some forbidden state transition since it only determines the most likely state at every instant, without regard to the probability if occurrence of sequences of states.

To solve the given conflict, one can think in modifying the optimality criterion. For example, instead of maximizing only the probability of one single state q_t, the same can be done to pairs $q_t q_{t+1}$ or triples $q_t q_{t+1} q_{t+2}$ of states, etc. However, the most commonly used criterion is to find the single best state sequence. To do this, Andrew J. Viterbi proposed the following algorithm [32][9] which is based on dynamic programming methods.

6.2.2.4.1 The Viterbi Algorithm

The target is to find the single best state sequence $Q = q_1 \ldots q_T$, for the given observation sequence $O = o_1 \ldots o_T$. For that, $\delta_t(i)$ is defined as

$$\delta_t(i) = \max_{q_1 \ldots q_{t-1}} P(q_1 \ldots q_t = i, o_1 \ldots o_t \mid \lambda) \tag{6.29}$$

It represents the best score or highest probability along a single path, at time t. By induction, $\delta_{t+1}(j)$ can be calculated as:

$$\delta_{t+1}(j) = \left(\max_i \delta_t(i) a_{ij}\right) b_j(o_{t+1}) \tag{6.30}$$

To actually retrieve the state sequence, it is necessary to keep track of the argument which maximised (6.30), for each *t* and each *j*. Thus, both values will stored in the array $\psi_t(j)$. The complete procedure to obtain the best state sequence can be stated as follows.

1) Initialisation:

$$\delta_1(i) = \pi_i b_i(o_1) \tag{6.31}$$

$$\psi_1(i) = 0 \tag{6.32}$$

2) Recursion:

$$\delta_t(j) = \max_{1 \le i \le N} \left(\delta_t(i) a_{ij} \right) b_j(o_t) \tag{6.33}$$

$$\psi_t(j) = \arg\max_{1 \le i \le N} \left(\delta_{t-1}(i) a_{ij} \right) \tag{6.34}$$

(6.33) and (6.34) are constrained to $2 \le t \le T,\ \ 1 \le j \le N$.

3) Termination:

$$P^* = \max_{1 \le i \le N} \left(\delta_T(i) \right) \tag{6.35}$$

$$q_T^* = \arg\max_{1 \le i \le N} \left(\delta_T(i) \right) \tag{6.36}$$

4) Path (state sequence) backtracking:

$$q_t^* = \psi_{t+1}\left(q_{t+1}^*\right),\ \ \text{t=T-1...1} \tag{6.37}$$

Thus, by using Viterbi decoding, it is possible to find the optimum state sequence that generated a given vector observation and, thus, obtain the most probable model that generated the observation.

In the case of a *left-to-right* HMM, this search is reduced to the lattice in Fig. 6.11. In that figure, the initial and last states are not-emitting as it will be seen in the HTK implementation of HMMs. The optimum path (represented in continuous trace) to generate the observation sequence $O = o_1 o_2 o_3 o_4$ is calculated by maximising (6.33) and (6.34) every step. The total likelihood of the model for a given observation can be obtained by keeping a track of $\delta_t(i)$ and finally apply (6.35).

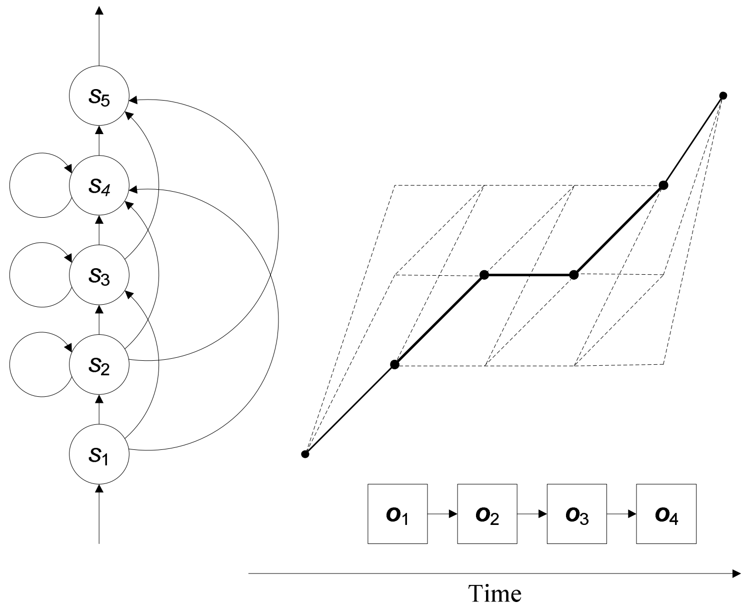


Fig 6.11 Search of the best path with Viterbi algorithm.

6.2.2.5 Solution to problem 3

This is the most difficult and important task in HMMs. There is not an analytical solution to the problem and, therefore, an optimal solution has to be reached by using a parameter re-estimation procedure based on the Baum-Welch iterative procedure.

The parameters of the model $\lambda = (A, B)$ will be estimated in order to maximize the probability $P(O \mid \lambda)$ that a given training observation vector was generated by that model. This is the training stage of the pattern matching algorithm. If the parameters are optimal enough, then a given speaker can be represented by this model λ.

The Baum-Welch re-estimation procedure can be found in [24] and [34]. It is based on the forward and backward probabilities described in section 6.2.2.3.1. Next section will present this algorithm.

6.2.2.5.1 Baum-Welch re-estimation procedure

Problem 1 attempts to adjust the parameters of the model to maximise the probability of the observation given the model. In the Baum-Welch procedure, the forward and backward probabilities are used as well as the probability $\xi_t(i, j)$ which is defined as

$$\xi(i, j) = P(q_t = S_i, q_{t+1} = S_j \mid O, \lambda) \tag{6.38}$$

This is the probability that given an observation sequence *O* and the model parameters, the system is in state S_i at time *t* and in state S_j at time *t*+1. This probability can be re-written in terms of the forward and backward basis probabilities accordingly to its respective definitions in (6.14) and (6.21) as

$$P\left(q_t = S_i, q_{t+1} = S_j \mid O, \lambda\right) = \frac{P\left(q_t = S_i, q_{t+1} = S_j, O \mid \lambda\right)}{P(O \mid \lambda)} \tag{6.39}$$

$$\left.\begin{aligned} &\alpha_t(i) = P(o_1 \dots o_t, q_t = S_i \mid \lambda) \\ &\beta_{t+1}(j) = P\left(o_{t+2} \dots o_T \mid q_{t+1} = S_j, \lambda\right) \\ &P\left(o_{t+1}, q_{t+1} = S_j \mid q_t = S_i, \lambda\right) = a_{ij} b_j(o_{t+1}) \end{aligned}\right\} \xi_t(i,j) = \frac{\alpha_t(i)\beta_{t+1}(j) a_{ij} b_j(o_{t+1})}{\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_t(i)\beta_{t+1}(j) a_{ij} b_j(o_{t+1})} \tag{6.40}$$

If $\xi(i,j)$ is summed for all the possible states S_j, $\gamma_t(i)$ is obtained

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i,j) \tag{6.41}$$

If $\gamma_t(i)$ is summed over time, accordingly to (6.25), the result measures the number of times the state S_i is visited. In addition, if the term $t = T$ is excluded from this summation, the average number of transitions made with origin in state S_i is obtained.

$$\sum_{t=1}^{T-1} \gamma_t(i) = \text{expected number of transitions from } \mathrm{S}_i \tag{6.42}$$

If the same is done to $\xi_t(i,j)$, an estimation of the number of transitions from S_i to S_j is obtained.

$$\sum_{t=1}^{T-1} \xi_t(i,j) = \text{expected number of transitions from } \mathrm{S}_i \text{ to } \mathrm{S}_j \tag{6.43}$$

Then, a valid method for re-estimation of the parameters of an HMM can be given.

- $\hat{a}_{ij}$ is the ratio between the expected number of transitions from state S_i and state S_j, and the number of transitions from state S_i.

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \tag{6.44}$$

- $\hat{b}_j(o_t)$ is a GMM as it had been stated previously. Hence, the parameters that have to estimated are each mixture weight c_m, the mean vector μ_m and the covariance matrix Σ_m. Mixture weights can be calculated in the same way the A matrix is obtained. It is only necessary to split each GMM into a set of one-dimensional Gaussians and assign each Gaussian to a substate as it is stated in Fig. 6.12. Thus, these parameters are obtained as

$$\hat{c}_{jk} = \frac{\sum_{t=1}^{T} \gamma_t(j,k)}{\sum_{t=1}^{T}\sum_{k=1}^{M} \gamma_t(j,k)} \tag{6.45}$$

$$\hat{\mu}_{jk} = \frac{\sum_{t=1}^{T} \gamma_t(j,k) o_t}{\sum_{t=1}^{T} \gamma_t(j,k)} \tag{6.46}$$

$$\hat{\Sigma}_{jk} = \frac{\sum_{t=1}^{T} \gamma_t(j,k)(o_t - \mu_{jk})(o_t - \mu_{jk})'}{\sum_{t=1}^{T} \gamma_t(j,k)} \tag{6.47}$$

where $\gamma_t(j,k)$ is the probability of being in state S_j at time *t* with the *k*-th mixture component accounting for o_t.

$$\gamma_t(j,k) = \left(\frac{\alpha_t(j)\beta_t(j)}{\sum_{j=1}^{N} \alpha_t(j)\beta_t(j)} \right) \left(\frac{c_{jk} N(o_t, \mu_{jk}, \Sigma_{jk})}{\sum_{m=1}^{M} c_{jm} N(o_t, \mu_{jm}, \Sigma_{jm})} \right) \tag{6.48}$$

The explanation for these formulas is given in [24]. (6.45) can be stated as the ratio between the expected number of times the system is in state S_j using the *k*-th mixture component, and the expected number of times the system is in state S_i.

In (6.46) , each numerator term is weighted by the observation, thereby giving the expected value of the portion of the observation vector accounted for by the *k*-th mixture component. Similar interpretation can be made for (6.47)

The general theory of HMMs has been explained. In next section, the HTK particular implementation of HMMs will be described. It will be done from a more practical point of view in order to introduce the reader to the real procedure carried out in the speaker recognition system.

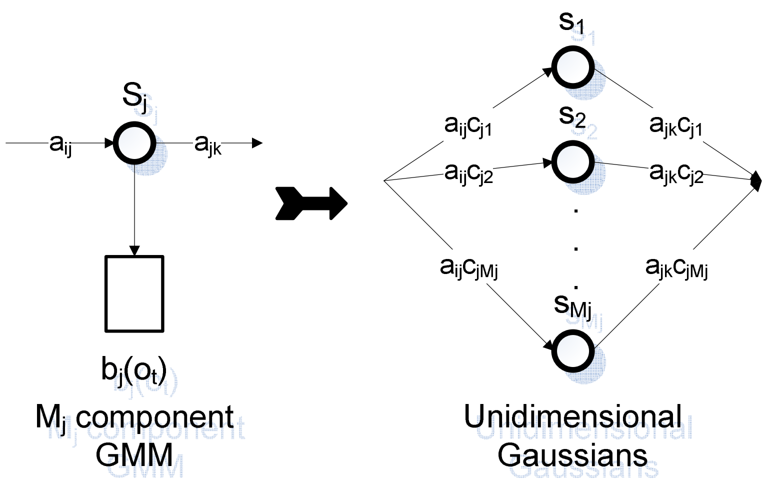


Fig 6.12 Decomposition of a HMM with *M*–dimensional GMM output probability specification into 1-dimensional gaussian output substates

6.3 HTK Toolkit

HTK is a toolkit designed for building Hidden Markov Models. It is specially intended to work in the speech recognition context although it can be adapted to the speaker recognition task. Only the most important characteristics of HTK which affect to this thesis will be described.

On a first approach, HTK is divided into several modules. Each module handles one of the multiple speech processing, pattern matching (only HMM) and language modelling tasks which are needed for speech recognition. In the case of speaker recognition, many of these functionalities can be ignored. It is only important to have in mind the way HMMs are defined, re-estimated and then used for testing. In addition, tools for speech processing can be interesting.

Next sections will describe the procedures needed to train a set of HMMs and how these models are used later in the testing stage.

6.3.1 HMMs Definition

HMM definition in HTK is related to the HMM initialisation stage. In this step, HMMs for each speaker are defined. This definition includes the number of states, output probability specification, initial means and variances, transition probability matrix, and the HMM architecture (like *left-to-right* HMMs). In fact, only the HMM architecture is important, since all the numbers assigned here will be replaced in the initialisation stage.

There is an important particularity in the way HMMs are defined. The first and last states of each HMM are not emitting. This means that these states cannot generate any observation. This is done to facilitate the construction of composite models including two or more HMMs. This singularity provokes a little modification in the formulae used to perform Baum-Welch parameter re-estimation or Viterbi decoding. Since this modification does not suppose a big change in the concept of Baum-Welch or Viterbi decoding. The reader is referred to [34] for the complete formulation used in HTK.

6.3.2 HMMs Initialisation

HTK implements two possible methods to initialise the HMM parameters before the re-estimation stage. Once, the HMM architecture is defined, there are two possible ways to obtain the initial values of each HMM.

The first approach is make all states equal and move forward to parameter re-estimation. This means that the first iteration in the training process will rely on a uniform segmentation. This process is normally used in speech recognition when no labelled data is available. In speaker recognition, the data is fully labelled (in the terms of who was the speaker that produced the data). In this case another more powerful approach can be taken.

The second approach needs some assumptions to be made. Each training sample can be thought as a sequence of observation vectors which were generated by a HMM. The target is to find the best possible parameters for this HMM. If the state that generated each observation vector was known, it would be easy to calculate the initial means and variances for each state. It is only necessary to associate each observation vector to the state which generated it and average its means and variances. In addition, the transition matrix can also be initially estimated just by counting the number of time slots that each state was occupied. The only problem is to find the state sequence that generated each observation sequence. This can be achieved using Viterbi training (its blocks diagram can be seen in Fig. 6.13). Firstly, the Viterbi algorithm finds the most likely state sequence that generated each training example and the HMM parameters are estimated as it was exposed above. As a side-effect, the log likelihood of the training data will be computed. Then, the whole process can be repeated until no further increment in the global likelihood is obtained.

This process requires some initial HMM parameters to get started. Firstly, the data is uniformly segmented and each segment is successively associated with successive states. This only makes sense if the HMM is left-to-right, and this is the case.

To deal with GMMs, the training vectors are associated with the gaussian component with the highest likelihood. The number of vectors associated with each component can be used to estimate each mixture weights. In the uniform segmentation stage, a *k*-means clustering algorithm is used to cluster the vectors within each state.

The whole Viterbi training formulae will be exposed in next section.

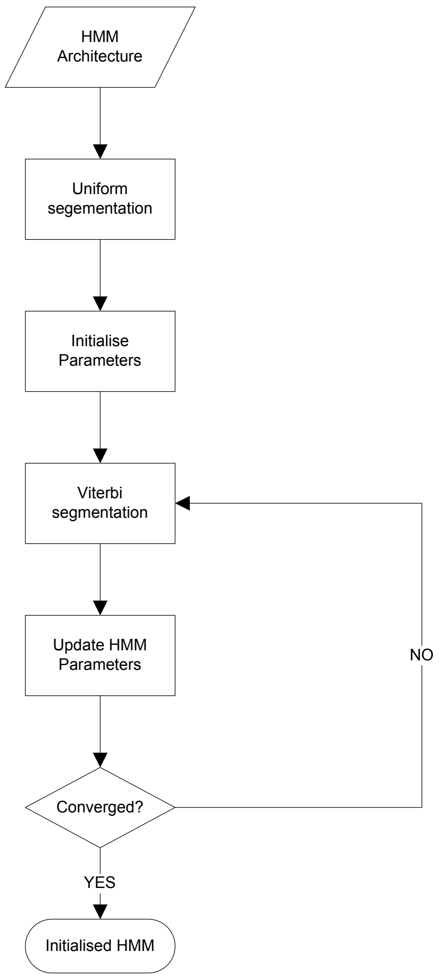


Fig 6.13 Viterbi training procedure

6.3.2.1 Viterbi Training

A set of training observations $\{O_r\}_{r=1...R}$ is used to estimate the parameters of a single HMM by iteratively computing Viterbi alignments. In this case it is used for parameters initialisation, so the first iteration will use a uniform segmentation (i.e., each training observation is divided into *N* equal segments, being *N* the number of HMM states) instead of Viterbi segmentation.

Apart from the first iteration on a new model, each training sequence *O* is segmented using a state alignment procedure which results from maximising the expression. Please note that the formulation is similar to the one used for solving problem 2 in HMMs. However, a few changes are made considering the case of *left-to-right* models and the first and last non-emitting states in the HTK implementation.

$$\delta_T(N) = \max_i \delta_T(i) a_{iN}, \quad 1<i<N \tag{6.49}$$

where

$$\delta_t(j) = \left(\max_i \delta_{t-1}(i) a_{ij}\right) b_j(o_t) \tag{6.50}$$

with initial conditions given by

$$\begin{aligned}\delta_1(1) &= 1 \\ \delta_1(j) &= a_{1j} b_j(o_1) \quad 1<j<N\end{aligned} \tag{6.51}$$

As it was said before, a track the total number of transitions from a state S_i to a state S_j performed by the maximisation given can be kept. This is done with the variable A_{ij} which contains the total number of transitions from state S_i to state S_j in performing the maximisations in (6.49) and (6.50). Then, the transition probabilities can be estimated from A_{ij} as follows:

$$\hat{a}_{ij} = \frac{A_{ij}}{\sum_{k=2}^{N} A_{ik}} \tag{6.52}$$

The sequence of states which maximises (6.49) implies an alignment of training data observations with states. Each state can have various gaussian mixture components and therefore, the observations must be aligned to these components. This is done by associating each observation with the mixture component with the highest probability or by using clustering to allocate each observation to one of the mixture components.

In both cases, each observation is associated with a single mixture component. This association can be represented by the indicator function $\varphi_{jm}^{r}(t)$:

$$\varphi_{jm}^{r}(t) = \begin{cases} 1 & \text{if } \mathrm{o}_t^r \text{ is associated with mixture component } m \text{ of state } S_j \\ 0 & \text{otherwise} \end{cases} \tag{6.53}$$

The means and variances are the estimated using simple averages:

$$\hat{\mu}_{jm} = \frac{\sum_{r=1}^{R}\sum_{t=1}^{T_r} \varphi_{jm}^{r}(t)\, o_t^r}{\sum_{r=1}^{R}\sum_{t=1}^{T_r} \varphi_{jm}^{r}(t)} \tag{6.54}$$

$$\hat{\Sigma}_{jm} = \frac{\sum_{r=1}^{R}\sum_{t=1}^{T_r} \varphi_{jm}^{r}(t)\left(o_t^r - \hat{\mu}_{jm}\right)\left(o_t^r - \hat{\mu}_{jm}\right)'}{\sum_{r=1}^{R}\sum_{t=1}^{T_r} \varphi_{jm}^{r}(t)} \tag{6.55}$$

As it was said before, the mixture weights are obtained from the number of observations associated to each component.

$$c_{jm} = \frac{\sum_{r=1}^{R}\sum_{t=1}^{T_r}\varphi_{jm}^{r}(t)}{\sum_{r=1}^{R}\sum_{t=1}^{T_r}\sum_{l=1}^{M}\varphi_{jl}^{r}(t)} \tag{6.56}$$

6.3.3 HMM parameter re-estimation

The parameters of the models initialised in the previous stage are now re-estimated using the Baum-Welch Forward-Backward procedure explained is section 6.2.2.5.1. The difference between Viterbi and Baum-Welch training is that Viterbi makes a hard decision as to which state each training vector was generated by. On the other hand, Baum-Welch makes a soft decision. Since limits between phonemes are not hard, soft training is more adequate.

The flowchart for Baum-Welch training is shown in Fig. 6.14. HTK allows two kinds of training: isolated unit re-estimation and embedded training. In isolated unit re-estimation, each modelled is individually re-estimated by using the standard formulae applied to solve problem 3 in HMMs, whereas embedded training offers a way to re-estimate all the models in one single pass.

Embedded training simultaneously updates all the HMMs in a system using all of the training data. In this training mode, HTK reads the label files for each training file and constructs a composite HMM which spans the whole utterance. This means that the different models for voiced or unvoiced sounds per speaker are concatenated following the relation in the label files. Then, forward-backward procedure is applied to this composite HMM . When all the training files are processed, the new parameter estimates are formed from the weighted sums and the updated HMM set is output. In this thesis isolated unit re-estimation will be used. Embedded training is more adequate for speech recognition.

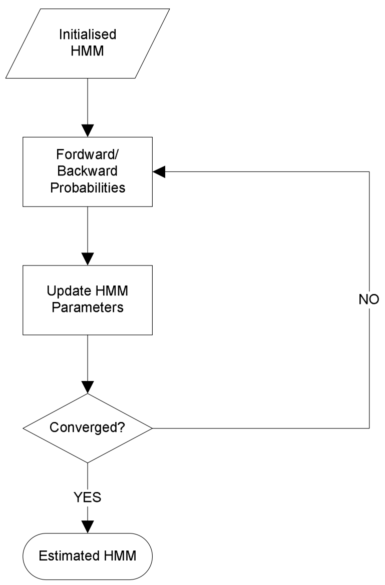


Fig 6.14 Baum-Welch re-estimation procedure

6.3.4 Viterbi decoding

The Viterbi algorithm will be used to find the best path along each HMM for the generation of a given speech utterance. In HTK, the *Token Passing Model* implementation of Viterbi decoding is used [34].

In the token passing model, each state S_j of a HMM at time *t* holds a single moveable token which contains the logarithm of the partial probability given by (6.29). This token then represents a partial match between the observation sequence $O = o_1 ... o_T$ and the model subject to the constraint that the model is in state S_j at time *t*. The path-searching recursive algorithm represented by (6.33) and (6.34) is replaced by the equivalent *Token Passing* algorithm that is executed at each time frame *t*. The key steps in this algorithm are

1. Pass a copy of every token in state S_i to all connecting states, incrementing the log probability of the token by $\log(a_{ij}) + \log(b_i(o_t))$.
2. Examine the tokens in every state and discard all but the token with the highest probability.

This algorithm permits calculating the best path along all the available models and, therefore, obtaining the most likely model that generated a sequence of observations.

The recognition result is outputted in a transcription file. Such file contains a list of frames or groups of frames and the most likely speaker who produced them. Then it is possible to compute the utterance percentage that was produced by each speaker. In Fig. 6.15 the result of the recognition for one single utterance is represented. Each bar represents the estimated percentage of the utterance that was produced by each speaker. It is clear that the most likely speaker is the second one since it is estimated that over the 45% of the utterance as produced by him. This likelihood score will be used in the next chapter by the decision algorithm to predict the speaker that produced the utterance. It must be clear also that the speaker recogniser is utterance oriented. This means, that each utterance is constrained to have been produced by only one speaker.

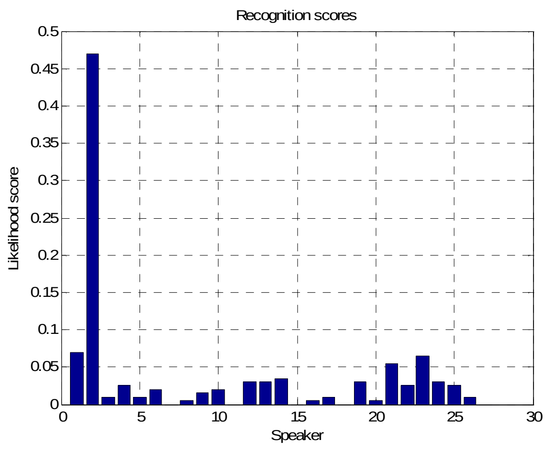


Fig 6.15 Recognition result for a single utterance

In next chapter this likelihood scored obtained from the estimated percentage of the utterance that was spoken by each speaker will be used to give a decision on who was the speaker that produced the utterance. Whether he is an impostor or not and, in the last case, provide speaker's identity.

7 DECISION

The last part of an ASR is the decision making algorithm. The likelihood score for each speaker will be used to determine if the speaker is an impostor or an unknown user. In the last case, speaker's identity should also be outputted by the decision making subsystem.

This decision will be done in terms of a given threshold. If a measure obtained from the likelihood scores outputted by the pattern matching subsystem is higher than that threshold, the utterance will be estimated as having been produced by an authorised speaker. Otherwise, the utterance will be classified as an identity phishing attempt.

Thus, the performance of an ASR can be measured in terms of its *false alarm* and *miss* probabilities. The *false alarm* probability is the probability that an utterance produced by an authorised speaker is classified as having been produced by an impostor. On the other hand, the term *miss* probability refers to the probability that an utterance produced by an impostor is confused with an authorised speaker model and thus, the impostor is granted access to the system.

The choice of the threshold will be a trade-off between a very safe but difficult to access system or the opposite, this is, high false alarm and low miss probabilities. In order to minimise both probabilities, the threshold must be compared to an optimum measure obtained from the likelihood scores. A simple maximum-likelihood (ML) approach can be taken. In the ML approach, the speaker with the highest likelihood score is selected and his likelihood score is compared with the threshold. However, it will be shown later that this is not the optimum approach.

Decision algorithms can rely on two main strategies. The first one is to base the decision only on the scores from the speakers' models. The second one is to use an additional impostor model which models all the possible impostors who can try to access the system. This kind of model is called *universal background model* (UBM). Obtaining a good background model requires a huge amount of training data [28] of about two hours with speech representing all the acoustical variety present in the target population. In this thesis, such a large database is not available. However, this approach will also be taken despite the fact that training data is limited.

In addition, the distribution of likelihood scores will vary depending on many factors as the ambient noise, the microphone used or even the speaker who is being modelled. In these cases a-posteriori normalisation techniques can be used to improve the system performance [17]. These techniques generally consist in using some of the training data to perform a training-by-testing operation. This is, the likelihood distribution obtained by testing some training data is used to adjust the threshold. This threshold can be different for each speaker. Since in this thesis all the data is recorded in clean atmosphere and with high quality microphones, these normalisation techniques are not considered. For further information the reader is referred to [12]. Furthermore, the use of cepstral mean normalisation is sufficient to avoid microphone distortions.

Following sections will describe the methods used for estimating speaker's identity using the likelihood scores corresponding to a spoken utterance.

7.1 Without background model

In the case that no background model is available decision must be based on the likelihood scores for each speaker model. Once more, the simplest way to do this is to consider the ML case. The speaker *i* with the highest probability is selected and his likelihood score is compared with a given threshold. If this score is higher than the threshold the user is accepted with identity *i*. Otherwise, he is considered an impostor.

However, this method is not the most accurate. In Fig. 7.1, the likelihood scores of two different tests are represented. The graphic on the left corresponds to an utterance produced by a known speaker called "Speaker 1". The one on the right belongs to an impostor. In the case of a speaker who has been enrolled in the system, the likelihood score for any of his utterances will be very high for his model and low for the rest. However, if the speaker is an impostor, the likelihood score will be more distributed since the system is unable to match his speech with an available model.

This property can be exploited to obtain a decision rule. In fact, in this thesis, when no background model is available, the decision algorithm will measure the distribution of likelihood scores. If this distribution is not concentred on a given speaker, it means that the utterance was produced by an impostor.

Two methods are proposed to measure the concentration degree of the likelihood score distribution. The first method is based on a very fast comparison of the highest score in the likelihood against the mean value of the rest of the speakers' scores. The second one uses Shannon information entropy measure. In the two following sections both methods will be described.

7.1.1 Fast likelihood degree of distribution estimation

To measure the likelihood degree of distribution in a very fast way, the first assumption to make is that the number of speakers is large enough. Then, the algorithm represented in Fig. 7.2 can be applied.

A candidate is chosen within all the speakers. This candidate has the highest likelihood score among the others. Candidate's ID and likelihood value are stored. Next step comprises obtaining the likelihood score of the three other speakers with the highest score associated. The mean value of these three scores is calculated. Finally, this mean divides the candidate's score value,

$$r = \frac{candidate's\ likelihood\ score}{\mathrm{mean}\left(3\ highest\ likelihood\ scores\ excluding\ the\ candidate\right)} \tag{7.1}$$

(7.1) gives a very fast approximation of the degree of dispersion of the likelihood score distribution. The higher value of *r*, the more probable the utterance was produced by speaker *i*. The lower value of *r*, the more dispersed likelihood score distribution is found. In this case, the utterance is more likely produced by an impostor.

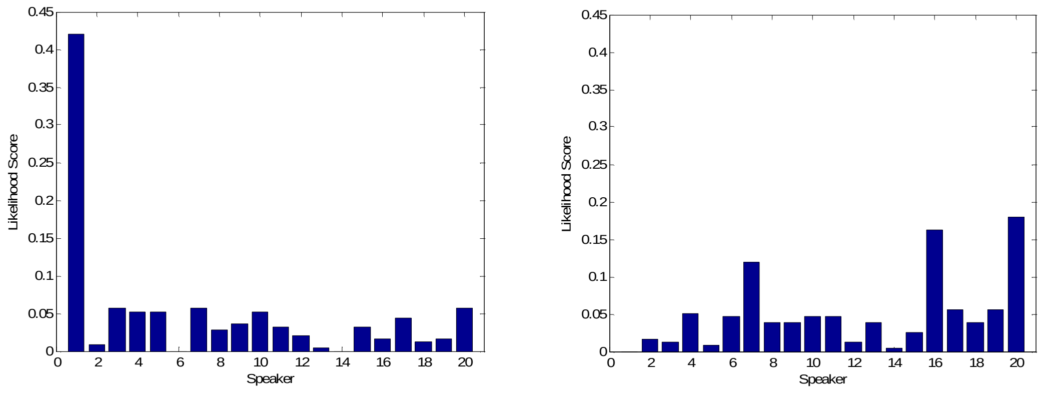


Fig 7.1 On the left, likelihood score for an utterance produced by "Speaker 1". On the right, likelihood score for an utterance produced by an impostor

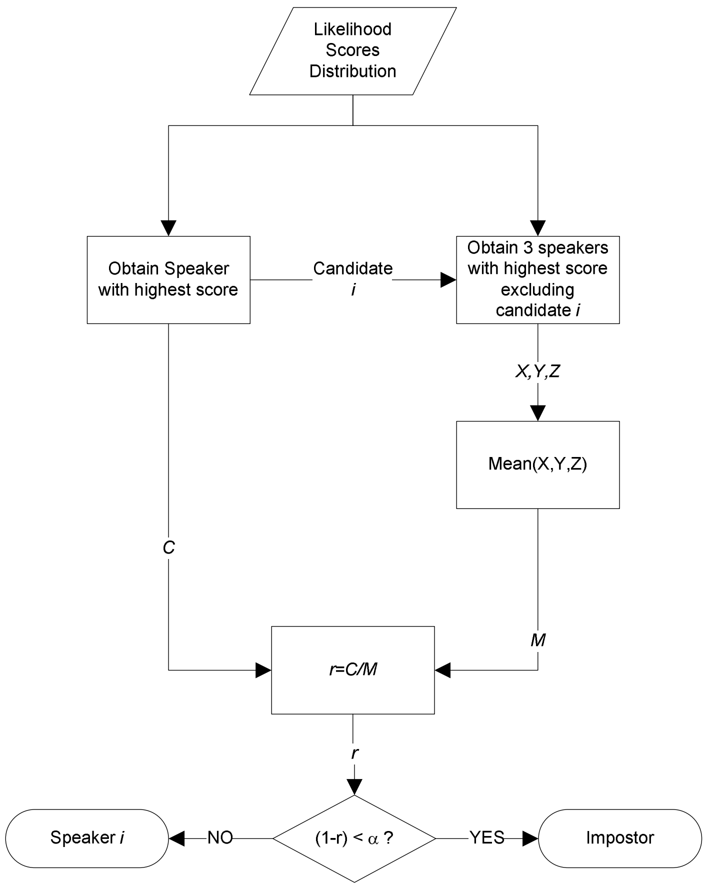

Likelihood Scores Distribution
Obtain Speaker with highest score
Candidate *i*
Obtain 3 speakers with highest score excluding candidate *i*
X,Y,Z
Mean(X,Y,Z)
C
M
r=C/M
r
Speaker *i*
NO
(1-r) < α ?
YES
Impostor

Fig 7.2 Decision algorithm when no background model is available

In order to use a convention for comparing the performance of background models against the algorithm presented in this section, a threshold α constrained to

$$0 \leq \alpha \leq 1 \tag{7.2}$$

is introduced. Value 0 of α gives the highest impostor acceptance whereas value 1 gives the toughest access permission. In order to follow this convention, (7.1) has to be re-ordered. The final expression to decide whether the candidate is an impostor or a known speaker is given by

$$dec = \begin{cases} \text{Speaker } i & \text{if } 1 - r \geq \alpha \\ \text{Impostor} & \text{otherwise} \end{cases} \tag{7.3}$$

In this section an algorithm for fast decision when no background model is available has been introduced. Its performance will be tested in next chapter.

7.1.2 Method based on Shannon Entropy

Shannon Entropy [30] is a measure of the uncertainty associated with a random variable. If the variable under scope is the likelihood score distribution for each utterance, this measure can be used to measure the degree of concentration of the likelihood scores. Shannon Entropy is obtained by calculating the following expression

$$H(l) = -\sum_k \left(\frac{l^2[k]}{\|l\|^2} \log\left(\frac{l^2[k]}{\|l\|^2} \right) \right) \tag{7.4}$$

where $l[k]$ is a sequence containing the likelihood score or each speaker *k*.

When the likelihood scores are very concentrated in a given speaker, (7.4) will be very low. On the other hand, very distributed scores which are likely to belong to an impostor, will result in a high value of (7.4). Basing on this reasoning it is feasible to set up a decision threshold based on this measure.

Both methods will be tested and compared in the following chapter of the thesis. Next section will describe how to obtain a valid background model and an algorithm to perform the decision operation in that case.

7.2 Background model

The concept of background model is based on sets theory. A background model attempts to model the space of all the speakers that can try to test the ASR. This background model can be considered as the universe in Fig. 7.3. Speakers who have been enrolled in the ASR can be found inside this universe as well as unauthorised impostors. If the speaker models are accurate enough, they can be extracted from this universal background model (UBM) so the only speakers remaining in that model are impostors.

To obtain this UBM, a very large amount of training data is needed. It should contain phonetically diverse speech produced by several speakers. The speakers should be carefully chosen so their voices represent large groups of speakers according to their dialect, intonation, tone, timber, and etcetera. This process is very complex since it requires a long study in the matter.

In this thesis a less accurate approach has been taken since such a large voice database is not available. The UBM will be obtained from all the available voice in TIMIT database [12], which is not used for training HMMs and neither for testing them and is phonetically different from that speech used in the testing experiments. In order to exclude each authorised speaker model from the UBM, the UBM will be trained in a "fuzzy way". This means that to obtain the UBM, less gaussian mixtures will be used than for the speakers' models. In addition, the number of iterations in training will be reduced comparing to the speakers' models. In this way, each authorised speaker model will arise accurately from the UBM.

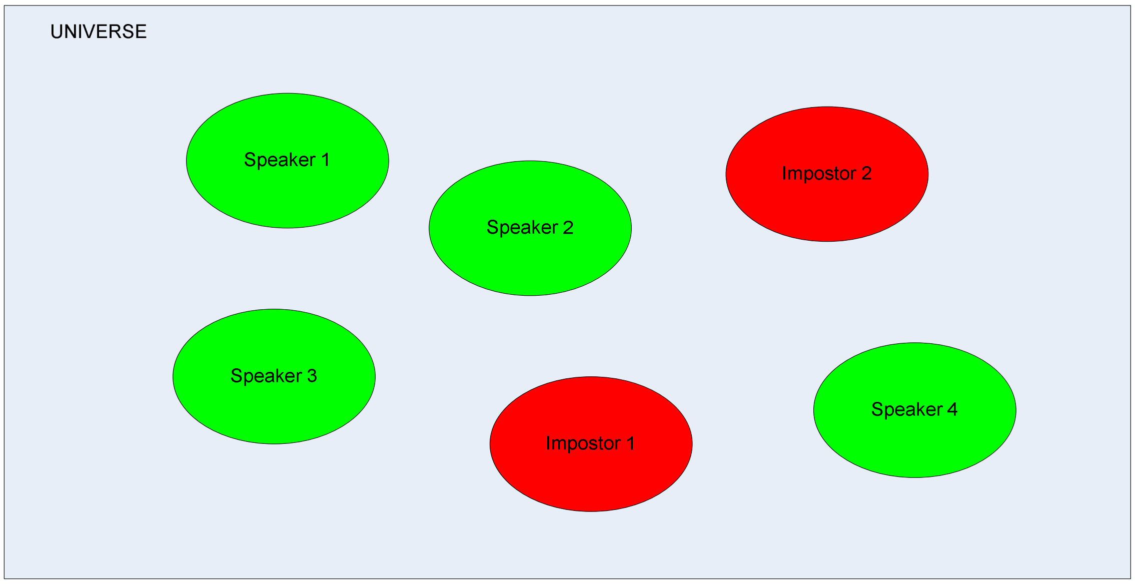
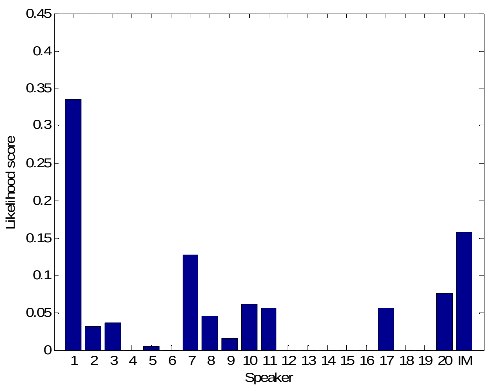


Fig 7.3 Background model role in speaker identification

When using a background model, setting up a threshold is easier since a direct comparison between the likelihood score for the impostor model and the candidate with the highest likelihood score can be made. In addition, scores normalisation can also be applied [26]. However, the same principle as in the case with no UBM will be considered. Training and testing will be performed in the same conditions. In Fig. 7.4 the likelihood scores distribution for an utterance produced by "Speaker 1" (on the left) and an impostor (on the right) are presented.

A direct comparison of the UBM likelihood score against the rest of the speakers' scores is discriminative enough. However, in Fig. 7.4, it can be seen how the effect of scores spread commented in previous section is also present. In the case of an impostor, the winner is usually the UBM but sometimes this does not happen. However, the likelihood scores will be highly distributed along all the speakers so the algorithms described in previous section can be used to avoid mistakes committed by the UBM.



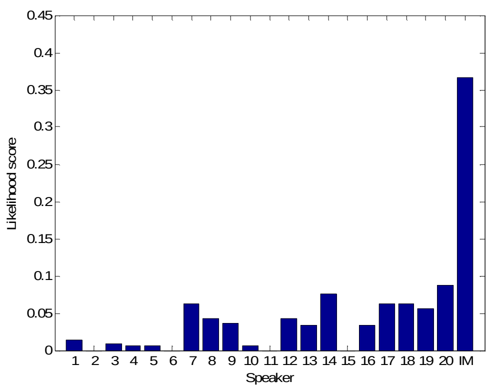


Fig 7.4 On the left, likelihood score for an utterance produced by "Speaker 1". On the right, likelihood score for an utterance produced by an impostor. Impostor model is expressed as "IM"

Use of UBMs will be tested and compared against the case where they are not considered. All this will be done in the following chapter.

8 EXPERIMENTS, RESULTS AND DISCUSSION

In this chapter all the techniques presented in previous chapters will be tested, optimised and discussed. Only the general theory of these techniques has been exposed without giving any further explanation on the performance, achievements or limitations that they offer. In this chapter this will be done following an increasing complexity order.

Firstly, the system will be considered as *closed-set*, this means that only speakers who have an attached model will be used to test the system. This is not the final goal of this thesis. However, this method can be used to obtain the system parameters that maximise the global recognition rate such as the dimensionality of the feature vectors, the number of states of each HMM, etcetera. Once these parameters are obtained, they will remain constant in the second stage of the experiments.

Secondly, once all the basic parameters are obtained for each ASR proposed, each system will be tested in the *open-set* domain. The different decision methods presented in chapter 7 will be tested and some conclusions will be obtained. In this part, the global recognition rate is not as important as the *detection error trade-off* (DET) curve. The DET curve compares the *false alarm probability* against the *miss probability* by varying the decision threshold. A very high threshold will result in a very safe system with a very high rejection rate. On the other hand, a low threshold means a very high acceptance rate but also a very high impostor acceptance rate.

The final threshold will have to be adjusted according to the context the system will work in. The power of the DET curve is that it represents how well a given ASR works in all the contexts (strict or soft threshold). A hypothetical DET curve is presented in fig 8.1. The line in red is closer to the axis and, thus, represents a more accurate system than the line in blue. Indeed, the miss and false alarm probabilities will be higher for the system in blue than for the one in red for any value of the threshold.

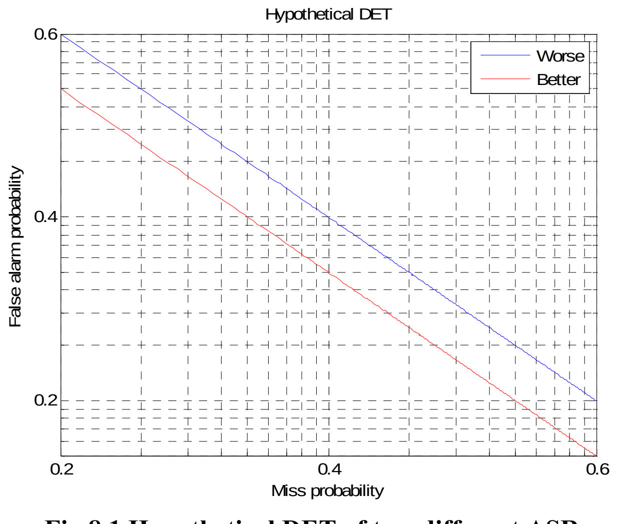


Fig 8.1 Hypothetical DET of two different ASRs

Both Polish and English voice corpuses are used for training and testing. The Polish CORPORA'97 [13] is a set of 365 utterances each spoken by females, males and kids (45 in total). It has been recorded in an office with the working computer in the background at a sampling rate of $F_s = 16$ kHz and 16 bits resolution. Only male utterances are used for both training and testing since they are acoustically closer. The total number of male speakers is 28. This limited number of speakers will reduce the testing possibilities. This is why another voice database is required.

TIMIT [12] contains a total of 6300 sentences, 10 sentences spoken by each of 630 speakers (438 males and 192 females) from 8 major dialect regions of the United States. There are 10 speech files for each speaker. Two of the files have the same linguistic content for all speakers, whereas the remaining 8 files are phonetically diverse. The corpus has been recorded in a soundproof environment with a high-quality microphone. Speech is also sampled at 16 kHz and a quantization resolution of 16 bits per sample.

The use of both an English and Polish database is done to test the language-independent property of the two of the systems here proposed.

UBMs will be obtained by using a subgroup of the TIMIT database. A total of 326 speakers and 30 sec of speech per user will be used. The speakers belong indistinctly to the 8 major dialect regions of the United States. This means around 2h30min of speech. Since no Polish speech is available for training the UBMs, the success of UBMs with Polish speech can not be assured.

The only parameters that will remain constant in all the experiments are the frame rate (30 ms), overlapping (33.3%), pre-emphasis coefficient ($\alpha = 0.95$), and cepstral liftering value (26).

In the following sections all the experiments carried out in this thesis will be described. The purpose, results and discussion and conclusions about the results obtained will be also provided.

8.1 Closed-set experiments

As it was said before, in this section experiments will be carried out in the closed-set context. This is done to obtain the best basic parameters for speaker models. A subset of 20 speakers of the Polish corpus will be used for this matter and 30 sec. of training speech per speaker. 10 utterances per speaker in the mentioned subset will be used for testing. The average duration of each testing utterance is 2.5 sec.

It is clear that since closed-set systems are constrained to be tested only by speakers with an associated model, no decision needs to be implemented. The decision given for an utterance will be the ID of the speaker with the highest likelihood score.

The parameters that have to be optimised are shown in Table 8.1. This optimisation will be done by testing the system performance robustness against noise. Noise robustness is not the main goal of this thesis so the results may be very poor in highly noisy environment. However, some algorithms such as CMN can reduce the negative effect of noise in the recognition results.

Level	Parameter
Feature Vectors	*Number of MFCCs*
	Number of filter banks
	Use of CMN
	Use of delta and acceleration coeffs.
Pattern Matching	*Number of states*
	Number of mixture components per state
	Use statistics for assigning number of states

Table 8.1 List of parameters to be optimised

8.1.1 Number of MFCCs vs. number of filter banks

Firstly, the number of feature vector components has to be found. This number is intrinsically linked to the number of filter banks used for processing the MFCCs. Indeed, the number of filter banks has to be at least the same as MFCCs being considered.

To obtain the best possible binomial, this experiment is carried out. A subset of 20 speakers from the Polish CORPORA with 30 s of speech per speaker will be used for training a set of 20 speaker models. Training is done in a noise free environment. However, in the testing stage white gaussian noise will be added so the final SNR is set to 30 dB in order to reduce the system accuracy for a better parameters comparison.

This experiment will be performed on all the systems. 3-state HMMs with 10 mixtures per state are used in all the experiments. Obviously, in voiced/unvoiced classification, voice is divided into two classes. When *k*-means algorithm is used for classifying voice, *k*=2 classes are used for better comparison against the voiced/unvoiced decision system. In the case of the speech recogniser, it is unavoidable to use less than 6 classes in order to obtain a reliable classification.

Feature vectors will be obtained without applying CMN and neither calculating delta or acceleration coefficients. The experiment will be done in two stages. Firstly, a number of MFCCs will remain constant and the number of filter banks will be iteratively increased. Thus, the number of discarded coefficients will be increased. This will be done for all the systems. The results can be seen in Fig. 8.2. Secondly, the opposite experiment will be done. This is, the number of filter banks will be fixed and but the number of MFCCs used for training and testing will be variable.

The experiment results are expected to be better as the number of coefficients grow. However, once a limit is reached, no improve is expected and even a drop-off in performance might be expected since the dimensionality of the MFCCs can grow to a point that the pattern matching algorithm is unable to deal with it.

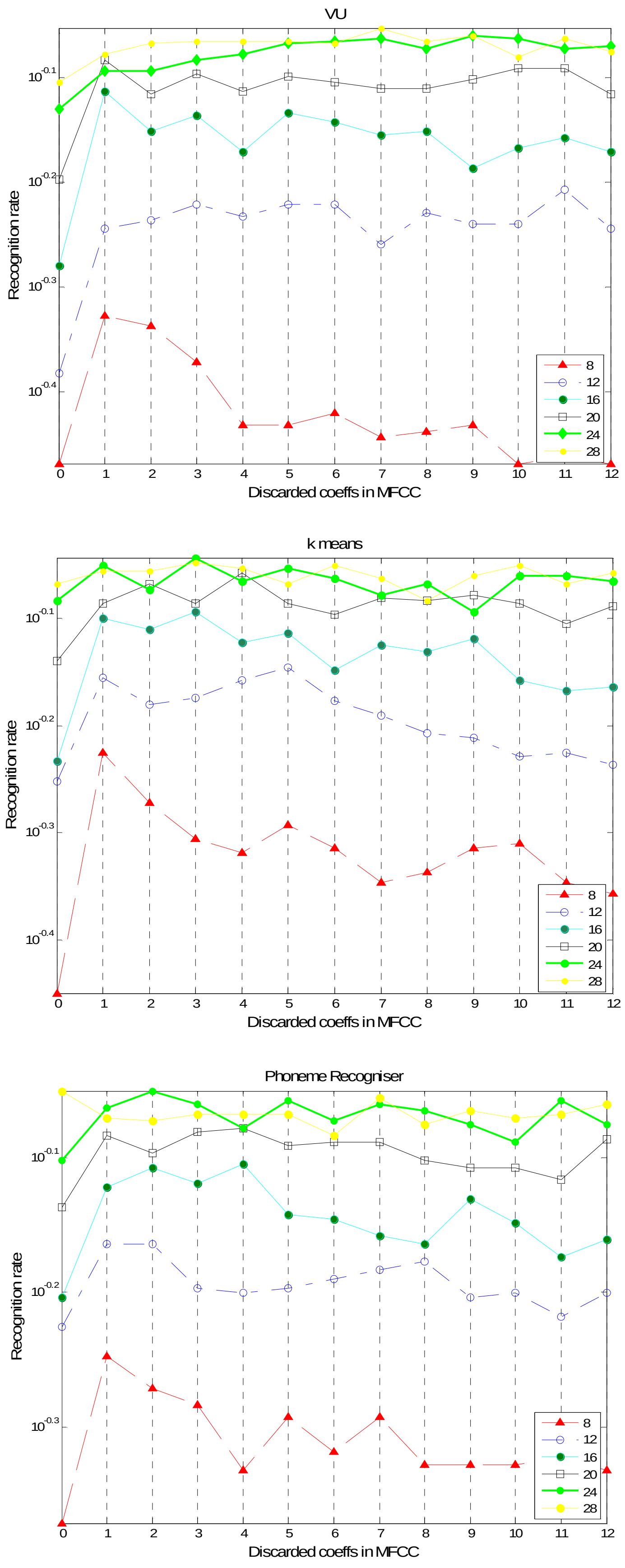


Fig 8.2 Recognition rate curves for voiced/unvoiced, *k*-means and phoneme-recogniser-based systems. Each line represents a fixed number of MFCC coefficients

The number of filter banks determines the frequency resolution of the MFCC analysis. In this section, this number is not going to be directly measured. Instead, the number of filter banks whose output is discarded for the MFCC computation is considered. In this sense the larger number of coefficients discarded, the higher spectral resolution. As it was stated in section 5.1.3, the higher order filter banks provide information about the pitch signal. This information is tricky since it offers good speaker discrimination but, on the other hand, it is relatively easy to disguise by an impostor who knows the authorised user's voice. Thus, it is desirable to discard high-order coefficients.

In Fig. 8.2, the results for each classification system are plotted. Each line represents the results for a fixed number of MFCCs which are obtained. Te abscises axis represents the number of filter bank outputs or coefficients that were discarded to obtain the fixed number of MFCCs. At first sight, the prediction that the number of coefficients is directly related to the system performance is confirmed. In fact, a very low number of coefficients, that could be sufficient for speech recognition, proves to be insufficient in the speaker recognition field. In addition, for a number of coefficients superior to 20 the results start to stabilise. However, this is clearer in Fig. 8.3.

It is clear that the effect of not discarding any of the filter bank outputs is negative in all the classification systems. When the number of coefficients is very low, discarding a coefficient means to discard a very high portion of the voice spectrum. This explains why the curve associated to 8 MFCCs is so fast-decaying, as the number of filter bank discarded outputs grows. When the number of coefficients grows, discarding a coefficient is not so critical and it is even positive because of the pitch information removal commented above.

In Fig. 8.3 the opposite experiment is done. The number of discarded filter bank outputs remains constant meanwhile the number of coefficients is variable. The experiment is repeated for several numbers of discarded filter bank outputs, each of them represented by a line. It is noticeable how in all the systems, if none of the filter bank outputs are discarded, the recognition is very poor. In addition, discarding a short number of outputs has good results if the number of coefficients is small.

For a very large number of MFCCs, the number of filter banks used is not critical but it is still positive for the final recognition results. In addition, as the number of coefficients grows, there is a clear increase in the system performance. In Fig. 8.3 it is easier to realise that for 22 MFCCs the results start to stabilise and for 24 they are already almost constant.

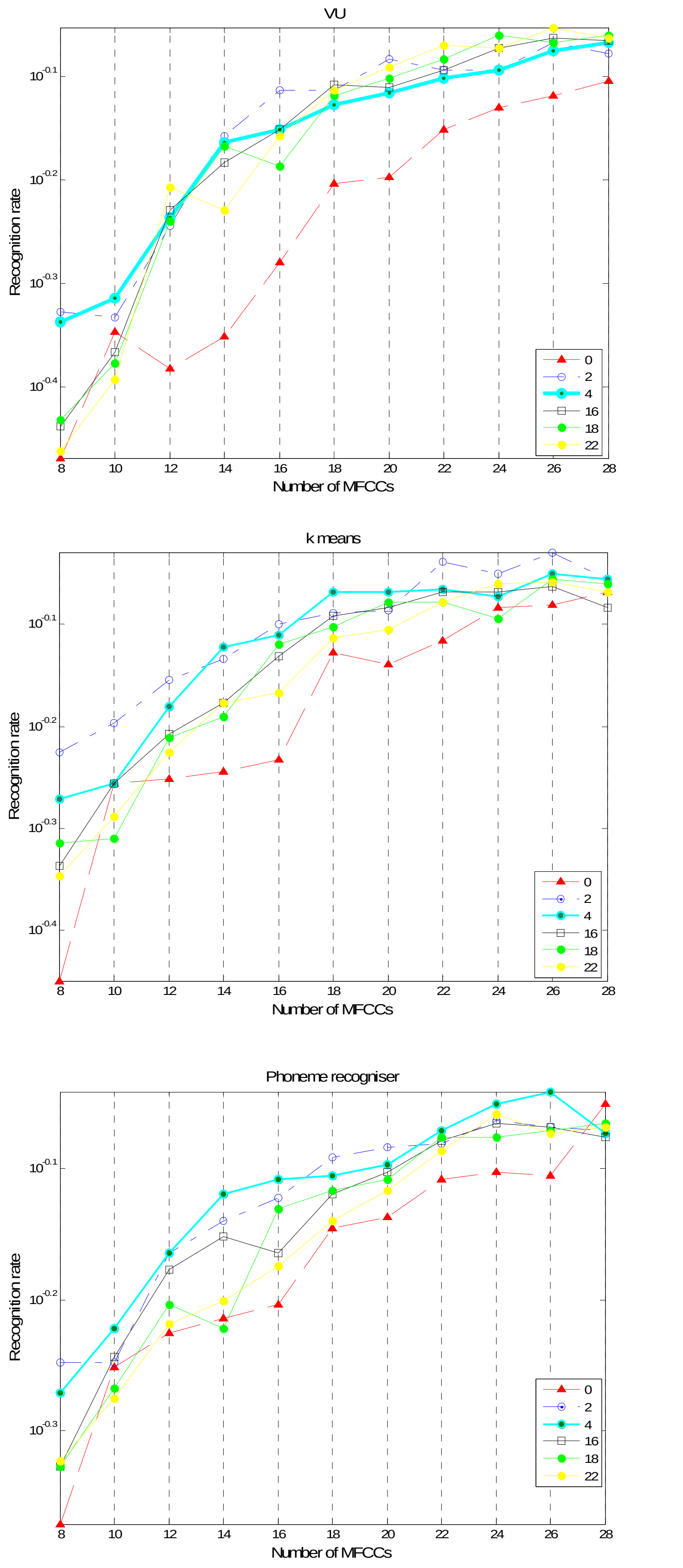


Fig 8.3 Recognition rate curves for voiced/unvoiced, *k*-means and phoneme-recogniser-based systems. Each line represents a fixed number of discarded filter bank outputs.

Another conclusion can be extracted from this graphic. If no filter bank outputs are discarded the performance of the voiced/unvoiced-decision-based system is lower than for the two other systems proposed. This might be due to the pitch information present in the voiced segments that will distort the recognition.

The target of these experiments is to find the best number of MFCCs and filter banks to maximise the recognition rate. These values should be the same for all the systems so a direct comparison of their performance can be done.

As a result of these experiments, 24 MFCCs obtained by analysing and 28 filter banks will be used in following experiments. These values are taken since they offer enough discrimination power. In addition, the recognition rate gain will not be worth increasing these values.

8.1.2 Use of CMN, and delta and acceleration coefficients

MFCCs had been chosen as the basic feature vectors. However, it was highlighted in chapter 5 how these coefficients can be more robust against noise or channel disturbances by using CMN. In addition, delta and acceleration coefficients can be appended to the basic MFCCs in order to increase dynamic information about the frame and, thus, increasing the feature vectors dimensionality.

The experiment layout will be similar to the previous one. In this case, it is interesting to obtain each method's response in noisy and clean environments. Thus, all methods will be checked under the effect of additive white gaussian noise.

The number of coefficients and filter banks will be set to the ones obtained in the previous section. The other configuration parameters will not be different from the ones used in the previous section. Once more, all the experiments will be done over the three systems proposed.

The results of the experiment for the three systems proposed are represented in Fig. 8.4. At first sight, CMN does not work as well as it was expected. Only between a SNR of 10 and 30 dB it outperforms other methods where no CMN is performed. It is clear that applying CMN in a clean environment is negative since there is not noise to remove and removing the cepstral mean will mean removing some information from the MFCCs. However, the performance increase when it is applied in a noisy environment is almost insignificant. The reason might reside in that the real application of CMN is to compensate the differences between channels in the training and the testing stage [11]. A higher performance increase would be noticed if the distortion introduced by the channel used for testing were something more than additive white gaussian noise such as the frequency distortion introduced by telephonic channels.

In addition, delta and acceleration coefficients seem to enhance the recognition. It was predictable since they add more speaker-dependent information to the basic MFCCs. MFCCs with delta and acceleration coefficients appended achieve the best results for the voiced/unvoiced and *k*-means-based systems. The system based on the phoneme recogniser obtains better results when only delta parameters are used.

However, the difference is very small so delta and acceleration coefficients will be used henceforth.

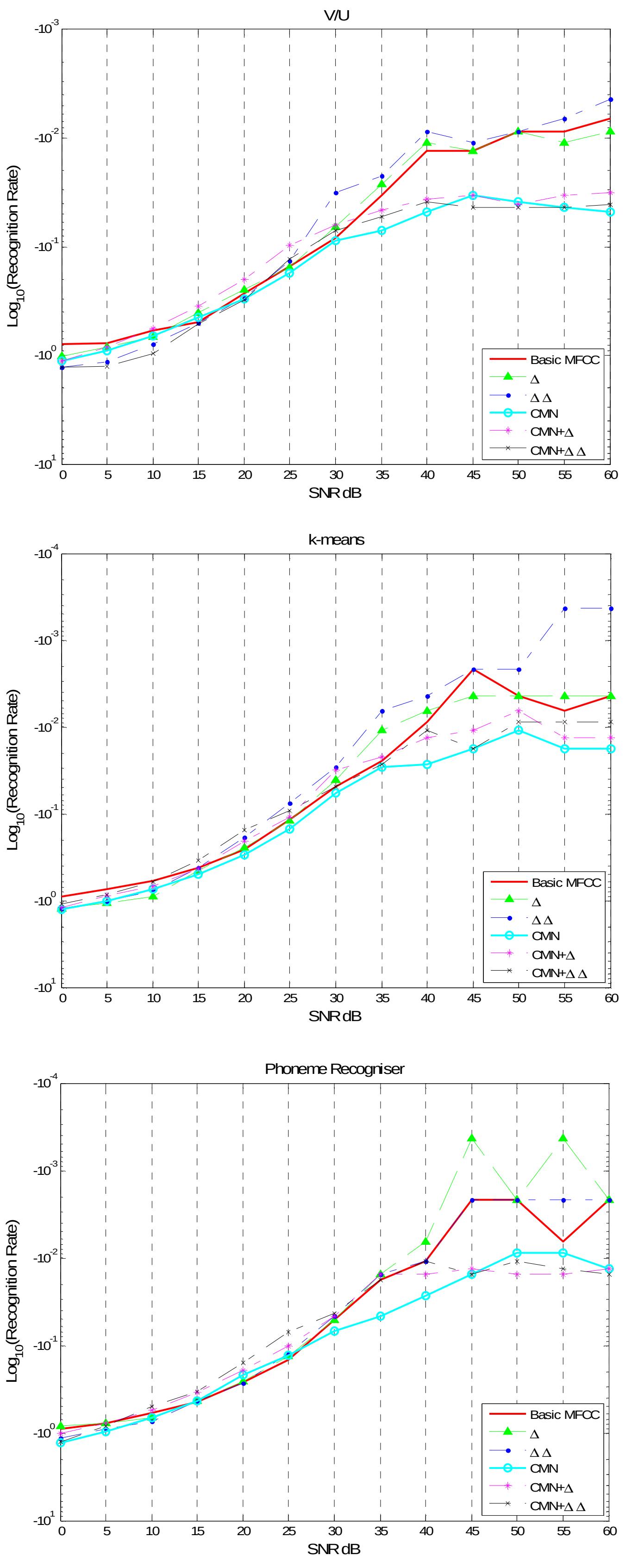


Fig 8.4 Performance of MFCC enhancing techniques against additive gaussian noise

8.1.3 Number of states

In this section, the optimum number of states will be researched. The number of gaussian mixtures per state will remain equal to 10 and the rest of parameters such as length of the feature vectors, number of classes for the *k*-means algorithm will be the same as in the previous experiments. In addition, MFCCs will be upgraded with delta and acceleration coefficients.

Once more, the system will be tested under different noise power conditions. The experiment will be repeated for different values of the number of HMM states. In the case of the voiced/unvoiced system, an extra method for selecting the number of states in each speaker model is provided.

This method obtains the mean duration of voiced and unvoiced sounds for each speaker by analysing the training data. Thus, the number of states for each speaker depends on this duration. The longer duration, the larger number of states. In the graphic on the top left in Fig. 8.5, it appears under the name of "variable" number of states in red colour.

The results in Fig. 8.5 show how for the voiced/unvoiced-decision-based on system, using a variable number of states improves the recognition results. Increasing the number of states is also desirable. However, since the training data is limited, using too long HMMs can be counterproductive. In addition, too long HMMs are not a good model for the short-time position variation of the articulatory organs.

Variable length HMMs were not implemented for *k*-means or speech-recogniser-based systems due to the elevated number of classes into which voice can be classified. This classification breaks voice into too short speech segments so long models can not be trained. In Fig. 8.5 k=2 classes are being used for *k*-means and 6 classes in the phoneme recogniser. For 30 s of training data, it was impossible to train a HMM with more than 5 states for the speech recogniser system. The same would happen for *k*-means.

The winner number of states is the variable one for voiced/unvoiced classification and 3 for the rest of the systems. In HTK the first and last states are no-emitting, so using three states means using mono-state models. This kind of HMMs is called *degenerated* and the only pattern matching power they show is obtained from the GMM associated to that state.

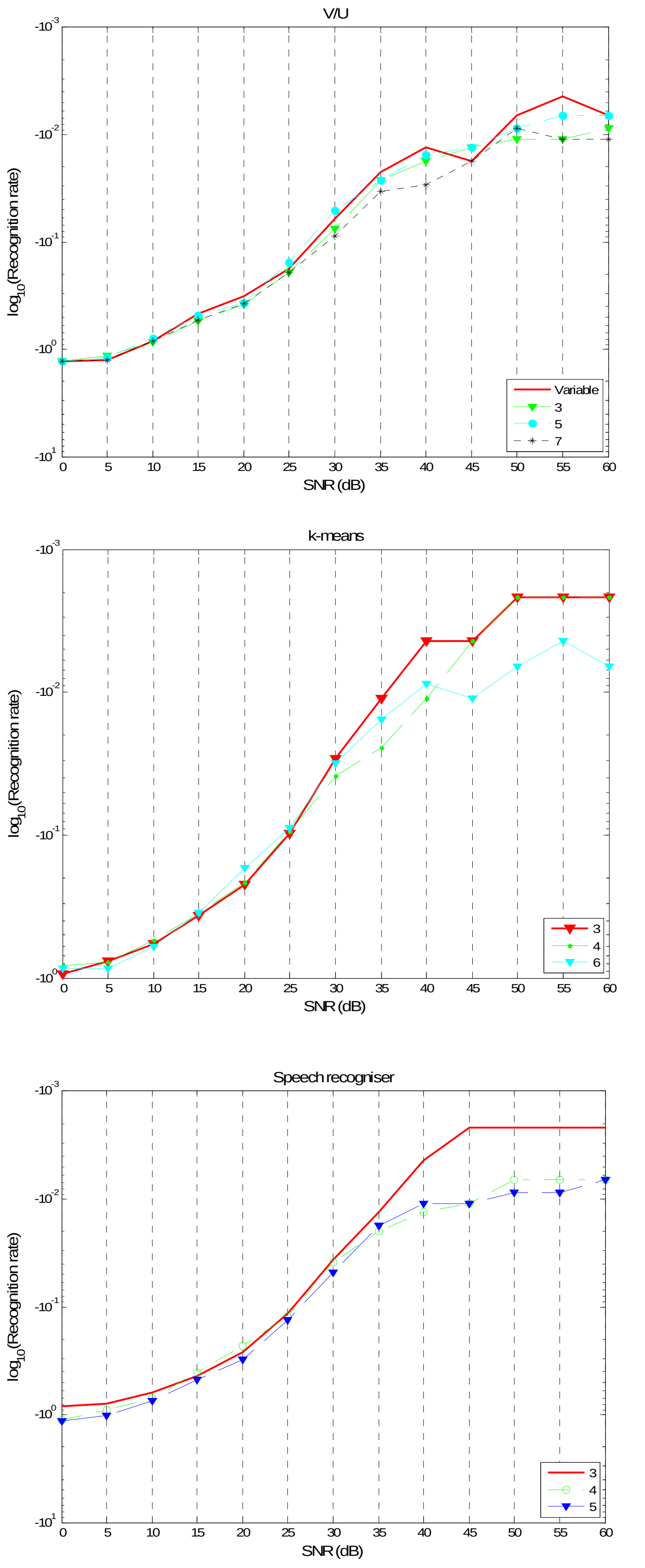


Fig 8.5 Performance of HMMs with different number of states against additive gaussian noise

8.1.4 Number of gaussian mixtures

The last parameter to be obtained is the number of gaussian mixtures per state. This number has to be large enough to provide a good modelling but short enough to avoid computation complications in both training and testing stage. The number of mixtures has a direct impact in the time needed for training a model as well as the time elapsed in testing one utterance. Thus, the shortest number of mixtures which provides acceptable results should be taken.

In this section an experiment for tuning the number of mixtures will be done. This experiment will use the same parameters as the previous ones. The number of HMM states will be set to 3 for all the systems in order to provide a good comparison. The experiment will consist in testing every system under different noise conditions and the test will be repeated for an increasing number (in powers of 2) of mixtures.

In Fig. 8.6 the results for the three systems implemented are presented. The systems are tested for 1, 2, 8, 16 and 32 mixtures. An attempt to train models with 64 mixtures was done. However, some of the models experienced problems regarding a lack of training data.

In the figure, it is clear that a number of mixtures lower than 8 is not accurate enough. At the same time, results for 16 and 32 mixtures are very similar. This is the reason why 16 mixtures per state will be chosen, since it is not extremely elevated and offers good discrimination power. Besides, if less than 30 s of training speech are available, it will still be possible to obtain accurate models. For 32 mixtures this can not be assured.

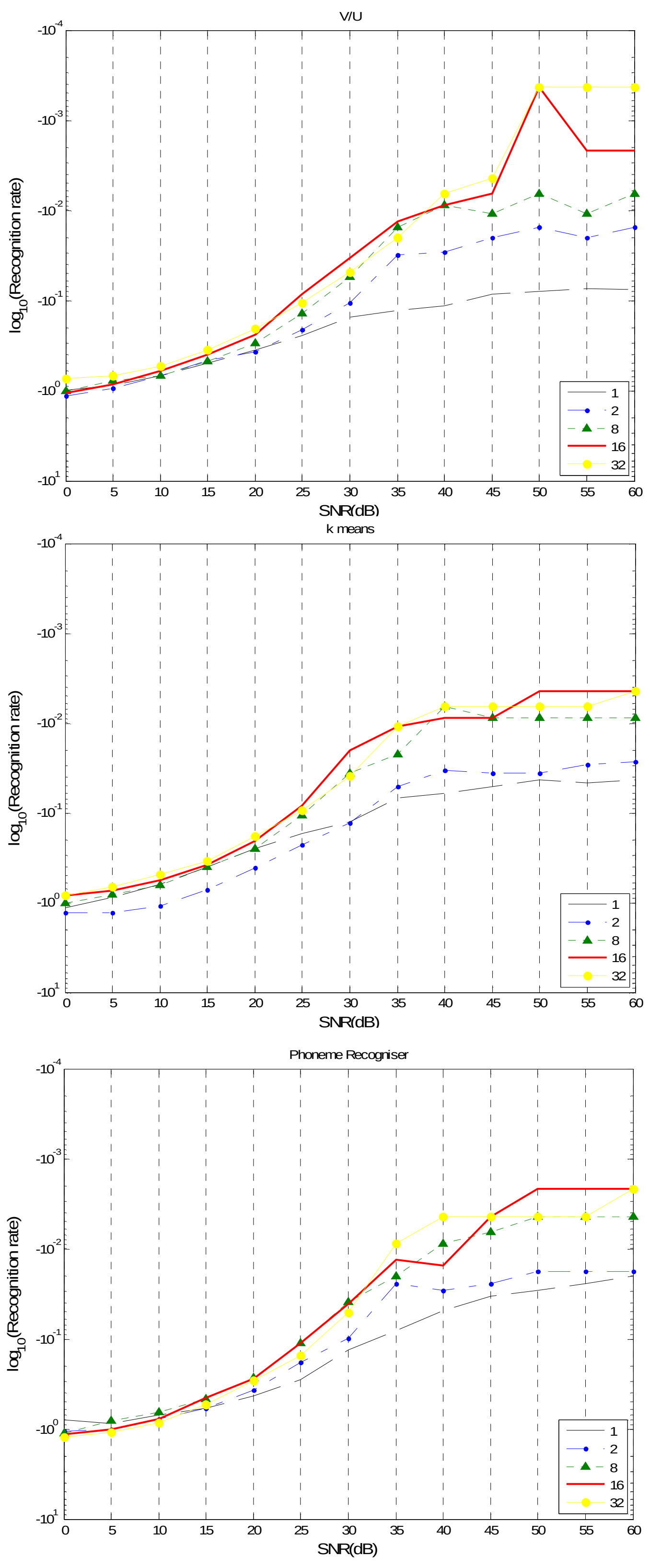


Fig 8.6 Performance of GMMs against gaussian noise for different values in the number of mixtures per state

8.1.5 Conclusions

In conclusion, the experiments carried out in the closed-set context have made possible to obtain a set of parameters which maximise the recognition rate. In this part of the experimental procedure, the three ASRs proposed have not been contrasted since the results are very similar. In addition, the focus of this thesis is to obtain a system which can work in the open-set context. It will be in this part of the experimental layout where the three systems will be extensively compared.

The parameters configuration which has been stated in this stage of the experiments to be the optimum one is summarised in Table 8.2. It will be used henceforth in the second stage of the experiments corresponding to the open-set context.

Level	Parameter	Value
Feature Vectors	*Number of MFCCs*	24
	Number of filter banks	28
	Use of CMN	No
	Use of delta and acceleration coeffs.	Both
Pattern Matching	*Number of states*	3
	Number of mixture components per state	16
	Use statistics for assigning number of states	Yes (in the case of V/U system)

Table 8.2 List of parameters obtained in the closed-set experiments

8.2 Open-set experiments

In this final stage of the experiments, the possibility than an impostor tries to enter the system will be considered. In this stage, the inclusion of a decision making algorithm like those ones presented in chapter 7 is critical. Therefore, DET curves will be plotted in order to measure the quality of each recogniser.

The experiments will be mainly based on the TIMIT database since it contains a much larger number of speakers than the Polish CORPORA. In a first stage, each system will be individually analysed in order to optimise some parameters which were not considered in the previous section such as the number of classes used for *k*-means clustering or a performance comparison between RAPT and zero crossing voiced/unvoiced decision algorithm.

Secondly, the performance of each system will be compared to the others. This final comparison will be done by varying the number of enrolled speakers, the length of the training samples and the noise conditions.

All the tests here presented are based on feature vectors and HMMs configured with the parameters described in Table 8.2. TIMIT database contains 10 sentences per speaker, so 6 of these sentences will be used for training and the 4 remaining ones will be used for testing. That sums around 15 s of training data for each speakers and 2.5 s for testing with 4 tests per speaker.

UBMs have also been obtained for every system. These models were obtained by using all the available speech of 326 male speakers in the TIMIT database. This sums up to 2 hours and a half of speech. These models were obtained by using only 3 states per HMM and 10 mixtures per state. In addition, the number of iterations for Viterbi training and Baum-Welch re-estimation was reduced in a 90 % respect from the other speakers' models. This was done in order to make each enrolled speaker model much more accurate than the UBM. Thus, an impostor will be recognised as an unauthorised user unless his voice is very close to one of the speakers' models.

The case of the system based on a speech recogniser is slightly different. It will need a larger amount of data in order to obtain convergent models. In addition, its performance for no English language speakers is expected to be very poor.

Following sections will describe all the experiments carried out in the open-set context. The final results will be shown and fully discussed.

8.2.1 Voiced/unvoiced-decision-based system

In this part of the open-set experiments some configurations will be tested in the voiced/unvoiced system. To begin with, in section 6.1.1, two algorithms were presented to perform the voiced/unvoiced decision needed in the classification subsystem. These algorithms, zero crossings and RAPT will be tested in this section. Secondly, variable length HMMs will be tested against fixed length ones in the open-set context. This comparison had already been done in the closed-set context. However, this new test will show interesting results.

In addition, two decision methods will be used. The first one, explained in section 7.1.1, uses the likelihood score distribution means to decide whether the utterance being studied belongs to an authorised speaker or an impostor. The second one was described in section 7.1.2 and is based on Shannon entropy measure.

As it was commented above, 20 speakers from TIMIT database will be enrolled in the system by using 6 spoken sentences per speaker. 2 of those sentences contain the same phonetic message, whereas the other 4 vary from one speaker to another. For testing 4 sentences which have not been used in the training stage will be used for each speaker. 20 impostors will also be chosen among the TIMIT speakers. For each of this impostors 4 sentences will be used to perform individual tests. The content of these sentences is not duplicated in any speaker.

Next two sections will describe the experiments carried out for the voiced/unvoiced based system.

8.2.1.1 Zero Crossings vs. RAPT

The first experiment is done in order to compare the performance of the fast voiced/unvoiced (V/U) decision algorithm based on the zero crossings (ZC) rate and another based on a robust algorithm for pitch tracking (RAPT). The V/U decision algorithm based on ZC is much faster but it is expected to be less accurate than the RAPT based one, mostly in noisy conditions. Thus, the results are expected to be better for the RAPT. However, the results should not be completely different for both systems since HMMs will tend to unify the results.

However, as it is shown in Fig. 8.7, the results contradict the previous assumption. ZC performs better than RAPT for every value of the SNR. RAPT makes an almost perfect classification of voiced and unvoiced sounds. However, this classification might not be the ideal one for a speaker recognition system. The reason why ZC gives better results can rely on the measure it performs in the speech signal. Very slow or very fast variation signals will be included in one class and medium variation signals in another one. This classification could be more accurate than a strict voiced/unvoiced decision.

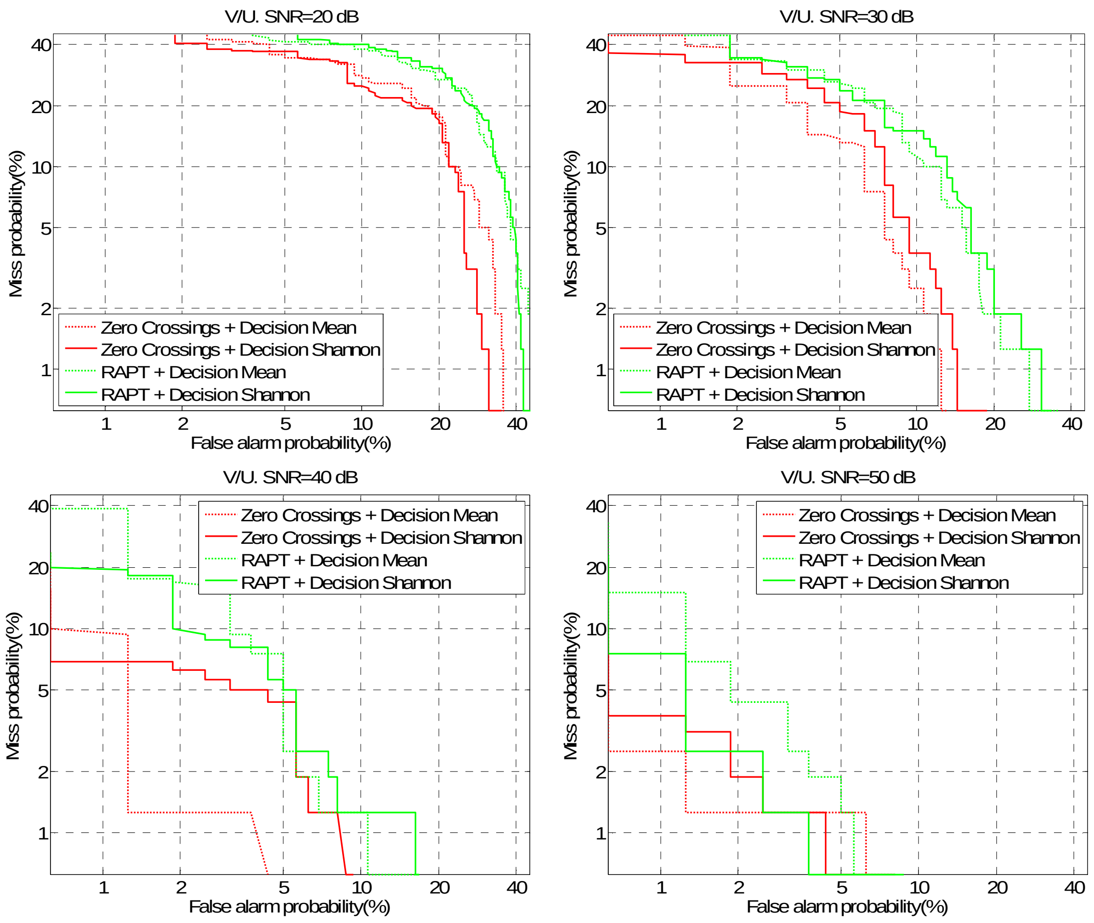


Fig 8.7 DET curves for systems based on V/U decision. Zero crossings vs. RAPT and decision based on likelihood distribution mean vs. Shannon entropy are compared

Two different decision methods have also been tested. The first one is based on a comparison of the highest value found in the likelihood scores distribution against the mean value of the 3 other highest distribution scores. This method, obtained empirically after many experiments is represented in dotted lines in Fig. 8.7. The other method is based on the Shannon entropy and is represented with a continuous line.

The results show a clear advantage for the mean-based system. However, in the following experiments, it will be shown that this is not always like that. There is not an optimum decision algorithm for all the systems. The results obtained by the decision algorithm rely on the likelihood scores distribution. Since each system here proposed gives a different shape of this distribution, decision methods will have different behaviour depending on the system under scope.

The recognition rate is calculated as the number of successful recognitions divided by the total number of utterances tests. In Table 8.3, the maximum recognition rate obtained for each experiment is shown. This maximum is obtained for a given value of the decision threshold. In Fig. 8.7 it corresponds to the closest point to the coordinate's origin in each curve.

	SNR = 20 dB	SNR = 30 dB	SNR = 40 dB	SNR = 50 dB
ZC/Mean Dec.	65.00%	**87.50%**	**97.50%**	**97.50%**
ZC/Shannon Dec.	**68.13%**	86.25%	92.50%	96.25%
RAPT/Mean Dec.	54.38%	80.00%	92.50%	94.38%
RAPT/Shannon Dec	55.00%	78.75%	91.88%	96.25%

Table 8.3 Maximum recognition rate obtained for each test

8.2.1.2 Number of HMM states

The experimental layout is the same as in the last experiment. In section 8.1.3, the number of HMMs had been tuned for the closed-set case. In this section this experiment is repeated for the open-set case.

It had been stated in section 8.1.3 that choosing a variable configuration for the number of HMM states depending on each speaker training data was the best choice. In this section, an experiment to contrast this result in the open-set case will be done. The system will be tested for a variable number of HMM states and a fixed HMM configuration of 3 states per speaker. The rest of the variables will be the same as for the previous experiment. In addition, both decision methods used also in the previous section will be adopted.

In Fig. 8.8 the DET curves for this experiment are plotted. At first sight, the system with the best performance seems to be the one with constant number of states. This difference is clearer in the case of a SNR of 30 dB. For cleaner speech, the results are not so clear since variable number of states offers lower false alarm probability but higher miss probability.

However, Table 8.4 that shows the maximum recognition rate achieved for each configuration shows that the same accuracy is obtained for both methods when the SNR rises to 50 dB. Now, the comparison between both decision methods is not so clear but decision based on measure of the mean offers slightly better performance than the Shannon entropy method.

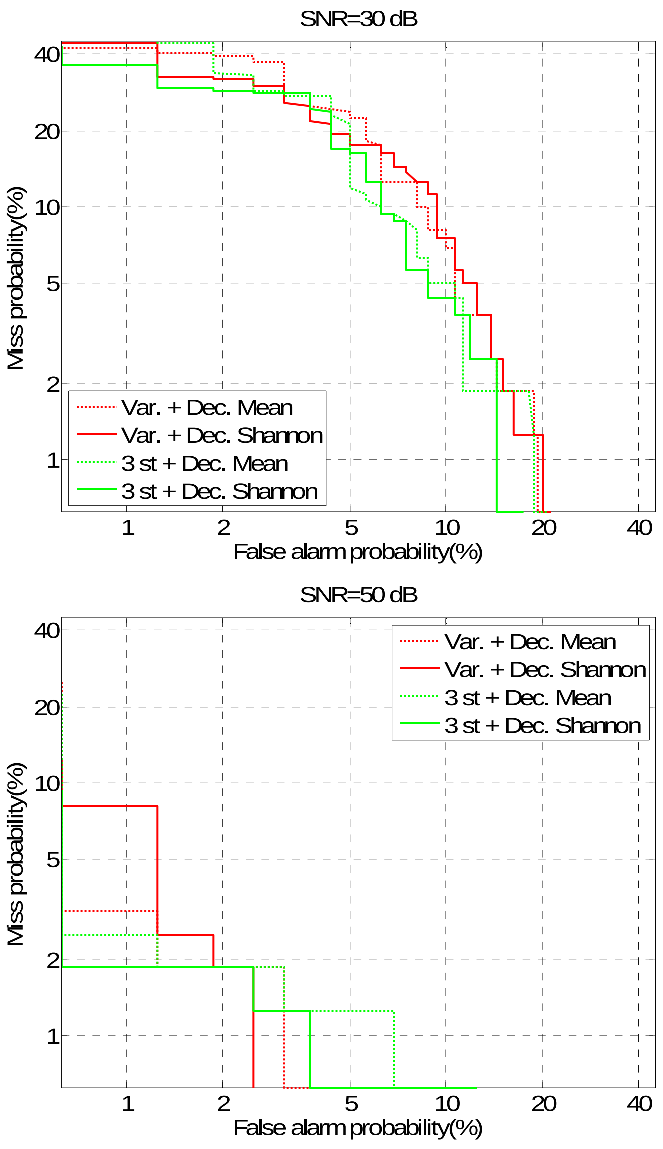


Fig 8.8 DET curves for systems based on V/U decision. Variable number of HMM states vs. 3-state fixed HMM length and decision based on likelihood distribution mean vs. Shannon entropy are compared

	SNR = 30 dB	SNR = 50 dB
Var/Mean Dec.	85.00%	96.88%
Var/Shannon Dec.	83.13%	**97.50%**
3 st/Mean Dec.	**86.25%**	96.88%
3 st/Shannon Dec.	**86.25%**	**97.50%**

Table 8.4 Maximum recognition rate obtained for each test

8.2.2 *k*-Means clustering . Number of classes

In this section the effect of increasing the *k* number of classes used for *k*-means clustering will be studied. Increasing this number should result in a performance improvement. However, a too elevated number of classes may cause an overflow effect. This means that the number of classes is so large that no accurate models can be trained since there is a lack of training data.

In this experiment, the same methodology as in the previous ones is followed. The experiment layout will be the same. However, in this case the performance of the system based on *k*-means clustering will be tested for different values of the number *k* of classes and different noise conditions. Once more, both mean and Shannon entropy based decision algorithms will be tested.

The DET curves obtained in this experiment can be seen in Fig. 8.9 for two different values of the SNR, 30 dB and 50 dB. The results for 30 dB show a clear advantage for *k*=4 means, whereas for 2 classes the results are the worst. These results match the assumption that increasing the number of classes increases the recognition accuracy. In the case of 7 *k*-means classes, the results are not so good mainly because of the saturation effect commented above. If more training data were available, the results would be much higher. Once more, decision based on measures of the mean seem to be more accurate.

For almost clean speech, the results are hard to interpret. For *k*=7 classes, the results have improve substantially. A conclusion can be extracted from this fact. Noise affects strongly to *k*-means clustering when the number of classes is elevated. The minimum number of classes 2 still has the worst results.

In table 8.5 the maximum recognition rates for the DET curves in Fig. 8.9 are listed. The results are coincident with the conclusions extracted from Fig. 8.9. For SNR=30 dB, the clear winner choice is to use 4 classes. However, when noise power is lower, using 7 classes offers the best results with a very short difference.

In Fig. 8.10, the number a bar graphic of the maximum recognition rate obtained for SNR=30 dB has been plotted. It includes results for a more values of *k*. Both decision methods based on likelihood mean and Shannon entropy are also considered. The saturation effect is very clear in this graphic. Before *k*=4 classes are reached, the results are monotonically increasing. Then, the saturation effect increased with the effect of noise for high values of *k* can be seen as the maximum recognition rate decreases.

The system based on 4 classes shows good results for all noise conditions and, in addition, guarantees a training convergence for relatively short training data sequences. Thus, it will be adopted in the last stage of the experiments where the three systems here proposed will be compared to each other.

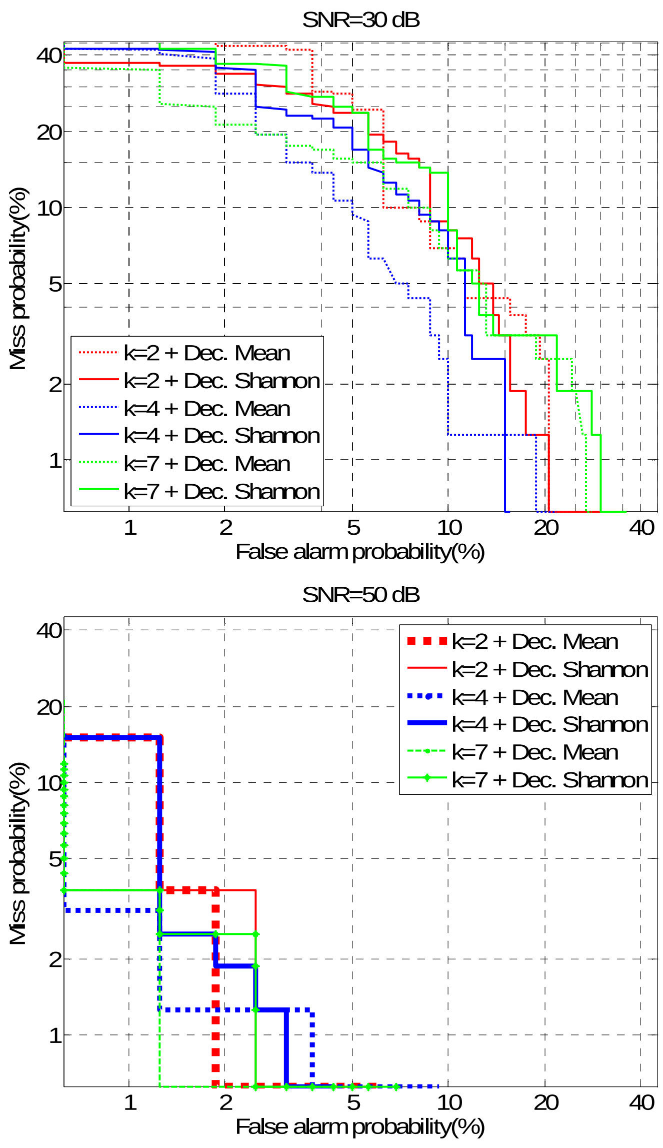


Fig 8.9 DET curves for systems based on *k*-means clustering. Different number of *k* classes and decision based on likelihood distribution mean vs. Shannon entropy are compared

	SNR = 30 dB	SNR = 50 dB
k=2 DM	84.38%	97.50%
k=2 DS	82.50%	96.88%
k=4 DM	**88.75%**	97.50%
k=4 DS	85.00%	96.25%
k=7 DM	83.75%	**98.13%**
k=7 DS	82.50%	96.88%

Table 8.5 Maximum recognition rate for DET curves in Fig. 8.9

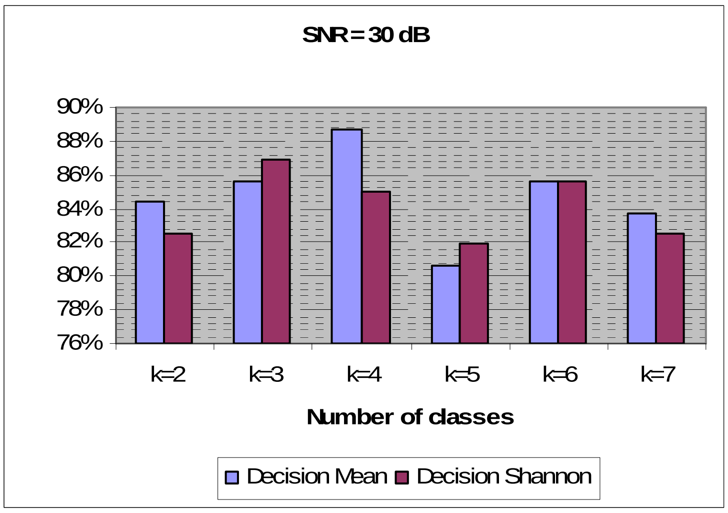


Fig 8.10 Maximum recognition rate for various values of *k*, SNR=30 dB and both decision methods

8.2.3 Phonetic classification

The last system to be tested is based on phonetic classification. This system uses a speech recogniser trained to work with Polish speech. Some attempts have done to make it work with English speech. However, the strong differences between both languages concerning the articulation procedures for each phoneme deny the possibility to train reliable models for English speakers. Thus, this system can only be tested with the Polish language available corpus. As it was commented at the beginning of this chapter, Polish CORPORA contains speech produced by only 28 male speakers. This will produce a limitation in the amount of speakers which can be used for testing.

In addition, the recording conditions are not the same for CORPORA or TIMIT. Language characteristics of Polish and English are also very different [36]. Because of these to problems, results obtained for TIMIT and CORPORA databases can not be compared in absolute terms.

In this experiment, the performance of decision methods based on measures of the likelihood scores distribution mean and Shannon decision will be tested for the speech-recogniser-based system. It will be done over a set of 14 speakers enrolled in the system whereas the rest 14 speakers from CORPORA will form the impostors set. 20 s of speech per speaker will be used for training so all the models are convergent. 10 sentences per speaker which do not belong to the training set are used for testing. Each sentence has around 2.5 s of duration. Feature vectors, HMMs and GMMs configuration parameters will be the same as in the previous experiments.

In Fig. 8.11 the DET curves obtained for SNR of 30, 40 and 50 dB are plotted. The results are not so good comparing to the ones obtained with TIMIT database. However, as it was stated above these results are not comparable. The results do not show a clear advantage for any of the two decision methods. Table 8.6 contains the maximum recognition rate obtained for each decision method and each SNR used for testing. This table shows a short advantage for the mean based decision. Thus, a conclusion can be reached after having tested both decision methods without the inclusion of a UBM for impostor modelling. The empirically obtained decision method based on a measure of the likelihood distribution mean is more accurate than the Shannon entropy method in the case that no UBMs are being used.

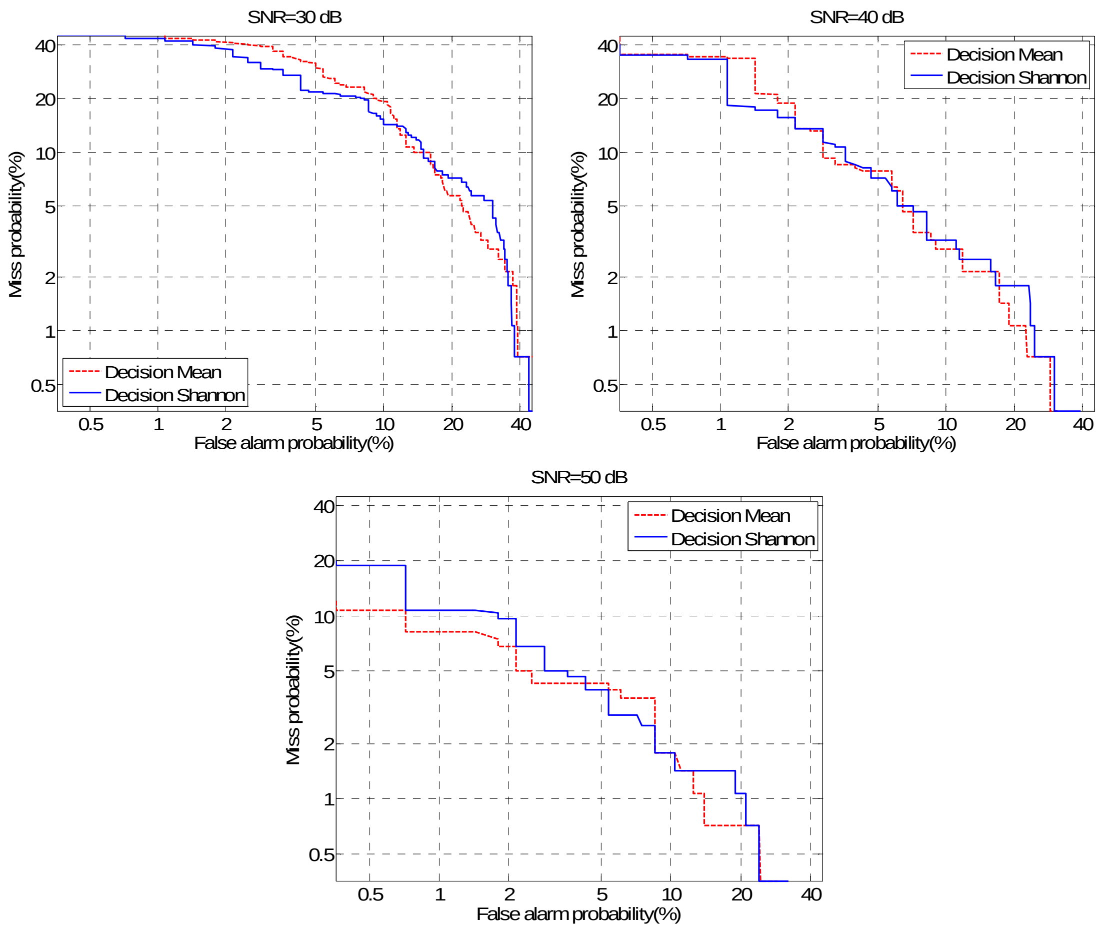


Fig 8.11 DET curves for system based on a phoneme recogniser. Likelihood distribution mean vs. Shannon entropy decision are Compared for different values of the SNR

	SNR = 30 dB	SNR = 40 dB	SNR = 50 dB
DM	**76.07%**	**89.29%**	**93.21%**
DS	75.00%	88.93%	92.14%

Table 8.6 Maximum recognition rate obtained for curves in Fig. 8.11

8.2.4 Use of UBMs

Previous sections have considered the case where no UBMs were used. In this section, the inclusion of a UBM to enhance the decision making performance will be studied.

For each system a set of impostor models has been trained as it has been described at the beginning of this chapter. The inclusion of these UBMs is supposed to strongly decrease the missing probability in each system since in all the previous graphics it can be noticed that generally the DET curves show a tendency towards higher missing than false alarm probabilities.

The experiments carried out in this part will be configured exactly the same way as the previous ones. It is expected that the miss probability and even the false probabilities decreases for all the systems. Nevertheless, UBMs have only been obtained from the TIMIT database. Thus the behaviour of the system based on the Polish language phoneme recogniser will be unpredictable. It might seem a contradiction that UBMs trained with English speech can be used with this system optimised for Polish. However, the speech data used for training the UBMs is so large that all the models for each phoneme will converge. It is expected that the Polish phoneme recogniser will classify each English phoneme as its closest Polish counterpart. This approach can not be taken for training normal speaker models since there is not enough training data for some of the phoneme models to converge.

In conclusion, next sections will test the performance of the three systems proposed when UBMs are incorporate to the set of models. A UBM will be treated one more of the speaker models by the decision algorithm. However, if the output decision corresponds to the impostor model, the utterance will be classified as having been produced by an impostor.

8.2.4.1 Voiced/unvoiced-decision-based system

The experiment will be done on variable length HMMs depending on the speaker voice statistics used in the training stage. The experiment will be carried out using the TIMIT database. The configurations will be the same as in previous experiments (except for those done on the speech-recogniser-based system).

The experiments are done for both Shannon and mean-based decision. However, decision based on Shannon always offers worse results than the other decision method. Because of this, Shannon decision will not be plotted in the graphics concerning to this experiment.

In Fig. 8.12 the results for mean-based decision are plotted for SNRs of 30, 40 and 50 dB. It is clear that the use of a UBM has increased the system performance. When the threshold is very close to the optimum one in terms of maximising the recognition rate this difference is not really appreciable. However, when the threshold is unbalanced to obtain higher security or higher acceptance rate, the results are clearly better. Thus the decision is upgraded.

In table 8.7 the maximum recognition rate obtained for both decision methods are presented. UBMs combined with mean-based decision achieve the highest results. However, the difference is very short. This was expected since this maximum is obtained for a single value of the decision threshold. However, UBMs improve the system performance for all the values of the decision threshold.

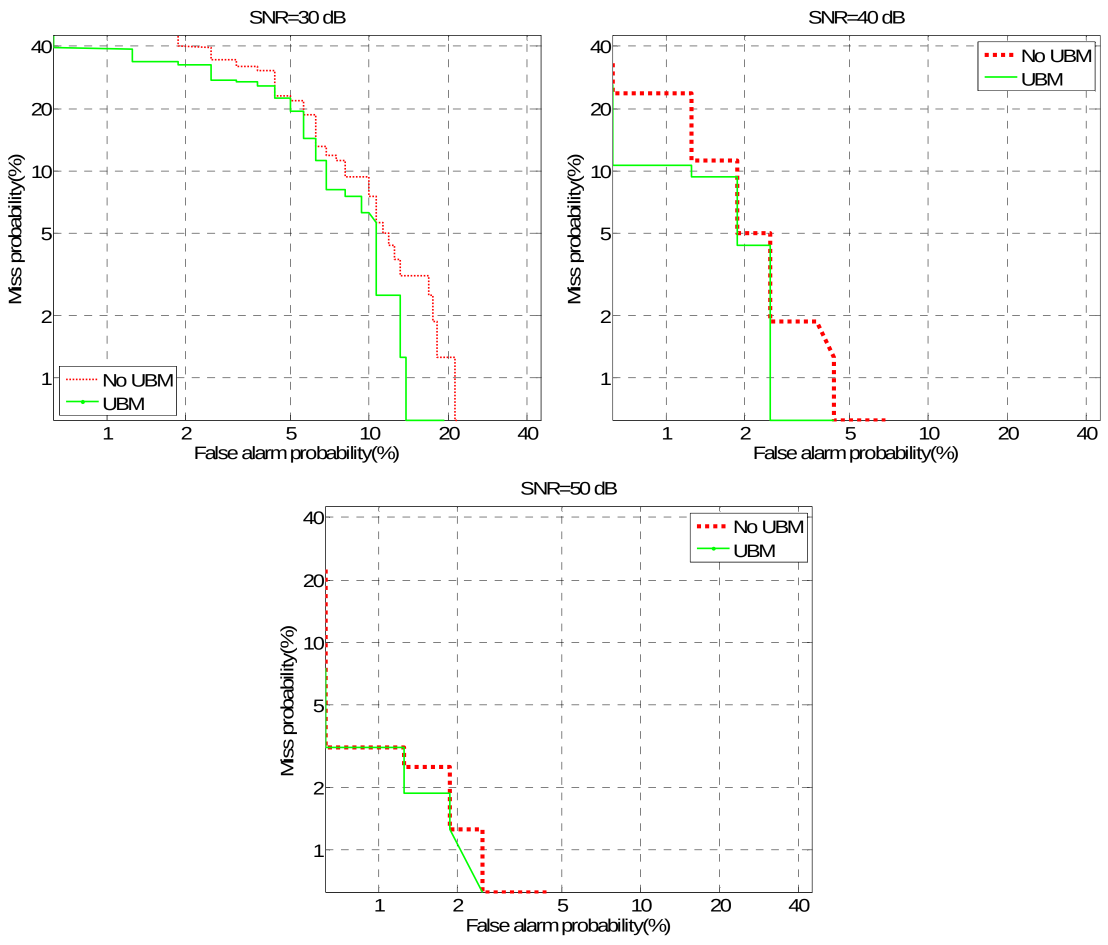


Fig 8.12 DET curves for system based on voiced/unvoiced decision. Comparison between inclusion or not of UBMs is done for different values of the SNR

	SNR = 30 dB	SNR = 40 dB	SNR = 50 dB
No UBM + DM	83.13%	95.63%	96.88%
No UBM + DS	83.13%	96.25%	**97.50%**
UBM + DM	**86.25%**	**96.88%**	**97.50%**
UBM + DS	83.13%	95.00%	96.88%

Table 8.7 Maximum recognition rate

8.2.4.2 *k*-Means-based system

Now the UBMs performance will be tested for the *k*-means clustering case. A classifier of *k*=4 classes will be used in this experiment since it was the one that achieved the highest performance as it was stated in section 8.2.2. Thus 4 UBMs (one per class) will be trained.

As usual, the experiment follows the same configuration in terms of number of speakers, length of the training and testing data and MFCCs parameters as the previous ones. 3-state-length HMMs are used. The results have been plotted only for mean-based decision. In Fig. 8.13 these results are presented for different values of the SNR in the testing stage. Now the performance has improved much more compared to the case of the V/U decision. The case of SNR=50 dB has been plotted twice since the scale used in the rest of the cases does not allow to observe the DET curve for the UBM case. On the right down corner of this figure, the graphic with a zoomed *x*-axis scale is plotted.

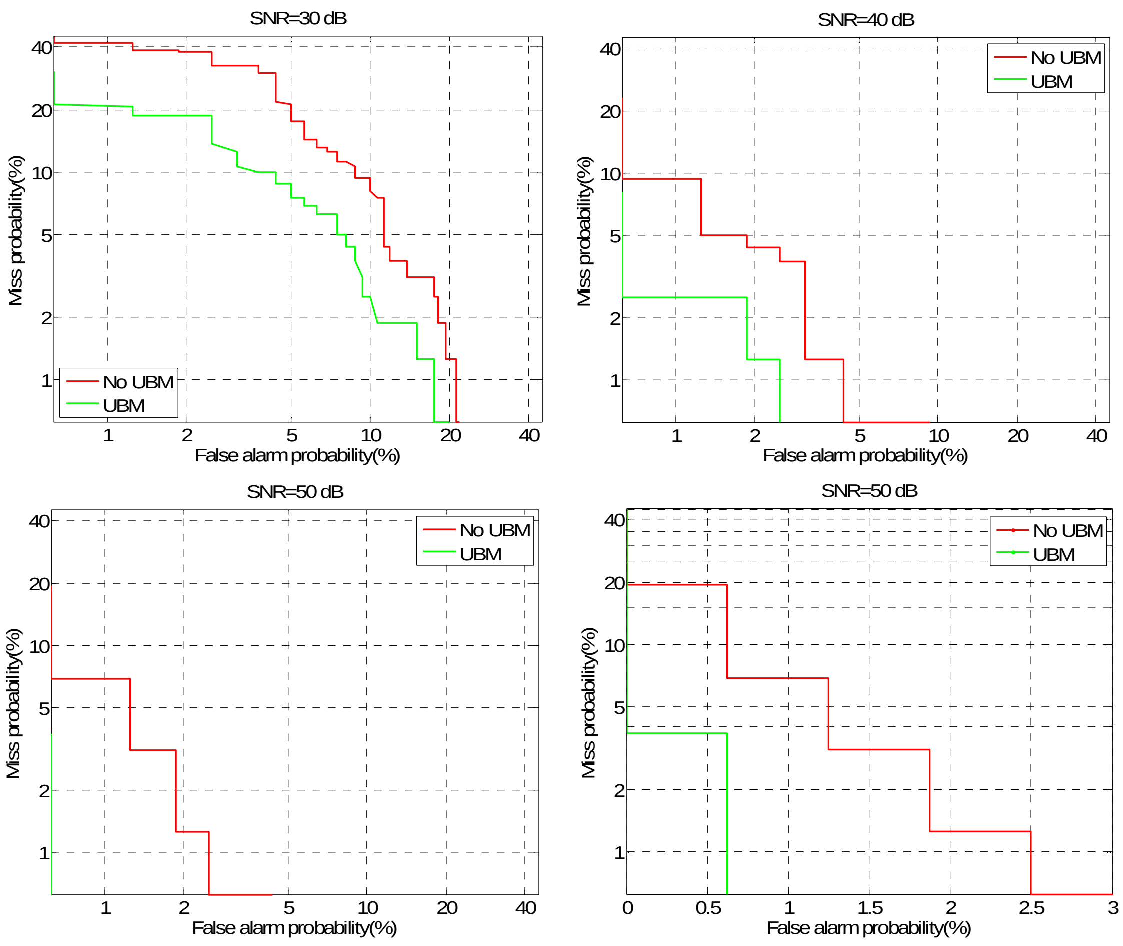


Fig 8.13 DET curves for system based on *k*-means clustering. Comparison between inclusion or not of UBMs is done for different values of the SNR

The results show an important upgrading in the decision making. Both false alarm and miss probabilities have been drastically reduced for all values of the decision threshold. In table 8.8 the maximum recognition rate obtained by iterating the value of the threshold is presented. Once more, UBMs combined with mean-based decision achieve the best results. The difference is not big. A conclusion can be reached from this fact. Including a UBM decreases both the false alarm and miss probability. However, the cost is increasing the probability of confusing one authorised speaker with another one. Nevertheless, this effect is clearly softer than the decrease in false alarm and miss probabilities.

	SNR = 30 dB	SNR = 40 dB	SNR = 50 dB
No UBM + DM	83.75%	95.63%	96.88%
No UBM + DS	84.38%	95.63%	96.25%
UBM + DM	**88.13%**	**97.50%**	**98.75%**
UBM + DS	86.25%	96.25%	**98.75%**

Table 8.8 Maximum recognition rate

8.2.4.3 Phoneme-recogniser-based system

In this experiment the performance of UBMs in the phoneme-recogniser-based system will be studied. It is expected that the results will not be as satisfactory as in the previous experiments since the UBMs are obtained from a large set of English speakers and the phoneme recogniser has been designed for Polish speech. However, the test will be performed.

A set of 14 speakers from the Polsih CORPORA will be enrolled in the system whereas the 14 remaining ones will be considered as impostors. 20 seconds of speech will be used to train each speaker models. For testing, each speaker will produce 10 different sentences of around 2.5 s per utterance. The rest of configuration parameters will be the same as in the previous experiments.

In Fig. 8.14 the results of the experiment are plotted for both decision methods (mean-based and Shannon entropy) and both cases (with and without UBMs). The hypothesis that the performance would not improve has been proved to be false. When the signal is noise-clean enough, the results for UBMs are higher. For noisy conditions, the results are very similar because of the sensitiveness of the speech recogniser to noise.

The reason is not completely clear. No one of the speakers used for building the impostor models belonged to the group of Polish native speakers. Therefore, the impostor models in this case are not a good representation for speakers belonging to this group. However, the results have improved slightly. It is believed that if more accurate impostor models trained with Polish speech would be available the results could be much higher.

In addition, now the decision method which achieves the best results is the one based on Shannon entropy measure. In Table 8.9, the maximum recognition rate for the cases where UBMs are and are not used is presented. The results in this table confirm the previous conclusions extracted from the graphics.

Once more again, it has to be said that the results can not be compared against the other systems' performance due to the use of different speech databases in both cases. In following sections, the three speaker identification systems proposed in this thesis will be compared in terms of their performance for different values of the number of speakers and length of the training data. In addition, an experiment will be carried out to obtain the degree of language independence of each system.

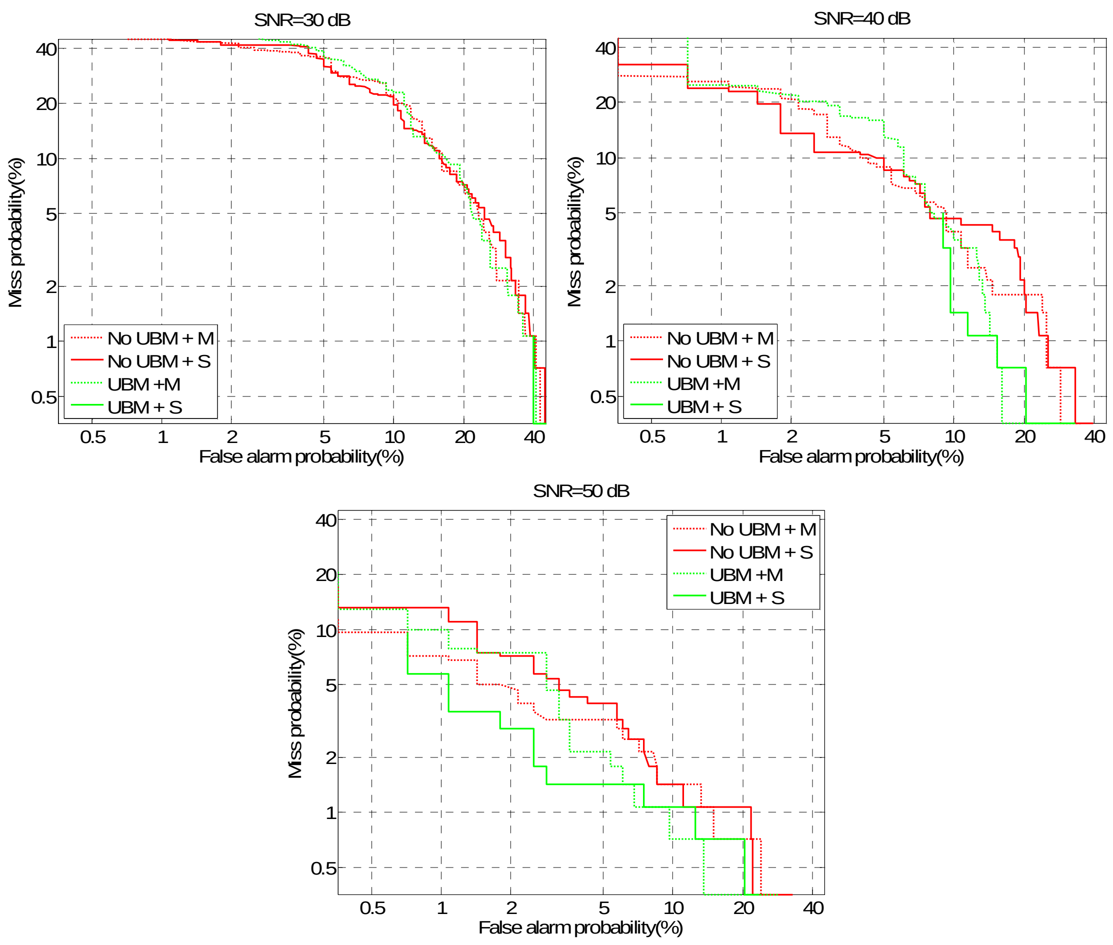


Fig 8.14 DET curves for system based on phoneme recogniser. Comparison between inclusion or not of UBMs is done for different values of the SNR. Decision method (Shannon or mean-based) is also tested.

	SNR = 30 dB	**SNR = 40 dB**	**SNR = 50 dB**
No UBM + DM	75.00%	87.50%	93.93%
No UBM + DS	**73.93%**	87.50%	92.14%
UBM + DM	73.57%	87.14%	94.29%
UBM + DS	59.64%	**88.57%**	**95.71%**

Table 8.9 Maximum recognition rate

8.3 Final systems comparison

The final stage of the experiments comprises a comparison of the three systems proposed in this thesis. This comparison will be done in three different ways and will be based on the DET curves for every system and the maximum recognition rate obtained by iterating the decision threshold value.

The first parameter which will be compared is the performance against the length of the training data. It is clear that reducing the amount of training data will produce a decrease in the performance for all the systems. However, some systems will experience a faster performance drop-off.

The second parameter used to compare these three systems is the robustness against noise. No additional noise cancelation algorithms have been used in this thesis. Therefore, when the systems performance is compared against noise a pure comparison is made. This means that this comparison will be a real measure of how naturally robust these systems are against noise before applying any kind of noise cancelling algorithm.

Finally, the third parameter under study will be the number of speakers. So far only sets of maximum 20 authorised speakers and 20 impostors have been used to test the system. However, it is believed that the models for each speaker are discriminative enough so larger populations can be stood by each system.

Furthermore all the experiments will be done on both TIMIT and CORPORA database so language-dependency can be measured with some restrictions. The fact that no convergent models can be obtained with TIMIT for the speaker-recogniser-based system constitutes a handicap since CORPORA does not contain enough speakers to perform the last experiment. Besides, the experiments where TIMIT database will only present results for the voiced/unvoiced-decision-based and the *k*-means-based systems.

In all the systems the configuration parameters for feature vectors, HMMs and GMMs will be the same. 24 MFCCs obtained from a filter bank analysis of 28 subfrequency bands with appended delta and acceleration coefficients, and 3-state HMMs with 16 mixtures per state. The number of classes in the case of *k*-means clustering will be set to 4. The decision algorithm will be the empirical method for all systems since it proved to offer the best discrimination. Finally, all the experiments will be done for the two cases of using and not using UBMs.

The following sections will describe the experiments carried out in this last stage. The results will be presented and discussed.

8.3.1 Performance vs. length of training data

The first comparison will test the behaviour of each system for different lengths of the training data. This will be done for both voice corpuses. Since the characteristics of both corpuses are quite different, the configuration of the experiments carried out will differ for both corpuses. Therefore, a language dependency comparison can only be done in relative terms.

8.3.1.1 TIMIT database

A set of 40 speakers from the TIMIT database will be enrolled in the system to perform this experiment. Other 40 speakers also from TIMIT will be used as impostors. No noise is added to the signals in the testing stage.

The length of the training data will be varied from 2 to 8 sentences per speaker where each sentence is approximately 2.5 s long. Since only 10 sentences per speaker are available in the TIMIT database, testing will be performed with the remaining sentences which are not used for training. The experiment will be done for the voiced/unvoiced based system (V/U) and the *k*-means-based one. In both cases the experiment will be repeated for both the cases where UBMs are considered or not.

In Fig. 8.15 the DET curves obtained for 3, 4, 5 and 6 training sentences are presented. DETs for 2, 7 and 8 sentences are not included since no convergent models could be obtained for *k*-means when only 2 sentences where used for training and the results for 7 and 8 sentences are similar to the ones obtained for 6.

These graphics show that the V/U system needs less training data to achieve better results. However, once there is enough training data, the DET curve corresponding to the *k*-means system offers lower false and missing probabilities. The reason why *k*-means needs more training data is due to the larger number of classes into which it clusters the training data. In addition, it is clear that systems which use UBMs always achieve better results mostly in the case of the *k*-means system.

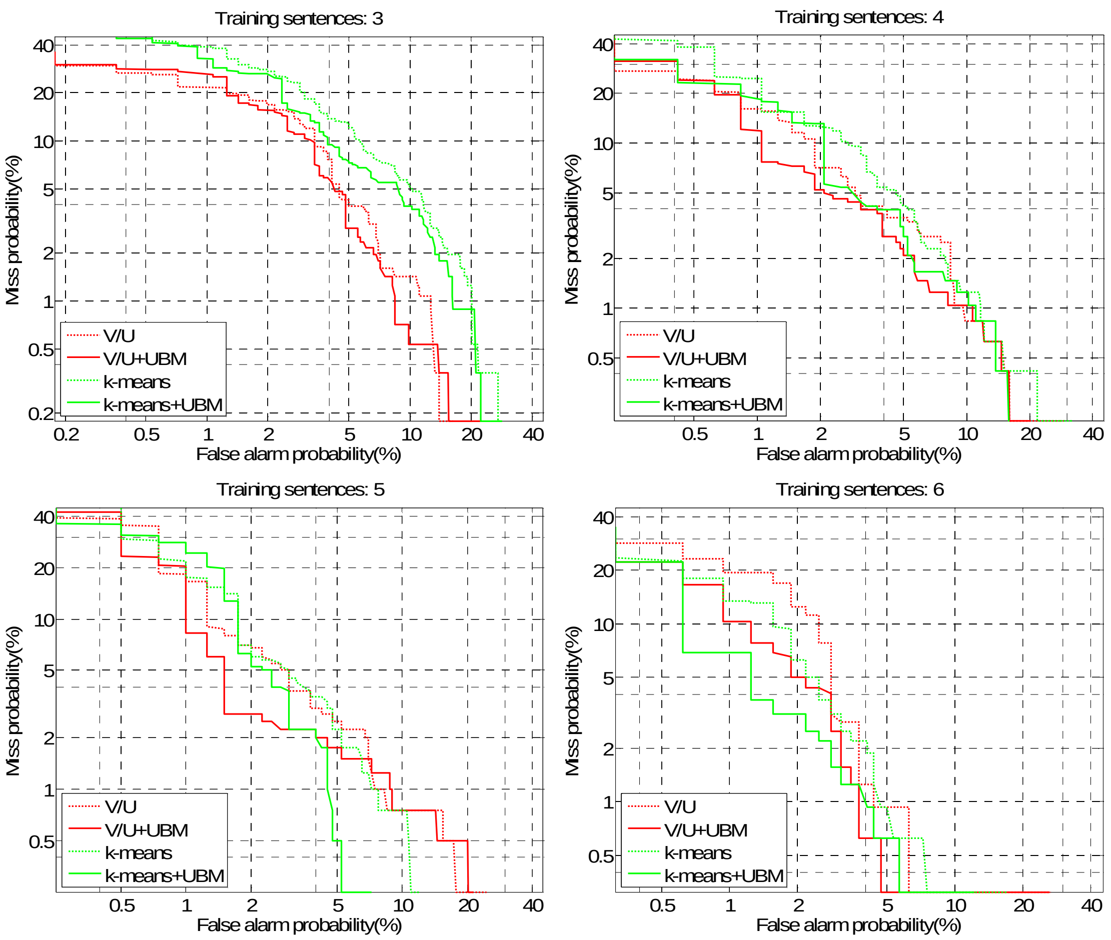


Fig 8.15 DET curves for V/U and *k*-means systems with and without UBMs for different number of training sentences

In Fig. 8.16 the maximum recognition rate for each system is represented for different number of training sentences. Both figures are the same but different y-axis scale was used in order to provide an easier visualisation. The effect of lack of training data for the *k*-means system is relevant for a short number of training sentences. In addition, despite the fact that the DET curves show better behaviour for the *k*-means systems for long training data, Fig. 8.16 shows that the maximum recognition rate is always slightly superior for the V/U system mostly if UBMs are being used. This effect might look contradictory. However, the short number of tests which are done when long testing data is used can bias the results in this sense.

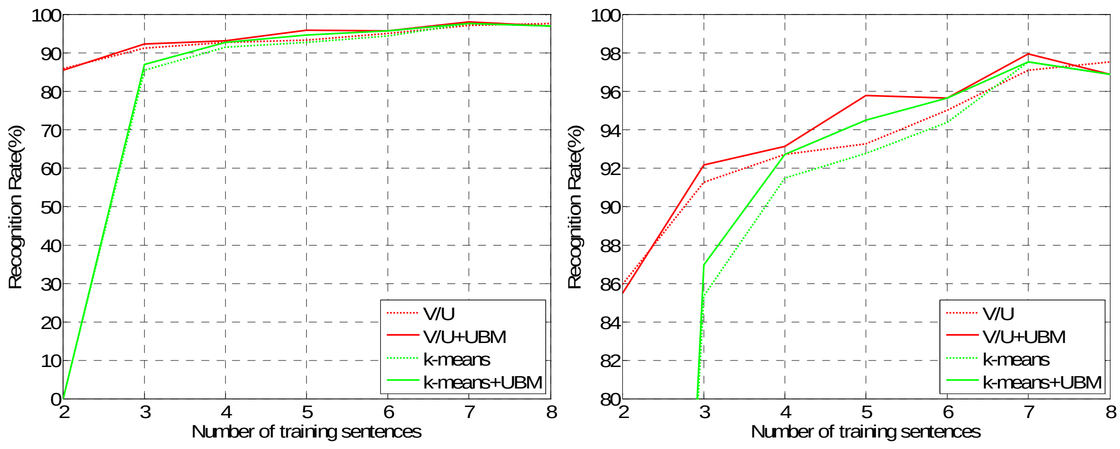


Fig 8.16 Maximum recognition rates for V/U and *k*-means systems with and without UBMs for different number of training sentences

8.3.1.2 CORPORA database

In the case of the CORPORA database, only 28 different speakers' speech is available. Therefore the sets formed by enrolled and impostor speakers will consist of 14 speakers each. The training data will be varied from 10 to 50 s in steps of 5 s. The number of tests for each speaker will be 10, each of them performed with a sentence of about 2.5 s. In this case, it has been possible to obtain models based on the speech recogniser (SR).

In Fig. 8.17 the DET curves for the three systems including or discarding the use of UBMs are represented for training durations of 15, 25, 35 and 45 s. It can be noticed how the performance of the SR system is strongly dependent on the length of the training data. It offers very poor results for 15 s but it is the winner when long training data is available. The *k*-means system suffers the same effect but with minor intensity. Once more the V/U system converges faster than the rest of the methods and should be chosen when very short training data is available. In the case of Polish speech, it can be seen how the difference in performance between the V/U and *k*-means systems is stronger and more. *k*-Means seems to offer a better performance for Polish speech which could be explained by a wider dispersion of the MFCCs for this language.

The use of UBMs is always positive even for the case of the SR system despite the fact that the UBMs where trained with speech original from the TIMIT database.

However, it can be noticed how the effect of using UBMs in this case does not produce such a strong performance increase as in the V/U or *k*-means systems.

In Fig. 8.18 the maximum recognition rate for each system is represented for different number of training sentences. Both figures are the same but different y-axis scale was used in order to provide an easier visualisation. These results confirm the previous conclusions extracted from Fig. 8.17. In addition, it can be seen how the SR systems is the last to converge. Furthermore, it was impossible to obtain reliable models for training duration inferior to 15 s. Now, the results for V/U and *k*-means are coherent as V/U always as poorer results than *k*-means when enough training data is available. The reason is that 10 tests per speaker are always possible since enough speech per speaker is available. Once more, UBMs always enhance the recognition results.

Next section will describe the experiments carried out to measure and compare each system performance in the presence of white gaussian noise during the testing stage.

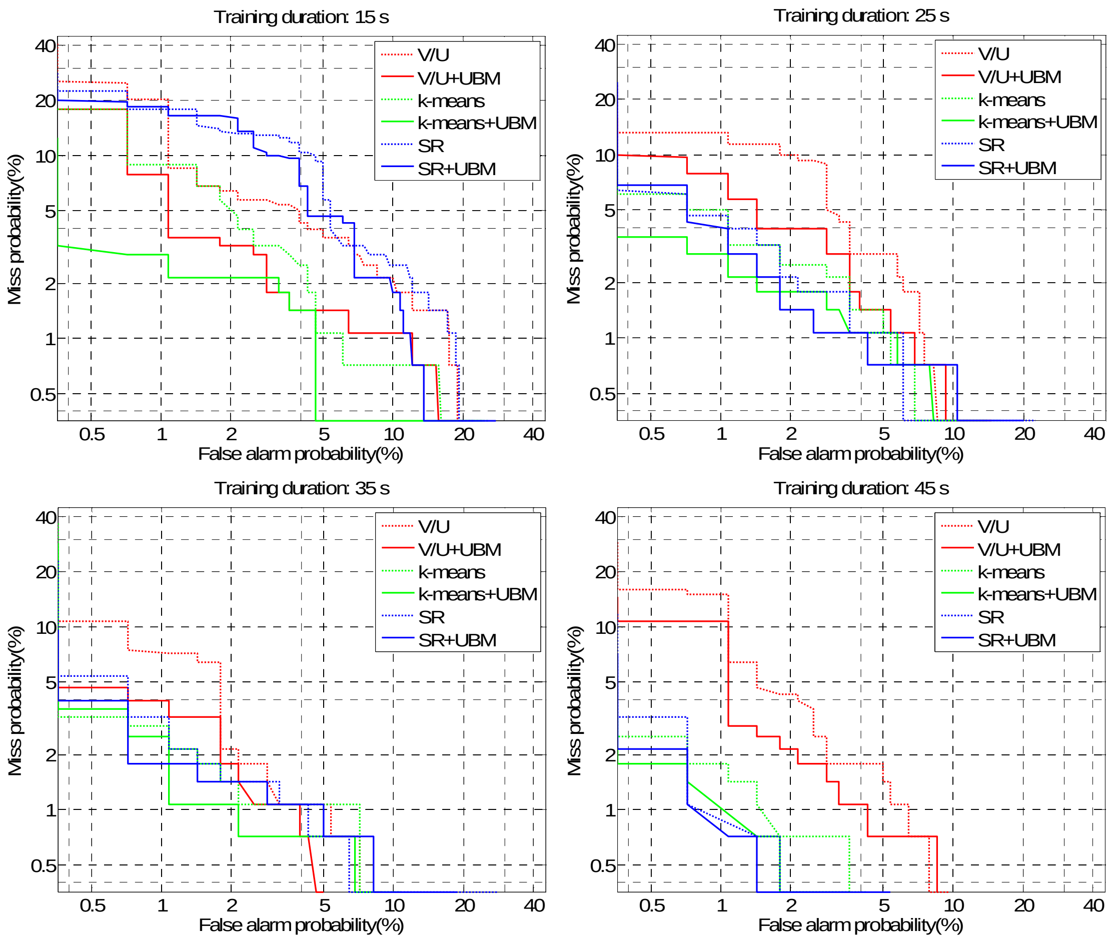


Fig 8.17 DET curves for V/U, *k*-means and SR systems with and without UBMs for different length of training data

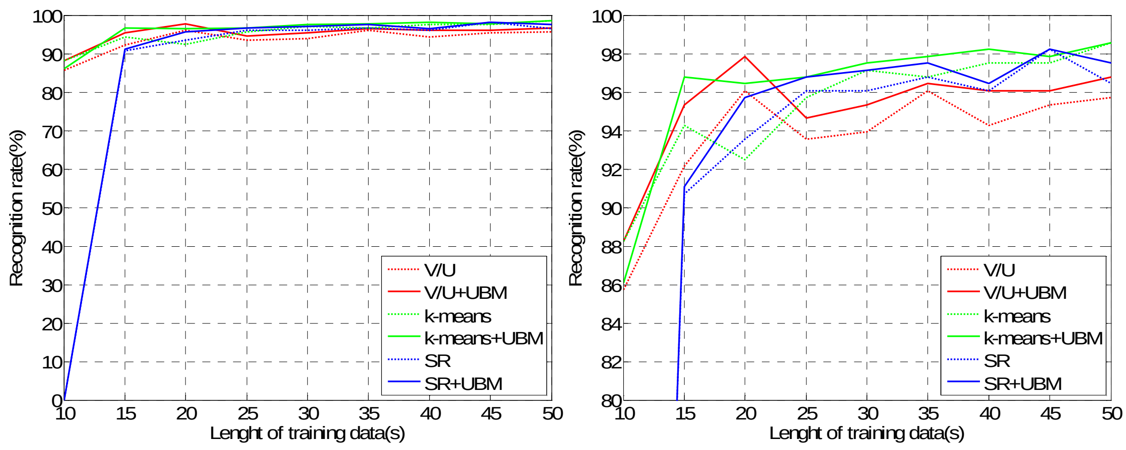


Fig 8.18 Maximum recognition rates for V/U, *k*-means and SR systems with and without UBMs for different length of training data

8.3.2 Performance vs. noise

The second parameter to be measured and contrasted for all systems is the robustness against noise. It will be measured by varying the SNR in the testing stage. The feature vectors, HMMs and GMMs configuration will be the same as in the previous experiment. However, the length of the training data will remain constant in this case.

This experiment has already been done somehow along all the experiments in section 8.2. However, no comparison was made among the three systems here presented. The two following sections will describe the experiments carried out on the TIMIT and CORPORA databases.

8.3.2.1 TIMIT database

Once more, 40 speakers will be enrolled in the system and other different 40 ones will form the impostors set. Training will be done with 6 sentences per speaker which sum around 15 s of speech. Testing will be performed by using 4 sentences which speaker each of them of around 2.5 s of duration.

Response of the V/U and *k*-means system against different noisy conditions will be tested by varying the SNR of the speech signal during the testing stage. Noise will be gaussian-like. The experiment will be done for both cases including or discarding UBMs.

In Fig. 8.19 the DET curves for this experiment are presented. The graphic in Fig. 8.20 corresponds to the maximum recognition rate achieved by each system for different values of the SNR. Both graphics show that the V/U system is more robust against noise whereas the *k*-means system offers higher performance when the speech signal is cleaner. The reason could be once more the larger number of classes into which voice is divided. Using fewer classes provides less accurate models. However, in the testing stage since no classification methods are used, the more models are being compared the more impact noise has in the recognition.

The performance of UBMs does not seem to be affected by the presence of noise. This method always enhances the recognition.

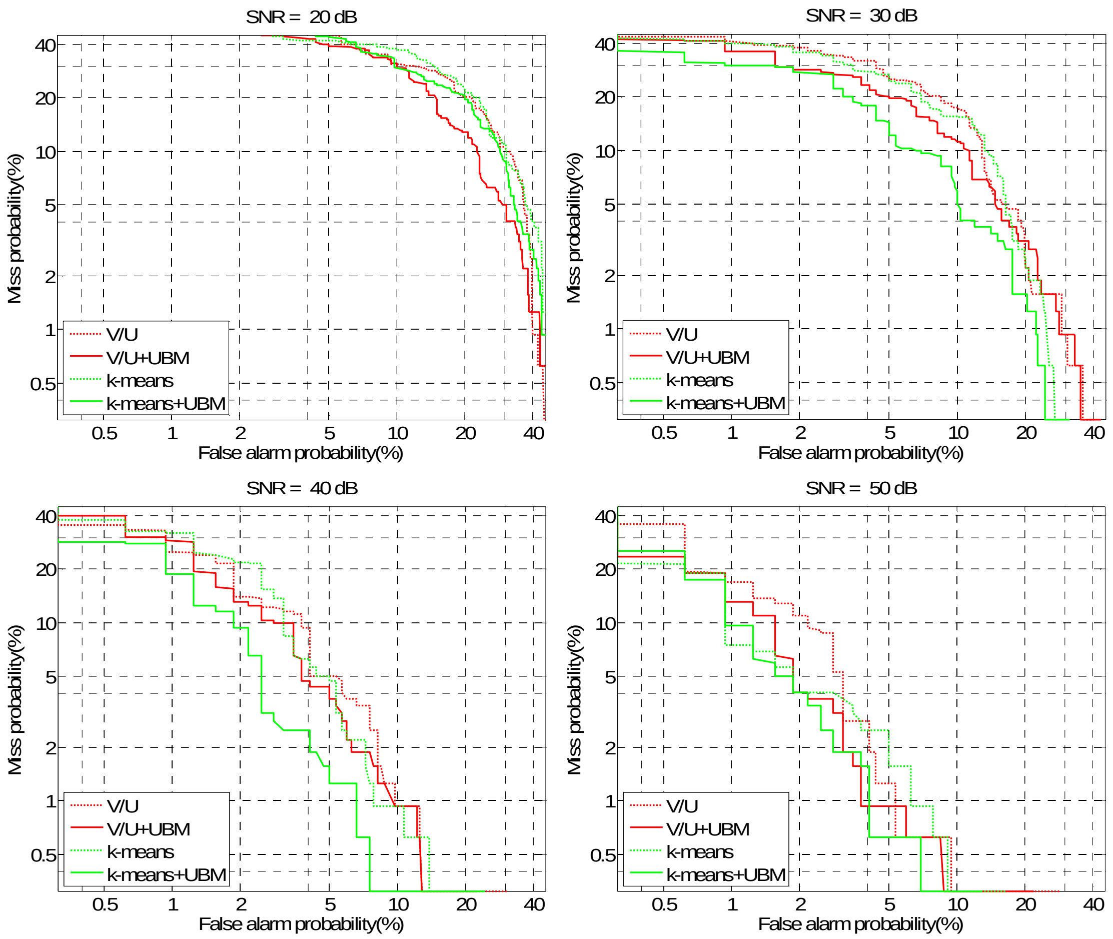


Fig 8.19 DET curves for V/U and *k*-means systems with and without UBMs for different SNRs in the testing stage

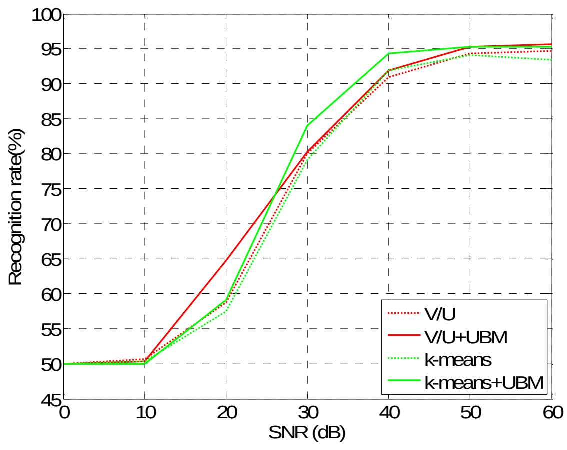


Fig 8.20 Maximum recognition rates for V/U and *k*-means systems with and without UBMs for different SNRs in the testing stage

8.3.2.2 CORPORA database

In this case the number of enrolled speakers and of impostors is 14 respectively. 20 s of speech per each speaker are used for training. Testing will consist of 10 utterances per speaker each of them around 2.5 s long.

The DET curves for each system and noise conditions are represented in Fig. 8.21 whereas the maximum recognition rates are plotted in Fig. 8.22. Once more, the methods which rely on a larger number of classes into which voice is split achieve the best results for noisy conditions. However, in this case the *k*-means and SR systems are highly affected by the presence of noise. Thus, it seems that Polish speakers are more difficult to recognise in the presence of noise. It is however odd that in this case the system with the highest performance in low noise conditions is the V/U system. The reason might be that very low noise power is enough to cause a strong distortion in the *k*-means and SR systems.

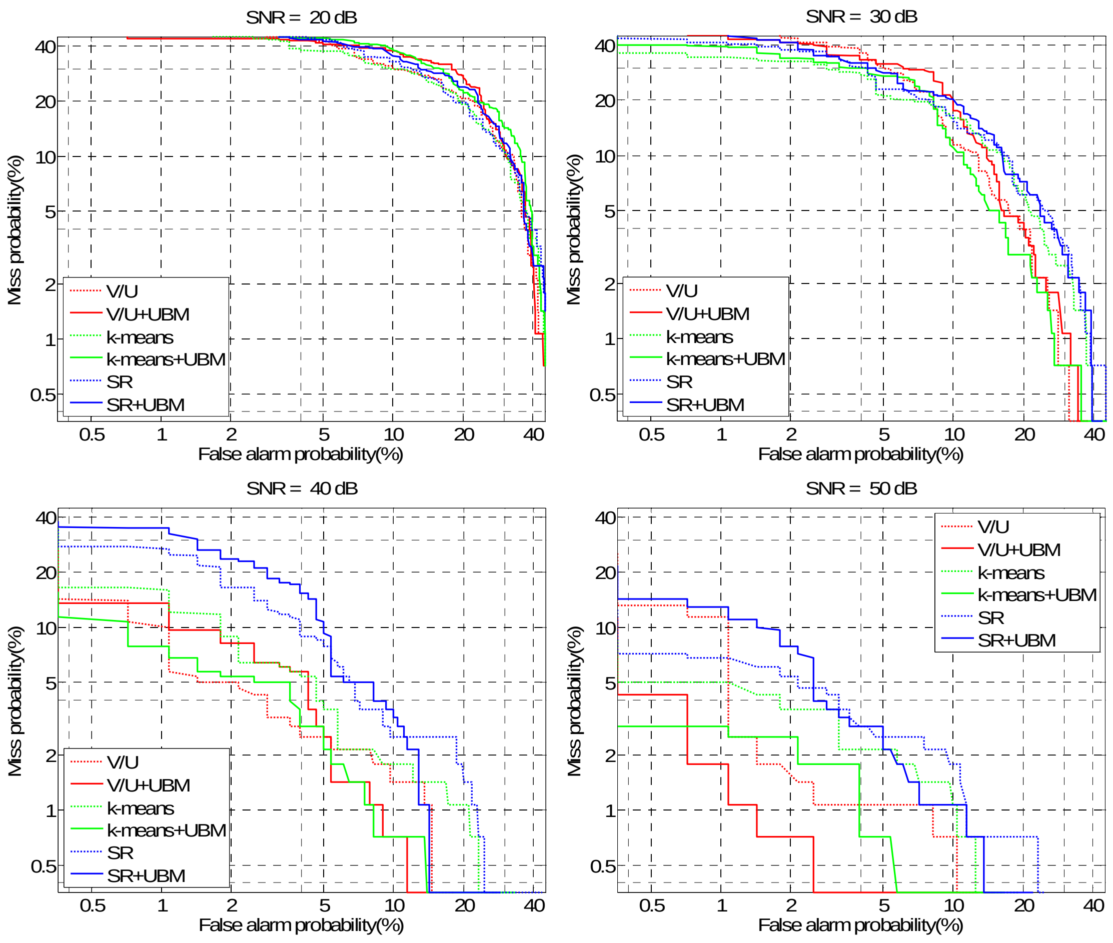


Fig 8.21 DET curves for V/U, *k*-means and SR systems with and without UBMs for different SNRs in the testing stage

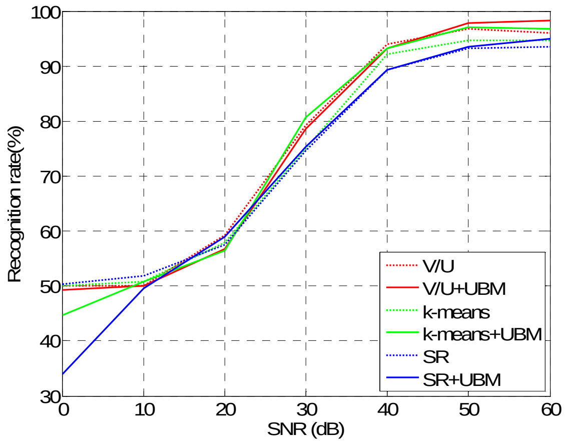


Fig 8.22 Maximum recognition rates for V/U, *k*-means and SR systems with and without UBMs for different SNRs in the testing stage

8.3.3 Performance vs. number of enrolled speakers

In this section, the performance of the proposed systems will be analysed for an increasing number of enrolled speakers. Since this experiment will require a wide variety of speakers, it can only be carried out by using the TIMIT database. CORPORA corpus only contains 28 different male speakers which is an insufficient population size.

In this case, 6 sentences which sum around 15 sec of speech will be used for training each speaker model. For testing 4 sentences per speaker will be used of about 2.5 s each. The speech signal in the testing stage is not corrupted by noise. The decision method is set to be the empirical one. In this case no UBMs will be used since some of the speakers who were used for training them take part of this experiment as enrolled or impostor speakers. In addition, only English speech is being used so the SR system can not be used.

In conclusion, the performance of the *k*-means and V/U systems will be compared for a number of enrolled speakers which will vary from 20 to 160 in steps of 20. The number of impostors will always be equal to the number of enrolled speakers.

The DET curves resulting in this experiment can be observed in Fig. 8.23. As it can be noticed, the results almost do not vary with the number of speakers. This is true for both the V/U and the *k*-means systems. However, it is also noticeable how the performance of the V/U system tends to get worse than the *k*-means one when the speakers population grows. Thus, it can be said that the performance of *k*-means is more robust against the number of speakers than the V/U system. However, the results for both systems are very satisfactory.

In Fig. 8.24 the maximum recognition rates achieved by each system by iterating the decision threshold value are presented. These results corroborate the conclusions extracted from Fig. 8.24 as the *k*-means system is slightly more accurate than the V/U one for an elevated population size. It has to be noticed that the scale in this graphic shows recognition rates from 90 to 100 % of accuracy. Thus, the differences between both systems are very short as the results oscillate between 95 and 97 % of accuracy.

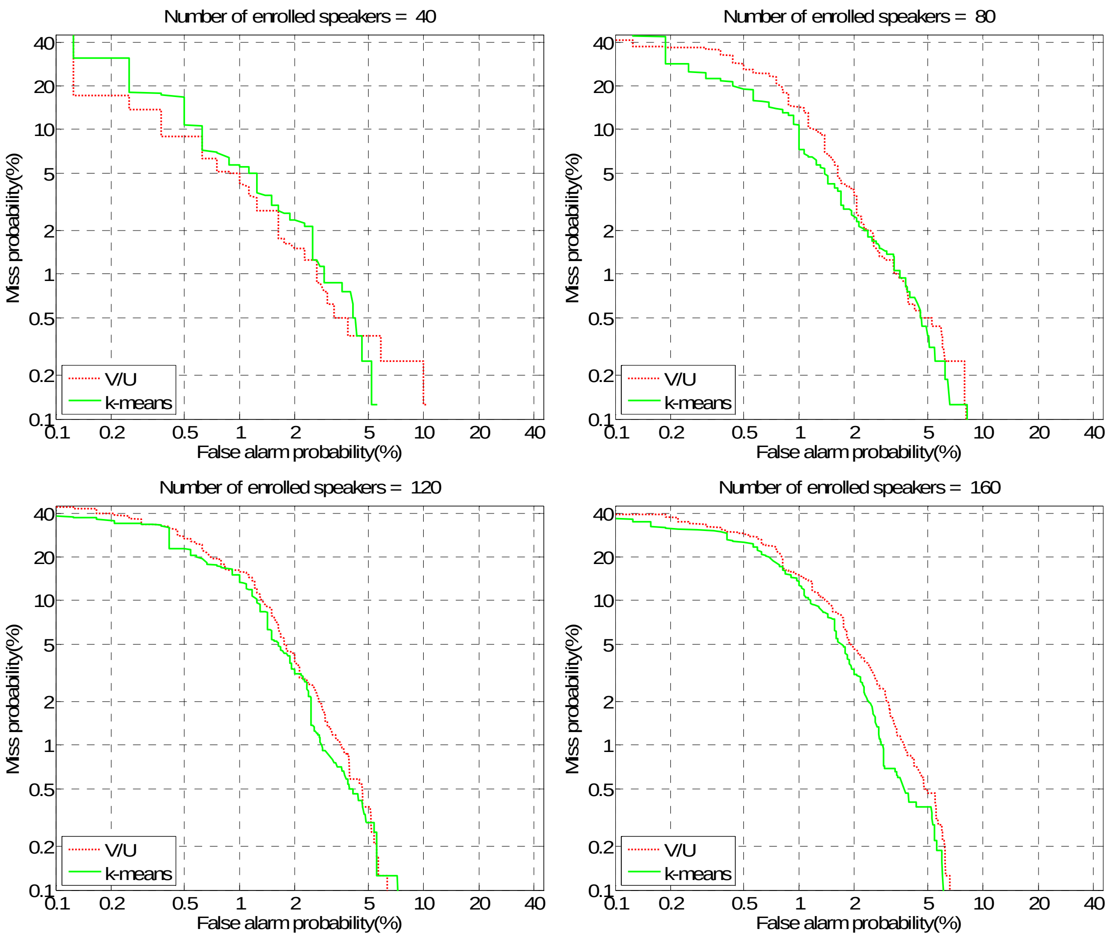


Fig 8.23 DET curves for V/U and *k*-means systems for different population sizes

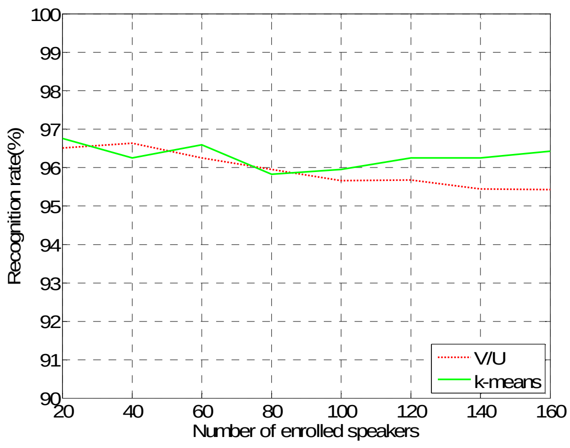


Fig 8.24 Maximum recognition rates for V/U and *k*-means systems for different population sizes

9 CONCLUSIONS AND FUTURE DIRECTIONS

Biometric keys are one of the latest advances in security systems. Using a mobile phone or a credit card no longer requires entering a password or a PIN which is difficult to remember. Inherent physical traits of the person are used instead. This thesis has presented voiceprints as one of the most promising and useful technologies belonging to the biometric security group.

The introduction of this thesis described the general issues and applications of speaker recognition. The aim of the thesis was established as trying to build a versatile speaker recognition system which could suit the widest possible range of application contexts. In addition, a general classification of this kind of systems was done. This classification explained how speaker recognition is a complex matter with very different fields the researcher can explore. Thus, this thesis has been focused on text-independent speaker identification since it matches the original purpose of the thesis. It is clearly the most general problem in speaker recognition and at the same time the most difficult to solve.

Such system was built monolithically by adding different subsystems. Each of them takes care of one task in the speaker identification procedure. In addition, two main functionalities of the system were defined. These functionalities are training, where stochastic models are obtained for each speaker; and testing, where the previously trained models are used to obtain a likelihood measure against an utterance produced by a speaker who has to be identified. In addition, two decision making algorithms were proposed in order to detect impostor access attempts.

Three different systems were proposed, the difference among them lies in the classification system used for clustering voice frames. The aim of this classification is to group speech frames which share certain properties in order to build more accurate models, one per speaker and per class. Three methods where defined, tested and compared.

Once these three systems were built, several experiments were carried out in order to tune their parameters for maximising the recognition rate. Some conclusions were reached in section 8.1.1. Firstly, increasing the dimensionality of the feature vectors as well as the spectral analysis resolution enhances the recognition results as it was expected. However, a limit can be found so that it is not worth increasing these values since the performance improvement is very low comparing to the computational cost increment. In addition, if a comparison is made against speech recognition systems, it is clear that the feature vectors resolution required is much higher in speaker than in speech recognition.

Some methods were presented to improve the performance of the basic MFCCs used as feature vectors in sections 5.1.6 and 5.1.7. The first one described in section 5.1.6 consists of using CMN for channel-dependent distortions removal. The performance of this method was tested in section 8.1.2. for different SNRs in the testing stage. The results were not satisfactory since the performance for low-noise channels was clearly worse than basic MFCCs. Furthermore the performance increment in noisy channels was almost inappreciable. Thus, it was stated that this method does not provide noise robustness to the systems here proposed.

Secondly, also in section 8.1.2 delta and acceleration coefficients were appended to the original MFCCs in order to use dynamic speech properties. Using these coefficients provided higher recognition rate. This fact proves that these coefficients can be good estimators of dynamic speaker-dependent information in the speech signal.

HMMs were chosen for pattern matching since they offer a good modelling of the sequential properties of speech. GMMs are incorporated to HMMs in order to provide good static parameterisation. The parameters used in both algorithms have been also tuned. The experiments carried out in sections 8.1.3 and 8.1.4 showed that short HMMs performed better than long ones. In addition, they need less training data to obtain convergent models. On the other hand, the larger number of gaussian components, the highest recognition rate. However, a trade-off between computational cost and accuracy has to be reached. In addition, too many mixture components derive in no convergent models.

Finally a new method for building speaker-dependent HMM architectures was proposed. It is based on changing the length of the HMMs for each speaker depending on some statistics obtained from the training data. This method enhanced the recognition in the closed-set case (section 8.2.1.1) where no impostors can try to break the system. However, this performance decreased in the open-set case where impostors can be detected. The reason could be that variable-length HMMs deform the likelihood scores distribution and therefore increase the error rate.

During the testing stage, the output of the pattern matching algorithm for an utterance is a likelihood score for each speaker. If just the maximum of this likelihood scores distribution is taken, the result of the recognition will be the ID of the most likely speaker who produced that utterance. However, this approach is not valid when the possibility that an impostor can try to break the system is considered. In this case a decision algorithm has to be implemented.

Two decision algorithms have been implemented and presented in sections 7.1.2 and 7.1.1. One of them has been empirically obtained after observation of the likelihood score distributions. The other one is based on Shannon entropy. Almost all the experiments showed that the empirical method beats the Shannon-based one.

Another method was introduced to enhance the decision making algorithm in section 7.2. The basics of this method consist in using UBMs in order to provide an extra model for the impostors. This method proved to enhance the decision making algorithm in all the experiments carried out. However, these results have to be reasoned since one has to be cautious when using UBMs. Firstly, they should be designed for a given population type with some shared speech attributes such as the intonation, melody and phonetics. Secondly the amount of data required for training them is extremely large. They have the advantage that they only have to be trained once and then they can be ported to different systems. However, a good UBM for a given population set can have no effect or even be harmful for other population groups.

Finally, the performance of the three systems here proposed was compared for different parameters. In section 8.3.1 the performance was compared for different lengths of the training data. The results showed a clear dependency on the number of classes used for voice classifying. The larger number of classes, the more training data is required. This is an intuitively reasonable result since training data will have to be distributed along a larger number of models. In addition, supposed language independence for V/U and *k*-means systems is not totally true. Some methods proved to work better for English or Polish. However, this was expected since the predominant sounds in both languages are clearly different.

Noise robustness is not one the aims of this thesis. However, it is clear that this is a critical factor in real world applications. In section 8.3.2 the performance of these systems was tested for different noise conditions. The results show that since no noise cancellation algorithms are being applied, the performance is strongly influenced by the presence of noise. When comparing the systems, the systems with the shorter number of classes seem to be more robust against noise. The phoneme-recogniser-based system which offers the highest performance among all the systems when both training and testing conditions are favourable is the most affected by noise.

Finally, the last experiment carried out in section 8.3.3 shows that the impact of an elevated population size does not alter significantly the results. This is an excellent result for real world applications where the number of speakers can grow to very high values. In addition, the advantage of using HMMs is that each speaker model is independent of the other ones. Therefore, enrolling a new user in the system does not require re-estimating the existing models. It is only necessary to obtain a new HMM for that user.

In conclusion this thesis has presented three methods for performing the text-independent speaker identification task. Choosing one of them is difficult since they offer different behaviours depending on some variables such as noise, number of speakers, and length of the training data. In addition, two of them can be used in any language but the third one is optimised for Polish language since a speech recogniser is being used.

Future works should focus in researching noise robustness so these methods can be used in real-world applications. In addition, experiments to measure the performance over different telephone channels can be very interesting from the telephone services point of view. MFCCs have been used as feature vectors. However other new parameterisations have latterly appeared. These parameterisations researched by Data [7], Gałka [10] and Sarikaya [29] are based on discrete wavelet transform (DWT) have proved to upgrade the performance of speech recogniser systems. It can be interesting to test how these parameterisations perform in speaker recognition systems.

10 BIBLIOGRAPHY

[1] Becchetti, C., Ricotti, L.P., *Speech Recognition, Theory and C++ Implementation*. John Wiley & Sons, 1999, pp. 122-143.

[2] Bimbot, F., Bonastre, J.F., Fredouille, C., Gravier, G., Magrin-Chagnolleau, I., Meignier, S., Merlin, T., Ortega-García, J., Petrovska-Delacrétaz D. and Reynolds, D.A., "A Tutorial on Text-Independent Speaker Verification", EURASIP Journal on Applied Signal Processing, April 2004, pp. 430–451.

[3] Boakye, K.A., "Speaker Recognition in the Text-Independent Domain Using Keyword Hidden Markov Models", M.Sc. Thesis, University of California at Berkeley, May 2005.

[4] Brookes, M., "VOICEBOX: Speech Processing Toolbox for MATLAB", Department of Electrical & Electronic Engineering, Imperial College, London,UK.

[5] Campbell, J.P., Jr., "Speaker recognition: a tutorial," *Proceedings of the IEEE* , vol.85, no.9, pp.1437-1462, Sep 1997

[6] Fant, G., Liljencrants, J., and Lin, Q., "A four parameter model of vocal flow", STL-QPSR April 1985, pp. 1-13.

[7] Farooq, O.; Datta, S., "Wavelet based robust sub-band features for phoneme recognition," *Vision, Image and Signal Processing, IEE Proceedings -* , vol.151, no.3, pp. 187-193, 1 June 2004

[8] Faundez-Zanuy, M.; Monte-Moreno, E., "State-of-the-art in speaker recognition," *Aerospace and Electronic Systems Magazine, IEEE* , vol.20, no.5, pp. 7-12, March 2005.

[9] Forney, G.D., Jr., "The viterbi algorithm," *Proceedings of the IEEE* , vol.61, no.3, pp. 268-278, March 1973

[10] Gałka, J., Kępiński, M., "WFT context-sensitive speech signal representation", *Advances in Soft Computing: Proceedings of the International Intelligent Information Systems, Intelligent Information Processing and Web Mining Conference*, Springer 2006

[11] Garcia, A.A.; Mammone, R.J., "Channel-robust speaker identification using modified-mean cepstral mean normalization with frequency warping," *Acoustics, Speech, and Signal Processing, 1999. ICASSP '99. Proceedings., 1999 IEEE International Conference on* , vol.1, no., pp.325-328 vol.1, 15-19 Mar 1999

[12] Garofolo, J., Lamel, L., Fisher, W., Fiscus, J., Pallett, D., Dahlgren, N., DARPA TIMIT acoustic-phonetic continuous speech corpus, 1993.

[13] Grocholewski, S., "Założenia akustycznej bazy danych dla języka polskiego na nośniku cd rom (eng. Assumptions of acoustic database for Polish language)," *Mat. I KK: Głosowa komunikacja człowiek-komputer*, Wrocław, 1995.

[14] Guorong Xuan; Wei Zhang; Peiqi Chai, "EM algorithms of Gaussian mixture model and hidden Markov model", *Image Processing, 2001. Proceedings. 2001 International Conference on* , vol.1, no., pp.145-148 vol.1, 2001.

[15] HTK Toolkit v3.3. Cambridge University Engineering Department. http://htk.eng.cam.ac.uk/.

[16] Kinnunen, T., "Spectral Features for Automatic Text-Independent Speaker Recognition", Licenciate's thesis, University of Joensuu Department of Computer Science, 2003.

[17] Li, K.-P.; Porter, J.E., "Normalizations and selection of speech segments for speaker recognition scoring," *Acoustics, Speech, and Signal Processing, 1988. ICASSP-88., 1988 International Conference on* , vol., no., pp.595-598 vol.1, 11-14 Apr 1988.

[18] Martinez, W.L., Martinez, A.R., *Exploratory Data Analysis with Matlab*, Computer Science and Data Analysis Series, Chapman &Hall/CRC Press, 2005.

[19] Matlab and Simulink for Technical Computing. The Mathworks. www.mathworks.com.

[20] Moon, T.K., "The Expectation-Maximization Algorithm", IEEE Signal Processing Magazine, vol. 13, pp. 47-60, Nov. 1996.

[21] Nakagawa, S.; Zhang, W.; Takahashi, M., "Text-independent speaker recognition by combining speaker-specific GMM with speaker adapted syllable-based HMM," *Acoustics, Speech, and Signal Processing, 2004. Proceedings. (ICASSP '04). IEEE International Conference on* , vol.1, no., pp. I-81-4 vol.1, 17-21 May 2004.

[22] Oppenheim, A.V., Schafer, R. W.; and Buck, J. R., *Discrete-time Signal Processing*, Upper Saddle River, N.J.: Prentice Hall, 1999.

[23] Quatieri, T.F., *Discrete-Time Speech Signal Processing, Principles and Practice*, Prentice Hall PTR, 2001.

[24] Rabiner, L.R., "A tutorial on hidden Markov models and selected applications in speech recognition," *Proceedings of the IEEE* , vol.77, no.2, pp.257-286, Feb 1989.

[25] Rabiner, L.R., Schafer, R.W., *Digital Processing of Speech Signals*, Prentice Hall, 1978.

[26] Reynolds, D.A., "Comparison of background normalization methods for text- independent speaker verification," in *Proc. Eurospeech '97*, Rhodes, Greece, 1997, pp. 963–966.

[27] Reynolds, D.A.; Rose, R.C., "Robust text-independent speaker identification using Gaussian mixture speaker models," *Speech and Audio Processing, IEEE Transactions on* , vol.3, no.1, pp.72-83, Jan 1995

[28] Rong Zheng; Shuwu Zhang; Bo Xu, "Text-independent speaker identification using GMM-UBM and frame level likelihood normalization," *Chinese Spoken Language Processing, 2004 International Symposium on* , vol., no., pp. 289-292, 15-18 Dec. 2004

[29] Sarikaya, R., Hansen, J. H. L., "High Resolution Speech Feature Parameterization for Monophone – Based Stressed Speech recognition", *IEEE Signal Processing Letters*, Vol. 7, No. 7, 2000

[30] Shannon, C.E., "A Mathematical Theory of Communication", *Bell System Technical Journal*, vol. 27, pp. 379-423, 623-656, July, October, 1948

[31] Talkin, D., "A robust algorithm for pitch tracking (RAPT)", in Speech coding and Synthesis, W.B.Klejin and K.K.Paliwal, Eds. Elseiver Science, 1995.

[32] Viterbi, A., "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *Information Theory, IEEE Transactions on* , vol.13, no.2, pp. 260-269, Apr 1967.

[33] White, G. B., Fisch E.A., Pooch, U. W., *Computer System and Network Security*, CRC Press, 1995

[34] Young, S., Odell, J., Ollason, D., Valtchev, V., Woodland, P., *The HTK Book. Version 3.3*, Cambridge University Press, UK, 2005.

[35] Zheng, F., Guoliang, Z., and Song, Z., "Comparison of Different Implementations of MFCC", Journal of Computer Science and Technology, vol. 16, iss. 6, November 2001.

[36] Ziółko, B., "Analysis of Phonetic Similarities in Wrong Recognitions of the Polish Language", *ICALIP,* 2008.