Departament de Teoria
del Senyal i Comunicacions de Terrassa

An Automatic Audio Classification System for Radio Newscast

Final Project

ADVISOR

Professor Ignasi Esquerra

STUDENT

Giuseppe Dimattia

March 2008

Preface

The work presented in this thesis has been carried out at the Department of Signal Theory and Communications (TSC) at the UPC Terrassa. The thesis is the final requirement for obtaining the Master degree in Telecommunication Engineering. The work has been supervised by the professor Ignasi Esquerra.

CONTENTS

Abstract

Current web search engines generally do not enable searches into audio files. Informative metadata would allow searches into audio files, but producing such metadata is a tedious manual task. Tools for automatic production of metadata are therefore needed. This project describes the work done on the development of an automatic audio classification system which can be used for this metadata extraction. In order to design this system I used adapting it to our case of study, the matlab code of the MPEG-7 Experimental Model [15].

At the beginning of this work I presented the MPEG-7 Low Level Descriptor and their possible application for the classification of piece of radio newscast. Then in order to find a balanced tradeoff between reducing dimensionality and retaining maximum information content I performed the feature transformation algorithm presenting in the MPEG-7 Standard on the Normalize Audio Spectrum Envelope (NASE) descriptor.

This de-correlated dimension-reduced log-spectral feature is used to train a classifier based on continuous hidden Markov models. Various audio classification experiments are presented, in which audio sounds are classified into selected sound classes.

On the base of the results of these experiments I designed a hierarchical classification system.

In order to apply this classification system to an entire radio newscast track I used the segmentation system developed by Vincenzo Dimattia in the same laboratory in his thesis [37].The segments obtained by this automatic audio segmentation and classification system were then described in an MPEG-7 compliant XML document.

CHAPTER 1
INTRODUCTION

The amount of audio available in different databases on the Internet today is immense. Successful access to such large amounts of data requires efficient search engines. Traditional web search engines, such as Google, are often limited to text and image indexing, thus many multimedia documents, video and audio, are excluded from these classical retrieval systems. Even systems that do allow searches for multimedia content, like AltaVista and Lycos, only allow queries based on the multimedia filename, nearby text on the web page containing the file, and metadata embedded in the file such as title and author. This might yield some useful results if the metadata provided by the distributor is extensive. Producing this data is a tedious manual task, and therefore automatic means for creating this information is needed.

Today many radio stations provide entire news or talk shows in form of podcasts or streaming services, with general headlines of the content. In general no detailed index of the file is provided, which makes it time consuming to look for the part of interest.

The optimal division of radio shows, such as broadcast news, debate programs, and music programs, should be based on the topics covered in each part. Such a division requires that it is possible extract topics and the parts related to this topic. Topic detection requires transcription of the speech parts of the audio, but adding audio cues would aid in retrieving coherent segments.

Adding audio cues is done by segmenting the audio based on the characteristics of the audio stream. The segments generated can then be summarized on basis of the type of audio.

- Music clips would be described by genre and artist.
- Speech summaries naturally and would consist of identities of speakers and a transcription of what is said.

1.1 Audio Retrieval Systems

Research in audio retrieval has mainly been focused on music and speech. Music retrieval has focused on quantifying different characteristics of the music such as mood, beat, and other characteristics to classify genre or find similarities between songs. This area is not the focus of this thesis and will not be covered further.

Approaches to spoken document retrieval have included automatic broadcast news transcription and other speech retrieval systems.

1.1.1 Automatic Broadcast News Transcription

Broadcast news transcription system has been heavily researched and is considered to be the most demanding assignment in speech recognition, as the speakers and conditions vary much in the course of a news show. The speakers range from anchor speaking in an ideal environment to reporters speaking from noisy environments on a non-ideal telephone line. Non-native English speakers make the problem even more challenging. The shows often use music between different parts of the show and in the background of speakers.

These systems therefore often include segmentation preprocessor that removes the non-speech parts and finds segments with homogenous acoustical characteristics. In this way the speech recognizers can be adjusted to the differing acoustical environments. Solving the task has required advances in both preprocessing and speech decoding.

The performances of these systems are evaluated in annual Rich Transcription workshops arranged by [1]. The participants include commercial groups such as IBM and
BBN and academical groups such as LIMSI and CU-HTK, whose systems are described in [2] and [3], respectively.

1.2 Segmentation Approaches

The segmentation approaches used in the systems mentioned above and in other retrieval systems covers a wide range of methods. In general the methods can be divided into two groups: audio classification and change detection. These approaches have different attributes that qualify them for different uses as presented below.

1.2.1 Audio Classification

As mentioned above the typical first part of a speech retrieval system concerns identifying different audio classes. The four main classes considered are speech, music, noise, and silence but depending on the application more specific classes such as noisy speech, speech over music, and different classes of noise, have been considered.

The task of segmenting or classifying audio into different classes has been implemented using a number of different schemes. Following the paper by [4] a multitude of approaches have

been proposed. Two aspects that must be considered are feature and classification model selection.

Different features have been proposed, based on different observations on the characteristics that separate speech, music and other possible classes of audio. The features are generally divided on basis of the time horizon they are extracted.

The simplest features proposed include time domain and spectral features. Time domain features typically represent a measure of the energy or zero crossing counts.

Cepstral coefficients have been used with great success in speech recognition systems, and subsequently have shown to be quite successful in audio classification tasks as well [2]. Other features have also been proposed based on psychoacoustic observations, see e.g., [5].

The other aspect to be considered is the classification scheme to use. A number of classification approaches have been proposed that can be divided into rule-based and model-based schemes. The rule-based approaches use some simple rules deducted from the properties of the features. As these methods depend on thresholds, they are not very robust to changing conditions, but may be feasible for real-time implementations.

Model-based approaches have included Maximum A Posteriori (MAP) classifiers, Gaussian Mixture Model (GMM), K-nearest-neighbor (K-NN), and linear perceptrons. Another approach in this context is to model the time sequence of features, or the probability of switching between different classes. Hidden Markov Models (HMM) take this into account.

1.2.2 Speaker Change Detection

Another approach to identify homogenous audio segments could be done by performing event detection. Approaches to speaker change detection can be divided into supervised and unsupervised methods. If the number of speakers and identities are known in advance, supervised models for each speaker can be trained, and the audio stream can be classified accordingly. If the identities of the speakers are not known in advance unsupervised methods must be employed.

1.3 Audio Feature Extraction

Feature extraction is the process of converting an audio signal into a sequence of feature vectors carrying characteristic information about the signal. These vectors are used as basis for various types of audio analysis algorithms. It is typical for audio analysis algorithms to be

based on features computed on a window basis. These window based features can be considered as short time description of the signal for that particular moment in time.

Feature extraction is a very important issue to get optimal results in this application. Extracting the right information from the audio increases the performance of the system and decrease the complexity of subsequent algorithms.

Generally applications require different features enhancing the characteristics of the problem.

A wide range of audio features exist for classification tasks. These features can be divided into two categories: time domain and frequency domain features.

In the frequency domain, spectral descriptors are often computed from the Short Time Fourier Transform (STFT). By combining this measurement with perceptually relevant information, such as accounting for frequency and temporal masking, one can produce an auditory spectrogram which can then be used to determine the loudness, timbre, onset, beat and tempo, and pitch and harmony [6]. In addition to spectral descriptors, there also exist temporal descriptors, which are composed of the audio waveform and its amplitude envelope, energy descriptors, harmonic descriptors, derived from the sinusoidal harmonic modeling of the signal, and perceptual descriptors, computed using a model of the human hearing process.

1.4 MPEG-7

In September 2001, Moving Picture Experts Group (MPEG) specified an international standard called Multimedia Content Description Interface, or MPEG-7.

MPEG-7 is the first complete work for description of multimedia data. MPEG-7 is a work to define a standard set of descriptors for various types of multimedia data and methods to define other descriptors as well as structures of descriptors and their relationships. Although MPEG-7 is not aimed at any particular application area, one of its main applications areas will be searching and retrieving multimedia content.

This standard, also known as "Multimedia Content Description Interface," provides a standardized set of technologies for describing multimedia content. The standard addresses a broad spectrum of multimedia applications and requirements by providing a metadata system for describing the features of multimedia content [8].

The following are specified in this standard:

- *Description Schemes (DS)* describe entities or relationships pertaining to multimedia content. Description Schemes specify the structure and semantics of their components, which may be Description Schemes, Descriptors, or datatypes.
- *Descriptors (D)* describe features, attributes, or groups of attributes of multimedia content.
- *Datatypes* are the basic reusable datatypes employed by Description Schemes and Descriptors
- *Description Definition Language (DDL)* defines Description Schemes, Descriptors, and Datatypes by specifying their syntax, and allows their extension.
- *Systems tools* support delivery of descriptions, multiplexing of descriptions with multimedia content, synchronization, file format, and so forth.

1.4.1 Description Schemes (DSs)

Description Schemes (DSs) specify the structure and semantics of the relationships between their components, which may be both Descriptors and Description Schemes. The DSs provide a standardized way of describing in XML the important concepts related to AV content description and content management in order to facilitate searching, indexing, filtering, and access.

The DSs are defined using the MPEG-7 Description Definition Language (DDL), which is based on the XML Schema Language, and are instantiated as documents or streams. The resulting descriptions can be expressed in a textual form (i.e., human readable XML for editing, searching, filtering) or compressed binary form (i.e., for storage or transmission).

The MPEG-7 DSs are designed primarily to describe regions, segments, objects, events which are high-level AV features and also some constant metadata like title, author, and creation date. By combining Ds, DSs and presenting relationships between them complex descriptions can be defined. In MPEG-7, the DSs are categorized according to their related media such as audio, visual domain or multimedia. Typically, the multimedia DSs describe content consisting of a combination of audio, visual data, and possibly textual data, whereas, the audio or visual DSs refer specifically to features unique to the audio or visual domain, respectively. [7] The DSs can be grouped into 5 different classes according to their functionality:

- *Content description:* Representation of perceivable information

- *Content management:* Information about the media features, the creation and the usage of the AV content;
- *Content organization:* Representation the analysis and classification of several AV contents;
- *Navigation and access:* Specification of summaries and variations of the AV content;
- *User interaction:* Description of user preferences and usage history pertaining to the consumption of the multimedia material.

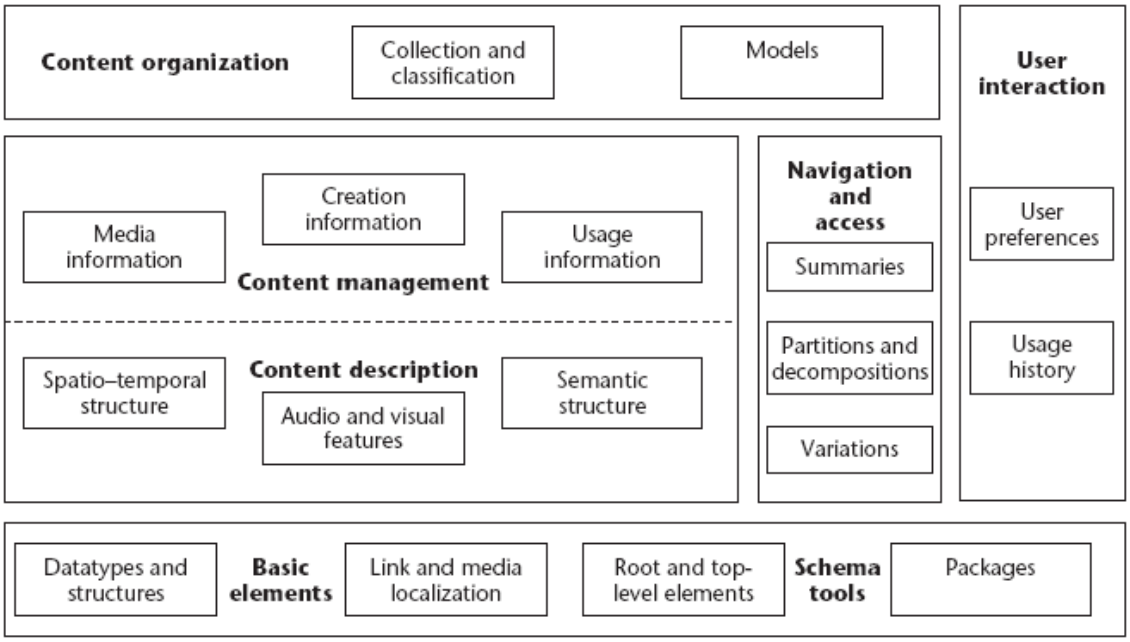

Figure 1.1: Overview of the MPEG-7 Multimedia DSs

1.4.2 MPEG-7 Description Definition Language (DDL)

For MPEG-7, one of the main works was to define a language (Description Definition Language) to describe the Ds and DSs. During the design process of Description Definition Language (DDL), the designers concerned to achieve following points by DDL:

- Prevention of being sophisticated in other words satisfying the main features and also ability to be extended;
- Coverage of MPEG-7 requirements and usage of MPEG-7 Ds, DSs;
- Usage of XML structure and interoperability with XML Schema.

The DDL is the language which allows the creation of MPEG-7 DSs and Ds. A DDL schema (a DDL file) specifies the constraints that a valid MPEG-7 description should respect. It is

encoded in XML. The DDL uses XML Schema for interoperability reasons. However because multimedia content descriptions require specific features that are not defined in XML Schema (such as array and matrix data types), the DDL adds those features to the language. Hence the MPEG-7 DDL adopts most the of the XML Schema specification and adds MPEG-7-specific mechanisms on top of it. Some of the important issues, which are supported by DDL, are extended XML Schema, structural and datatype constraints over Ds and DSs, usage of Ds, DSs in other DSs, and group definitions like Attribute Group for simple DSs definitions.

With the DDL one can express structural constraints and data type constraints. Structural constraints specify the rules that a valid description should respect in terms of inclusion of elements. Data type constraints specify the type and the possible values for data within the description. The DDL allows defining complexTypes and simpleTypes. The complexTypes specify the structural constraints while simpleTypes express datatype constraints. Moreover the DDL allows reusing the existing complexTypes or simpleTypes in a way similar to inheritance in object-oriented programming [14].

1.4.3 MPEG-7 System Tools

Some applications of MPEG-7's system tools are support binary coded representation for efficient storage and transmission, transmission mechanisms (both for textual and binary formats), error resilience, multiplexing of descriptions, synchronization of descriptions with content, management and protection of intellectual property in MPEG-7 descriptions. These issues result in the ability to search, browse, and filter MPEG-7 descriptions.

1.5 XML

XML [10] is a text based format to represent hierarchical data. XML uses named tags enclosed between angle brackets to mark the begin and the end of the hierarchical organizers, the XML elements. Elements contain other elements, attributes and plain content.

The power of XML is that you can adapt your own tags (elements) and tag attributes (attributes) in order to describe your own data. Another important advantage of the XML format is that it is structured and human-readable. For these reasons XML is starting to spread rapidly as a multimedia description language (see MPEG7 language [11], for instance). On the downsides, its main inconvenience is that, because it is a textual format, it is very inefficient both in size and in loading/storing speed. The XML specification defines the concepts of well-formedness and validity.

An XML document is well-formed if it has a correct nesting of tags. In order for a document to be valid, it must conform to some constraints expressed in its document type definition (DTD) or its associated XML Schema.

On the other hand XML-Schema [12] is a definition language for describing the structure of an XML document using the same XML syntax and it is bound to replace the existing DTD language. The purpose of a schema is to define a class of XML documents by using particular constructs to constrain their structure: data types, elements and their content, attributes and their values. Schemas are written in regular XML and this allows users to employ standard XML tools instead of specialized applications.

1.6 MPEG-7 Standard parts

This standard is subdivided into eight parts:

- *Part 1 – Systems*: specifies the tools for preparing descriptions for efficient transport and storage, compressing descriptions, and allowing synchronization between content and descriptions.
- *Part 2 – Description Definition Language*: specifies the language for defining the standard set of description tools (DSs, Ds, and data types) and for defining new description tools.
- *Part 3 – Visual:* specifies the description tools pertaining to visual content.
- *Part 4 – Audio:* specifies the description tools pertaining to audio content [9].
- *Part 5 – Multimedia Description Schemes:* specifies the generic description tools pertaining to multimedia including audio and visual content [13].
- *Part 6 – Reference Software*: provides a software implementation of the standard.
- *Part 7 – Conformance:* specifies the guidelines and procedures for testing conformance of implementations of the standard.
- *Part 8 – Extraction and Use*: provides guidelines and examples of the extraction and use of descriptions.

In this work the parts of the MPEG-7 Standard which I have used as references are only the part 4 ,5 and 6.

1.6.1 Part 4: MPEG-7Audio

MPEG-7 Audio provides structures—in conjunction with the Multimedia Description Schemes part of the standard (Part 5)—for describing audio content. Utilizing those structures are a set of low-level Descriptors, for audio features that cut across many applications (e.g., spectral, parametric, and temporal features of a signal), and high-level Description Tools that are more specific to a set of applications. Those high-level tools include general sound recognition and indexing Description Tools, instrumental timbre Description Tools, spoken content Description Tools, an audio signature Description Scheme, and melodic Description Tools to facilitate query-by-humming [14].

1.6.2 Part 5: MPEG-7 Multimedia Description Schemes

MPEG-7 Multimedia Description Schemes (also called MDS) comprises the set of Description Tools (Descriptors and Description Schemes) dealing with generic as well as multimedia entities.

Generic entities are features, which are used in audio and visual descriptions, and therefore "generic" to all media. These are, for instance, "vector", "time", textual description tools, controlled vocabularies, etc.

The MDSs are metadata structures for describing and annotating multimedia content, and provide a way to describe in XML the important concepts related to multimedia content. The objective is to allow interoperable searching, indexing, filtering and access by enabling interoperability among devices that deal with multimedia content description.

1.6.3 Part 6: MPEG-7 Reference Software: the eXperimentation Model

The eXperimentation Model (XM) software is the simulation platform for the MPEG-7 Descriptors (Ds), Description Schemes (DSs), Coding Schemes (CSs), and Description Definition Language (DDL). Besides the normative components, the simulation platform needs also some non-normative components, essentially to execute some procedural code to be executed on the data structures. The data structures and the procedural code together form the applications [14].

1.7 Project Overview

The remainder of this thesis is organized into the following chapters:

- *Chapter 2* presents the MPEG-7 Low Level Descriptor and shows some example of this descriptor extract from audio segments of radio newscasts.
- *Chapter 3* presents some algorithms applied to the feature vector in order to reduce their dimensions.
- *Chapter 4* presents the classifier used in the classification system based on continuous hidden Markov models (HMM).
- *Chapter 5* describes the audio database which I have create in order to train and test the classifier and besides various audio classification experiments are presented, in which audio sounds are classified into selected sound classes.
- *Chapter 6* describes how can connect the classification system to a segmentation system and presents more in detail the MDS tools in order to create automatically an MPEG-7 Compliant XML document for the description of the radio newscast segments.

CHAPTER 2

MPEG-7 LOW LEVEL DESCRIPTORS

The MPEG-7 low-level audio descriptors are of general importance in describing audio. There are 17 temporal and spectral descriptors that may be used in a variety of applications. These descriptors can be extracted from audio automatically and depict the variation of properties of audio over time or frequency. Based on these descriptors it is often feasible to analyze the similarity between different audio files. Thus it is possible to identify identical, similar or dissimilar audio content. This also provides the basis for classification of audio content [16].

The low level audio descriptors can be roughly divided into the following groups:

- *Basic Spectral Descriptors*: These are time domain descriptions of the audio content.
- *Basic Spectral Descriptors*: The four basic spectral audio descriptors are all derived from a single time–frequency analysis of an audio signal. They describe the audio spectrum in terms of its envelope, centroid, spread and flatness.
- *Signal Parameters Descriptor*s: The two signal parameter descriptors apply only to periodic or quasi-periodic signals. They describe the fundamental frequency of an audio signal as well as the harmonicity of a signal.
- *Temporal Timbral Descriptors*: Timbral temporal descriptors can be used to describe temporal characteristics of segments of sounds. They are especially useful for the description of musical timbre (characteristic tone quality independent of pitch and loudness);
- *Spectral Timbral Descriptors* Timbral spectral descriptors are spectral features in a linear frequency space, especially applicable to the perception of musical timbre;

  The Timbral Descriptors are not adapted to describe audio segments extract from radio newscasts so will not be covered further.
- *Spectral basis representations*: The two spectral basis descriptors represent low-dimensional projections of a high-dimensional spectral space to aid compactness and recognition. These descriptors are used primarily with the sound classification and indexing description tools, but may be of use with other types of applications as well.

  Those descriptors are covered separately in chapter 3.

2.1 Basic parameters and notations

There are two ways of describing low-level audio features in the MPEG-7 standard:

- An LLD feature can be extracted from sound segments of variable lengths to mark regions with distinct acoustic properties. In this case, the summary descriptor extracted from a segment is stored as an MPEG-7 *Audio Segment* description. An audio segment represents a temporal interval of audio material, which may range from arbitrarily short intervals to the entire audio portion of a media document.
- An LLD feature can be extracted at regular intervals from sound frames. In this case, the resulting sampled values are stored as an MPEG-7 *ScalableSeries* description.

This section provides the basic parameters and notations that will be used to describe the extraction of the frame-based descriptors.

2.1.1 Time Domain

In the time domain, the following notations will be used for the input audio signal:

- l is the index of time frames.
- $hopSize$ is the time interval between two successive time frames.
- $Nhop$ denotes the integer number of time samples corresponding to $hopSize$.
- Lw is the length of a time frame (with $Lw \geq hopSize$).
- Nw denotes the integer number of time samples corresponding to Lw.
- L is the total number of time frames in $s(n)$.

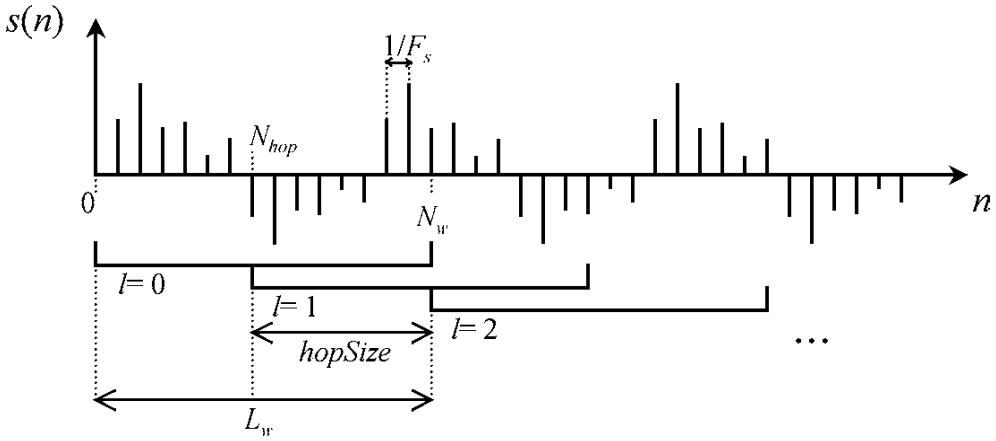

Figure 2.1: Notations for frame-based descriptors

The choice of $hopSize$ and Lw depends on the kind of descriptor to extract.

However, the standard constrains $hopSize$ to be an integer multiple or divider of 10 ms (its default value), in order to make descriptors that were extracted at different $hopSize$ intervals compatible with each others.

2.1.2 Frequency Domain

The extraction of some MPEG-7 LLDs is based on the estimation of short-term power spectra within overlapping time frames. In the frequency domain, the following notations will be used:

- k is the frequency bin index.
- $S_l(k)$ is the spectrum extracted from the lth frame of s(n).
- $P_l(k)$ is the power spectrum extracted from the lth frame of s(n).

MPEG-7 does not standardize the technique for spectrum estimation, a number of implementation features are recommended (e.g. an Lw of 30 ms for a default $hopSize$ of 10 ms).

2.2 Scalable Series

An LLD may be instantiated either as a single value for a segment or a sampled series. Both of these possibilities are embodied in two LLD types. $AudioLLDScalarType$ is inherited for scalar values, such as power or fundamental frequency, and $AudioLLDVectorType$ is inherited for vector types, such as spectra. Any descriptor inheriting from these types can be instantiated, describing a segment with a single summary value or a series of sampled values, as the application requires.

The sampled values themselves may be further manipulated through another unified interface: they can form a scalable series. The scalable series allows one to progressively down-sample the data contained in a series, as the application, bandwidth or storage require. In fact a series can be described at full resolution or after a *scaling* operation. In the latter case, the series of original samples is decomposed into consecutive sub-sequences of samples. Each sub-sequence is then summarized by a single *scaled sample*.

An illustration of the scaling process and the resulting scalable series description is shown in figure 2.2 [9], where 'k' is an index in the scaled series. In this figure, the 31 samples of the original series (filled circles) are summarized by 13 samples of the scaled series (open circles).

The first three scaled samples each summarizes two original samples, the next two six, the next two one, etc.

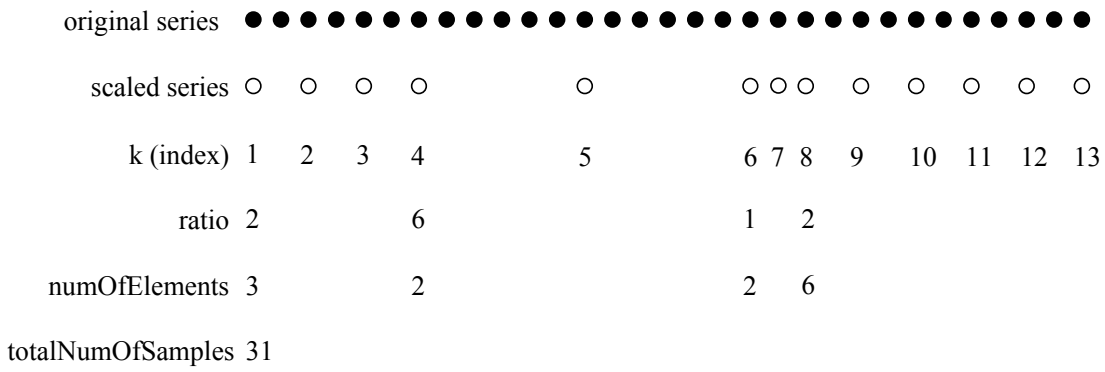

Figure 2.2: Structure of a scalable series description

The attributes of a $ScalableSeries$ are the following:

- $Scaling$: is a flag that specifies how the original samples are scaled. If absent, the original samples are described without scaling.
- $totalNumOfSamples$: indicates the total number of samples of the original series before any scaling operation.
- $ratio$: is an integer value that indicates the scale ratio of a scaled sample, i.e. the number of original samples represented by that scaled sample. This parameter is common to all the elements in a sequence of scaled samples. The value to be used when *Scaling* is absent is 1.
- $numOfElements$: is an integer value indicating the number of consecutive elements in a sequence of scaled samples that share the same scale ratio. If *Scaling* is absent, it is equal to the value of $totalNumOfSamples$.

Two distinct types of scalable series are defined for representing series of scalars and series of vectors in the MPEG-7 LLD framework. Both types inherit from the scalable series description. The following sections present them in detail.

2.2.1 Series of Scalars

The MPEG-7 standard contains a $SeriesOfScalar$ descriptor to represent a series of scalar values, at full resolution or scaled. This can be used with any temporal series of scalar LLDs.

The attributes of a $SeriesOfScalar$ description are:

- *Raw*: may contain the original series of scalars when no scaling operation is applied. It is only used if the *Scaling* flag is absent to store the entire series at full resolution.
- *Weight*: is an optional series of weights. If this attribute is present, each weight corresponds to a sample in the original series. These parameters can be used to control scaling.
- *Min*, *Max* and *Mean*: are three real-valued vectors in which each dimension characterizes a sample in the scaled series. For a given scaled sample, a *Min*, *Max* and *Mean* coefficient is extracted from the corresponding group of samples in the original series. The coefficient in *Min* is the minimum original sample value, the coefficient in *Max* is the maximum original sample value and the coefficient in *Mean* is the mean sample value. The original samples are averaged by arithmetic mean, taking the sample weights into account if the *Weight* attribute is present (see formulae below). These attributes are absent if the *Raw* element is present.
- *Variance*: is a real-valued vector. Each element corresponds to a scaled sample. It is the variance computed within the corresponding group of original samples. This computation may take the sample weights into account if the *Weight* attribute is present (see formulae below). This attribute is absent if the *Raw* element is present.
- *Random*: is a vector resulting from the selection of one sample at random within each group of original samples used for scaling. This attribute is absent if the *Raw* element is present.
- *First*: is a vector resulting from the selection of the first sample in each group of original samples used for scaling. This attribute is absent if the *Raw* element is present.
- *Last*: is a vector resulting from the selection of the last sample in each group of original samples used for scaling. This attribute is absent if the *Raw* element is present.

These different attributes allow us to summarize any series of scalar features [9].

Such a description allows *scalability*, in the sense that a scaled series can be derived indifferently from an original series (*scaling* operation) or from a previously scaled $SeriesOfScalar$ (*rescaling* operation).

Initially, a series of scalar LLD features is stored in the Raw vector. Each element $Raw(l)$ $(0 \leq l \leq L-1)$ contains the value of the scalar feature extracted from the lth frame of the signal. Optionally, the *Weight* series may contain the weights $W(l)$ associated to each $Raw(l)$ feature.

When a scaling operation is performed, a new $SeriesOfScalar$ is generated by grouping the original samples and calculating the abovementioned attributes. The *Raw* attribute is absent in the scaled series descriptor.

Let us assume that the ith scaled sample stands for the samples $Raw(l)$ contained between $l = lLo(i)$ and $l = lHi(i)$with:

$$lHi(i) = lLo(i) + ratio - 1 \tag{2.1}$$

where *ratio* is the scale ratio of the ith scaled sample (i.e. the number of original samples it stands for). The corresponding *Min* and *Max* values are then defined as:

$$Min(i) = min_{l=lL_0(i)}^{lHi(i)} Raw(l) \tag{2.2}$$

$$Max(i) = max_{l=lL_0(i)}^{lH_i(i)} Raw(l) \tag{2.3}$$

The *Mean* value is given by (2.4) :

$$Mean(i) = \frac{1}{ratio} \sum_{l=lL_0(i)}^{lH_i(i)} Raw(l) \tag{2.4}$$

if no sample weights $W(l)$ are specified in *Weight*. If weights are present, the *Mean* value is computed as (2.5):

$$Mean(i) = \frac{\sum_{l=lL_0(i)}^{lH_i(i)} W(l)Raw(l)}{\sum_{l=lL_0(i)}^{lH_i(i)} W(l)} \quad (2.5)$$

In the same way, there are two computational methods for the *Variance* depending on whether the original sample weights are absent:

$$Variance(i) = \frac{1}{ratio}\sum_{l=lL_0(i)}^{lH_i(i)}[Raw(l) - Mean(i)]^2 \quad (2.6)$$

or present:

$$Variance(i) = \frac{\sum_{l=lL_0(i)}^{lH_i(i)} W(l)[Raw(l) - Mean(i)]^2}{\sum_{l=lL_0(i)}^{lH_i(i)} W(l)} \quad (2.7)$$

Finally, the weights $W(i)$ of the new scaled samples are computed, if necessary, as (2.8):

$$W(i) = \frac{1}{ratio}\sum_{l=lL_0(i)}^{lH_i(i)} W(l) \quad (2.8)$$

2.2.2 Series of Vectors

Some LLDs do not consist of single scalar values, but of multi-dimensional vectors. To store these LLDs as scalable series, the MPEG-7 standard contains a $SeriesOfVector$ descriptor to represent temporal series of feature vectors. As before, a series can be stored at the full original resolution or scaled.

The $SeriesOfVector$ in order to summarize a series of vectors through scaling and/or rescaling operations has the same attributes of the $SeriesOfScalar$ adapted to the vectorial case and adds to them other attributes

Initially, a series of vector LLD features is stored in the Raw attribute. Each element $Raw(l)$ $(0 \leq l \leq L-1)$ contains the vector extracted from the l th frame of the signal. Optionally, the *Weight* series may contain the weights $W(l)$ associated to each vector.

When a scaling operation is performed, a new $SeriesOfVector$ is generated.

2.2.3 Binary Series

The standard defines a binary form of the aforementioned $SeriesOfScalar$ and $SeriesOfVector$ descriptors: namely, the $SeriesOfScalarBinary$ and $SeriesOfVectorBinary$ descriptors. These descriptors are used to instantiate series of scalars or vectors with a uniform power-of-2 *ratio*. The goal is to ease the comparison of series with different scaling ratios, as the decimation required for the comparison between two binary series is also a power of 2.

2.3 Audio Waveform Descriptor

A simple way to get a compact description of the shape of an audio signal $s(n)$ is to consider its minimum and maximum samples within successive non-overlapping frames (i.e. $Lw = hopSize$).

For each frame, two values are stored:

- $minRange$: Lower limit of audio signal amplitude;
- $maxRange$: Upper limit of audio signal amplitude.

The audio waveform (AWF) descriptor consists of the resulting temporal series of these ($minRange$, *maxRange*) pairs. The temporal resolution of the AWF is given by the $hopSize$ parameter. If desired, the raw signal can be stored in an AWF descriptor by setting $hopSize$ to the sampling period $1/Fs$ of $s(n)$.

The AWF provides an estimate of the signal envelope in the time domain. It also allows economical and straightforward storage, display or comparison techniques of waveforms. For example, the waveform may be displayed using a small set of values that represent extreme (min and max) of frames of samples. They may also be used for fast comparison between waveforms[9].

In order to compute this descriptor I have written a matlab code that reads the wav file and set the $scalingRatio$, that is the number of samples in the non-overlapping frames, so as to have a frame duration of 10 ms (that is the default value of the temporal resolution) and then with these parameters, the code calls the matlab function of the MPEG-7 Experimental Model [15]

$AudioWaveformD$ that uses the functions $h\_Max\_SeriesOfScalar$ and $h\_Min\_SeriesOfScalar$ which I have already described in the paragraph 2.2.1.

Figure 2.4 gives graphical representations of the AWF descriptor extracted from 7 seconds anchorman speech and 7 seconds of theme music. We can see that the MPEG-7 AWF provides a good approximation of the shape of the original waveform.

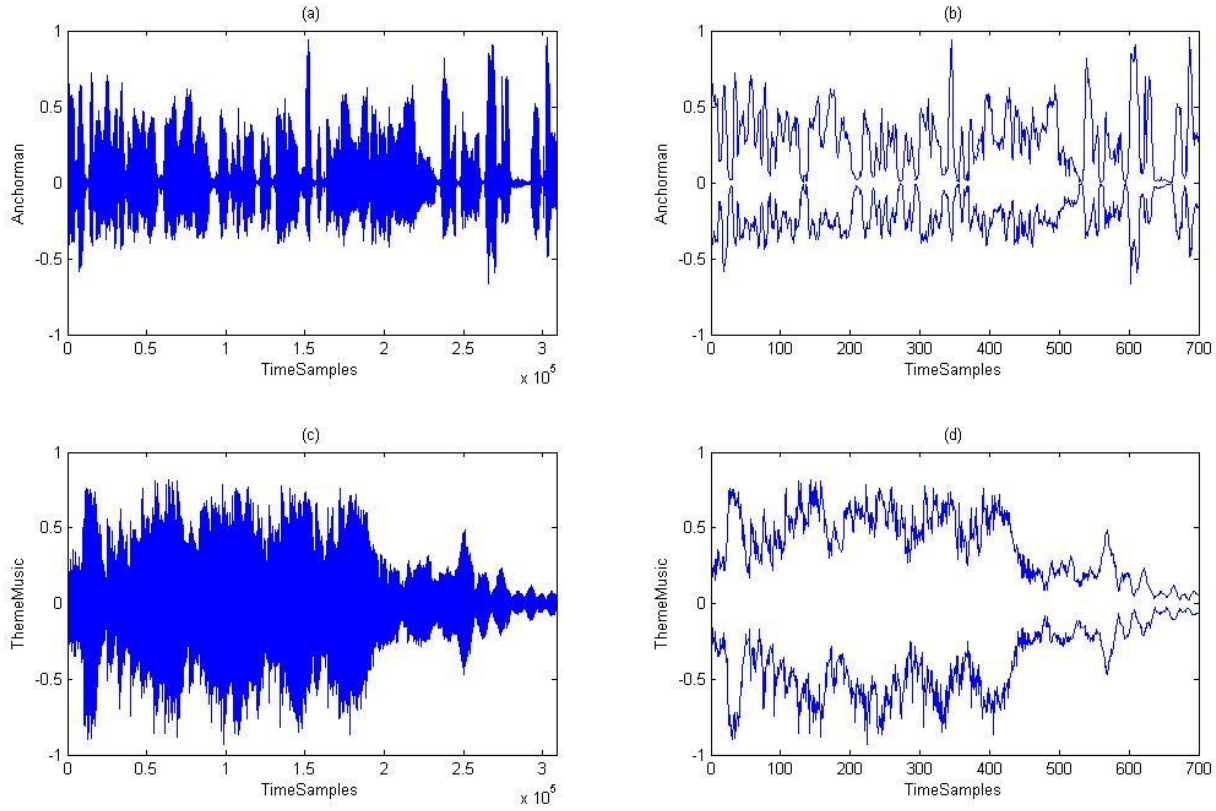

Figure 2.3:(a) Anchorman speech signal of 7 s; (b) AWF of the Anchorman speech; (c) Theme music signal of 7 s; (d) AWF of the Theme music.

2.4 Audio Power

The audio power (AP) LLD describes the temporally smoothed instantaneous power of the audio signal. The AP coefficients are the average square of waveform values $s(n)$ within successive non-overlapping frames ($Lw = hopSize$) [9]. The AP coefficient of the lth frame of the signal is thus:

$$AP(l) = \frac{1}{N_{hop}} \sum_{n=0}^{N_{hop}-1} \left| s\left(n + lN_{hop}\right) \right|^2 \quad (0 \le l \le L-1) \tag{2.9}$$

where L is the total number of time frames. The AP allows us to measure the evolution of the amplitude of the signal as a function of time. In conjunction with other basic spectral descriptors (described below), it provides a quick representation of the spectrogram of a signal.

2.4.1 Extraction

In order to extract this descriptor I have used the matlab code $AudioPowerD$ which is in the Experimental Model [15]. This code doesn't work good because it at first segments the input signal in frame of 1024 samples, and then for each frame makes the square of each element and pass it to the function $h\_Mean\_SeriesOfScalar$ which groups the samples corresponding to 10 ms and for each group compute the mean value as described in paragraph. The problem is that it doesn't store it and for each frame overwrite the content of the result. In order to solve this problem and to have a temporal resolution of 10 ms I have modified the code in this way: At first the code makes the square of each element of the entire signal and then pass it to the function $h\_Mean\_SeriesOfScalar$ which groups the samples corresponding to 10 ms and for each group compute the mean value.

This descriptor can be used to discriminate speech and music segments in fact as we can see from the figure 2.4 the speech signal is composed of altering voiced and unvoiced sounds and silence periods. These unvoiced and silence periods carry less energy than the voiced sounds. Thus, the Energy values for speech will have a large variation. On the contrary the energy of music due to the pitched nature of music is more constant and larger than speech.

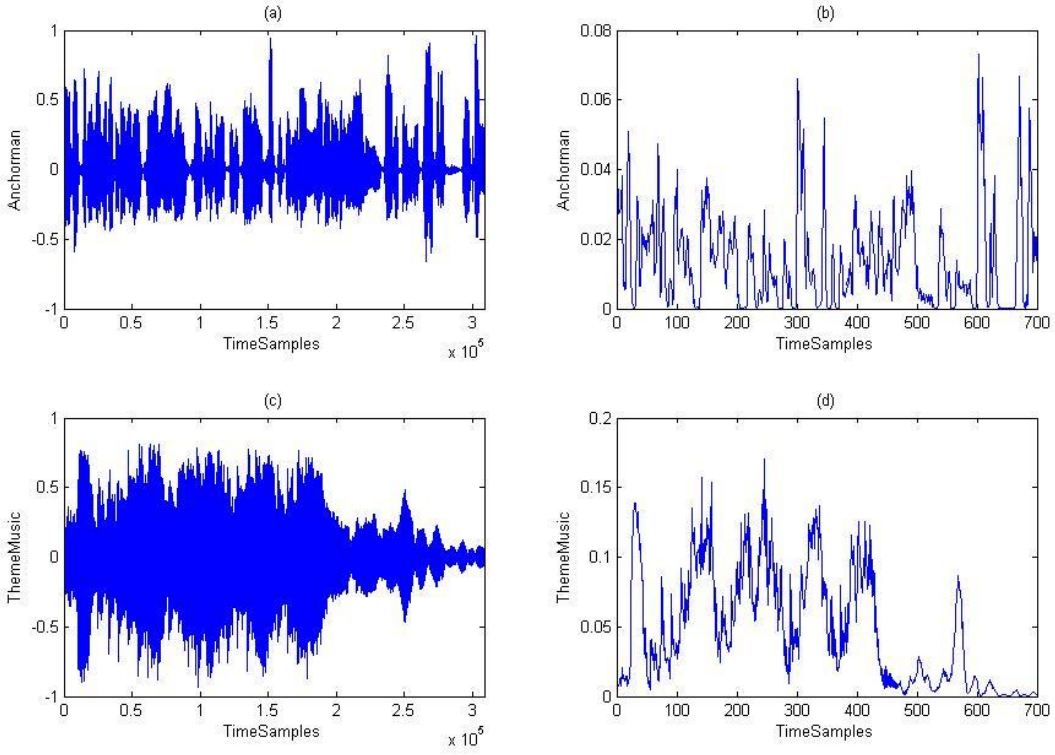

Figure 2.4: (a) Anchorman speech signal of 7 s; (b) AP of the Anchorman speech; (c) Thememusic signal of 7 s; (d) AP of the Thememusic.

2.5 Basic spectral descriptors

The four basic spectral LLDs provide time series of logarithmic frequency descriptions of the short term audio power spectrum. The use of logarithmic frequency scales is supposed to approximate the response of the human ear. All these descriptors are based on the estimation of short-term power spectra within overlapping time frames.

2.5.1 Audio Spectrum Envelope

The audio spectrum envelope (ASE) is a log-frequency power spectrum that can be used to generate a reduced spectrogram of the original audio signal. It is obtained by summing the energy of the original power spectrum within a series of frequency bands. The bands are logarithmically distributed (base 2 logarithms) between two frequency edges $loEdge$ (lower edge) and $hiEdge$ (higher edge) [16]. The spectral resolution r of the frequency bands within the [$loEdge$,

$hiEdge$] interval can be chosen from eight possible values, ranging from 1/16 of an octave to 8 octaves:

$$\mathrm{r} = 2^{\mathrm{j}} \text{ octaves}, \quad -4 \leq \mathrm{j} \leq +3 \tag{2.10}$$

Both $loEdge$ and $hiEdge$ must be related to 1 kHz in the following way:

$$Edge = 2^{rn} \times 1\ kHz \tag{2.11}$$

where r is the resolution in octaves and n is an integer value. The default value of $hiEdge$ is 16 kHz, which corresponds to the upper limit of hearing. The default value of $loEdge$ is 62.5 Hz so that the default $[loEdge, hiEdge]$ range corresponds to an 8-octave interval, logarithmically centered at a frequency of 1 kHz.
Within the default $[loEdge, hiEdge]$ range, the number of logarithmic bands that corresponds to r is $B_{in} = 8/r$. The low ($loFb$) and high ($hiFb$) frequency edges of each band are given by:

$$loF_b = loEdge \times 2^{(b-1)r} \tag{2.12}$$

$$hiF_b = loEdge \times 2^{br} \tag{2.13}$$

where $1 \leq b \leq B_{in}$

The sum of power coefficients in band b $[loFb, hiFb]$ gives the ASE coefficient for this frequency range. The coefficient for the band b is:

$$ASE = \sum_{k=loK_b}^{hiK_b} P(k) \quad (1 \leq b \leq B_{in}) \tag{2.14}$$

where $P(k)$ are the power spectrum coefficients and loK_b (resp. hiK_b) is the integer frequency bin corresponding to the lower edge of the band loF_b (the higher edge of the band hiF_b) obtained considering that in the FFT spectrum, the discrete frequencies corresponding to bin indexes k are:

$$f(k) = k\Delta F \ \ (0 \le k \le N_{FT}/2) \tag{2.15}$$

Where $\Delta F = F_s/N_{FT}$ is the frequency interval between two successive FFT bins. Inverting the preceding equation, we can map any frequency in the range $[0, Fs/2]$ to a discrete bin in $\{0, 1, \dots, N_{FT}/2\}$:

$$k = round(f/\Delta F) \ \ (0 \le f \le F_s/2) \tag{2.16}$$

2.5.1.1 Extraction

To extract the Audio Spectrum Envelope I have used the $AudioSpectrumEnvelopeD$ matlab code that is in the Experimental Model [15]. This code involves a sliding window FFT analysis, with a re-sampling to logarithmic spaced bands.

At first the code determines the required hop length h, corresponding to the hopsize. If the sampling rate is f_s, then

$$h \ = \ f_s \times hopSize \tag{2.17}$$

If $f_s \times hopSize$ is not a whole number of samples then generates a vector h such that

$$mean(h) \ = \ f_s \times hopSize \tag{2.18}$$

and interleaves minor $hopSize$ with major (e.g if 10 10 10 10 10 10 11 11 are the hops then the pattern should be 10 10 10 11 10 10 11). By cycling through the vector of hop lengths the analysis will not stray over time, but will give minor jitter from the defined $hopSize$. This enables reasonable comparison of data sampled at differing rates.

The analysis window length l_w. has been chosen to have a default value of 3 $hopSizes$, 30ms. After computes l_w the code determines the FFT size, N_{FT}. N_{FT} is the next-larger power-of-two number of samples from l_w.

Then performs a STFT using a Hamming window of length l_w, with the shifts which are in the vector h and computing the FFT with N_{FT} point, setting the out-of-window samples to 0.

After that, the code, to obtain the power spectrum coefficients $P(k)$, computes the square magnitude of the FFT coefficients $|S_l(k)|^2$.

Since the signal spectrum is symmetric around the Nyquist frequency $Fs/2$, it considers only the first half of the power spectrum ($0 \leq k \leq N_{FT}\ /2$) without losing any information and normalize it to respect the Parseval's Theorem in the following way

$$P_l(k) = \frac{1}{N_{FT}\,E_w}|S_l(k)|^2 \qquad for\ k = 0\ and\ k = \frac{N_{FT}}{2} \tag{2.19}$$

$$P_l(k) = 2\frac{1}{N_{FT}\,E_w}|S_l(k)|^2 \qquad for\ 0 < k < \frac{N_{FT}}{2} \tag{2.20}$$

In the end the code resamples the power spectrum coefficients to a logarithmic scale.

The repartition of the power spectrum coefficients $P(k)$ among the different frequency bands can be a problem, particularly for the narrower low frequency bands when the resolution r is high. It is reasonable to assume that a power spectrum coefficient whose distance to a band edge is less than half the FFT resolution (i.e. less than $\Delta F/2$) contributes to the ASE coefficients of both neighboring bands. How such a coefficient should be shared by the two bands is not specified by the standard. The method which is used in the code which I have used is depicted in the figure 2.5 [16].

The B_{in} *within-band* band power coefficients are completed by two additional values: the powers of the spectrum between 0 Hz and $loEdge$ and between $hiEdge$ and the Nyquist frequency $Fs/2$.These two values represent the *out-of-band* energy.

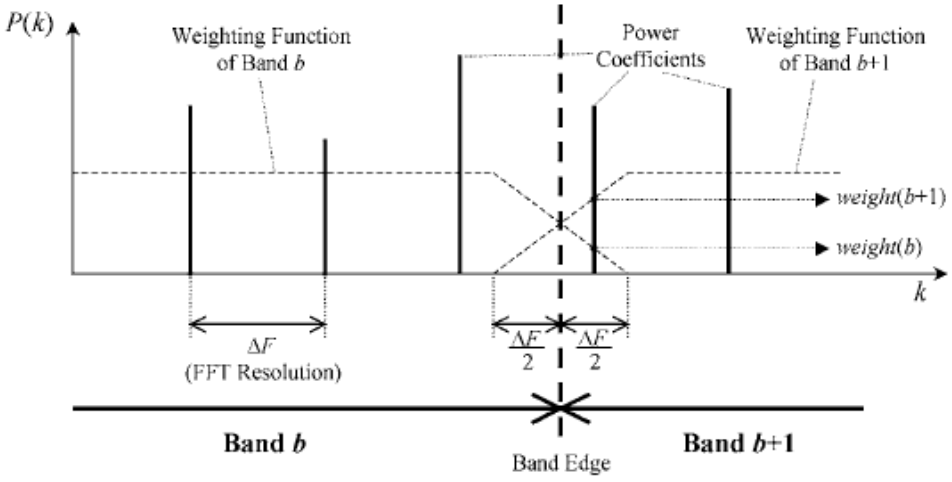

Figure 2.5 Method for weighting the contribution of a power coefficient shared by two bands

In the following, $B = B_{in} + 2$ will describe the total number of coefficients forming the ASE descriptor. With $loEdge$ and $hiEdge$ default values, the dimension of an ASE can be chosen between $B = 3$ ($B_{in} = 1$) with the minimal resolution of 8 octaves and $B = 130$ ($B_{in} = 128$) with the maximal resolution of $1/16$ octave.

The extraction of an ASE vector from a power spectrum descript above is depicted in figure 2.6 with, as an example, the $loEdge$ and $hiEdge$ default values and a 1-octave resolution. The ASE vectors comprise 10 coefficients: 8 *within-band* coefficients plus 2 *out-of-band* coefficients.

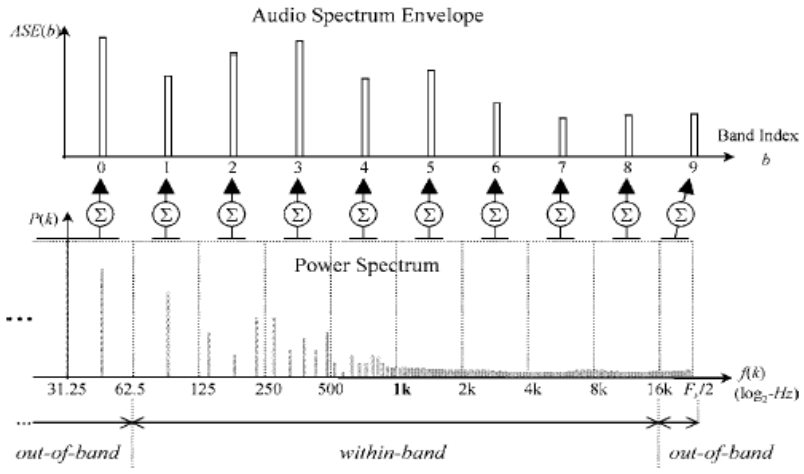

Figure 2.6: Extraction of ASE from a power spectrum with a single-octave resolution

This descriptor has useful scaling properties: the power spectrum over an interval is equal to the sum of power spectra over subintervals.

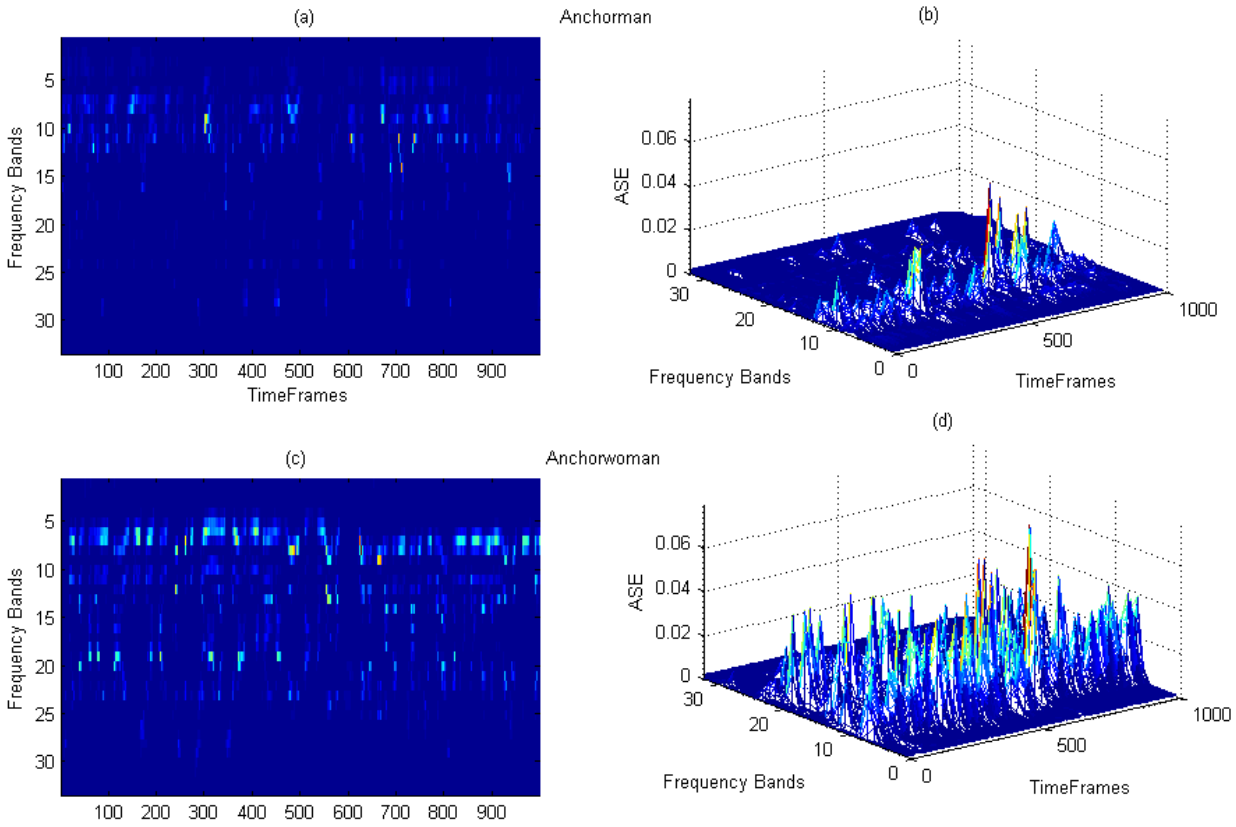

Figure 2.7: (a) ASE of 10 s of anchorman speech; (b) ASE of 10 s of anchorwoman speech

In figure 2.7 we can compare the ASE of 10 s anchorman speech with the ASE of 10 s anchorwoman speech. Each ASE vector is extracted from 34 frequency bands and consists of 32 within-band coefficients between $loEdge = 62.5Hz$ and $hiEdge = 16\ kHz$ (i.e. a $1/4$-octave resolution) and two out-of-band coefficients. ASE vectors are extracted every 10 ms from 30 ms frames and represented vertically at the corresponding frame indexes. And we can recognize the gender of the speaker only by view the plot in fact compared with the anchorwoman the anchorman produces more energy at the lower frequencies and less at the higher frequencies.

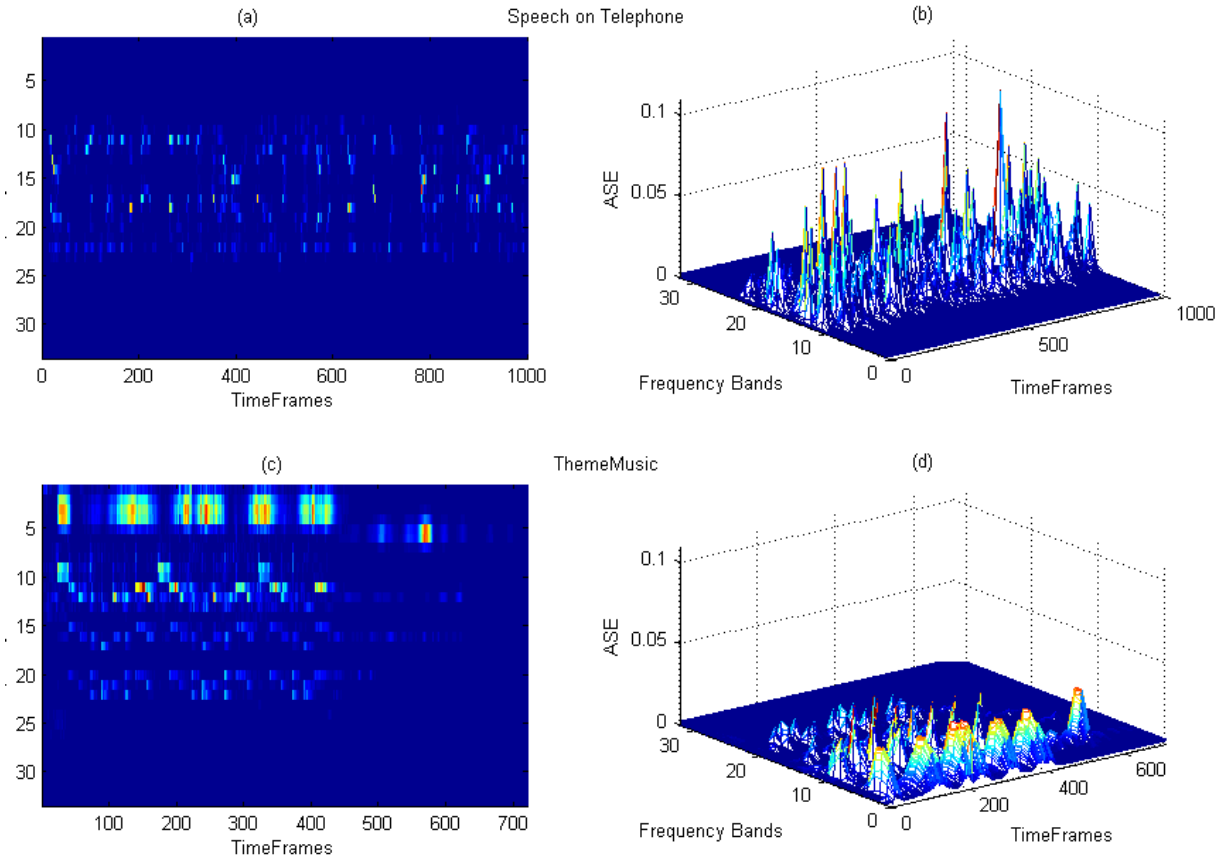

Figure 2.8: (a) ASE of 10 s of speech on the telephone; (b) ASE of 7 s Thememusic.

Instead in figure 2.8 I have reported at first the ASE of 10 s of speech on telephone in which the effects of the non ideal telephone line are evident, and the ASE of 10 s of the thememusic of the GR where we can see that concentrates more energy at lower frequency than the speech signal.

2.5.2 Audio Spectrum Centroid

To be coherent with other descriptors, in particular ASE the spectrum centroid is defined as the center-of-gravity of a log-frequency power spectrum. This definition is adjusted in the extraction to take into account the fact that a non-zero DC component creates a singularity, and eventual very-low frequency components (possibly spurious) have a disproportionate weight [9].

2.5.2.1 Extraction

In order to extract the Audio Spectrum Centroid I have used the $AudioSpectrumCentroidD$ matlab code that is in the MPEG-7 Experimental Model [15]. This code calculate the power spectrum coefficients $P_l(k)$, as described in the previous paragraph. Then power spectrum coefficients below 62.5 Hz are replaced by a single coefficient, with power equal to their sum and a nominal frequency of 31.25 Hz that in the discrete frequency bin scale corresponds to the index:

$$K_{low} = floor\left(\frac{62.5}{\Delta F}\right) \tag{2.21}$$

This results in a new power spectrum $P'(k')$ whose relation to the original spectrum $P(k)$ is given by:

$$P'(k') = \begin{cases} \sum_{k=0}^{K_{low}} P(k), & k' = 0 \\ P(k' + K_{low}), & 1 \le k' \le \frac{N_{FT}}{2} - K_{low} \end{cases} \tag{2.22}$$

The frequencies $f'(k')$ corresponding to the new bins k' are given by:

$$f'(k') = \begin{cases} 31.25, & k' = 0 \\ f(k' + K_{low}), & 1 \le k' \le \frac{N_{FT}}{2} - K_{low} \end{cases} \tag{2.23}$$

where $f(k)$ is defined as in equation 2.7. The nominal frequency of the low-frequency coefficient is chosen at the middle of the low-frequency band: $f'(0) = 31.5\ Hz$.

Finally, for a given frame, the ASC is defined from the modified power coefficients $P'(k')$ and their corresponding frequencies $f'(k')$ as:

$$ASC = \frac{\sum_{k'=0}^{(N_{FT}/2)-k_{low}} \log_2\left(\frac{f'(k')}{1000}\right) P'(k')}{\sum_{k'=0}^{(N_{FT}/2)-k_{low}} P'(k')} \tag{2.24}$$

Each frequency $f'(k')$of the modified power spectrum is weighted by the corresponding power coefficient $P'(k')$.

The ASC measure gives information on the shape of the power spectrum. It indicates whether a power spectrum is dominated by low or high frequencies and can be regarded as an approximation of the perceptual sharpness of the signal. In fact the log-frequency scaling approximates the perception of frequencies in the human hearing system [15].

Figure 2.9 depicts the temporal series of ASC values of 7 s of anchorman speech, 7 s of speech on telephone and of 7 s of theme music. In this example, the spectrum is dominated by lower frequencies.

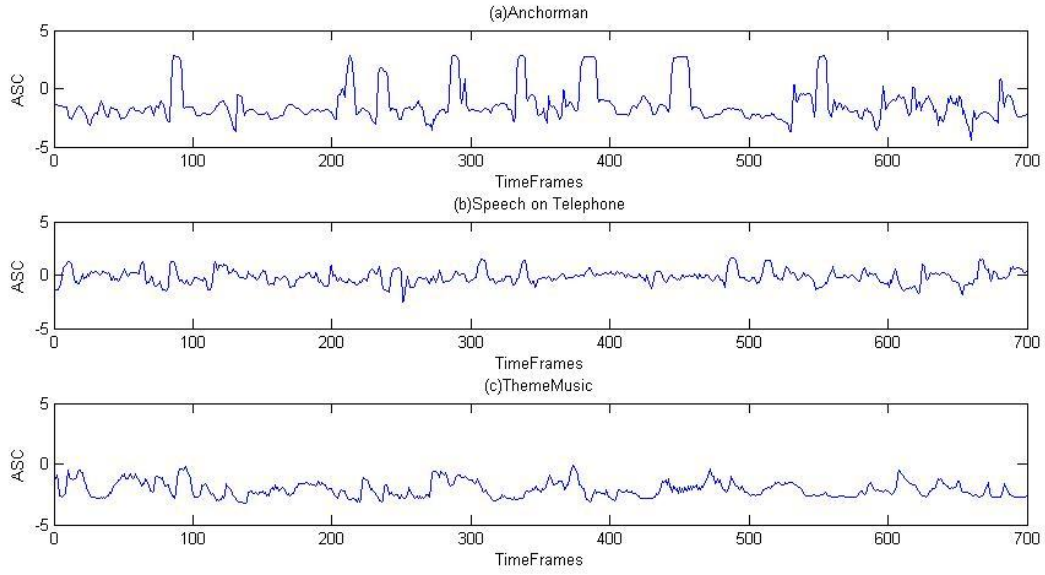

Figure 2.9: (a) ASC of 7 s of anchorman speech; (b) ASC of 7 of speech on the telephone; (c) ASC of 7 s of theme music

We can note that the observations that we have made when we have analyzed the ASE of the same segments are still valid. In fact the ASC values of the speech on the telephone, due to the telephonic line effects remain around 0 which means, according to equation 2.24, that the corresponding frequency centroids remain around 1 kHz. And the ASC values of the theme music remain below 0 and so the corresponding frequency centroids remain below 1 kHz, according to

the fact that the spectrum of the music segments is more predominant by lower frequency than the speech segments.

2.5.3 Audio Spectrum Spread

The audio spectrum spread (ASS) is another simple measure of the spectral shape. The *spectral spread*, also called *instantaneous bandwidth*, can be defined in several different ways. In MPEG-7, it is defined as the second central moment of the log-frequency spectrum [9]. For a given signal frame, the ASS feature is extracted by taking the root-mean-square (RMS) deviation of the spectrum from its centroid ASC:

$$ASS = \sqrt{\frac{\sum_{k'=0}^{(N_{FT}/2)-k_{low}} \left[\log_2\left(\frac{f'(k')}{1000}\right) - ASC\right]^2 P'(k')}{\sum_{k'=0}^{(N_{FT}/2)-k_{low}} P'(k')}} \tag{2.25}$$

where the modified power spectrum coefficients $P'(k')$ and the corresponding frequencies $f'(k')$ are calculated in the same way as for the ASC descriptor.

The ASS gives indications about how the spectrum is distributed around its centroid. A low ASS value means that the spectrum may be concentrated around the centroid, whereas a high value reflects a distribution of power across a wider range of frequencies. It is designed to help differentiate *noise-like* and *tonal* sounds [16].

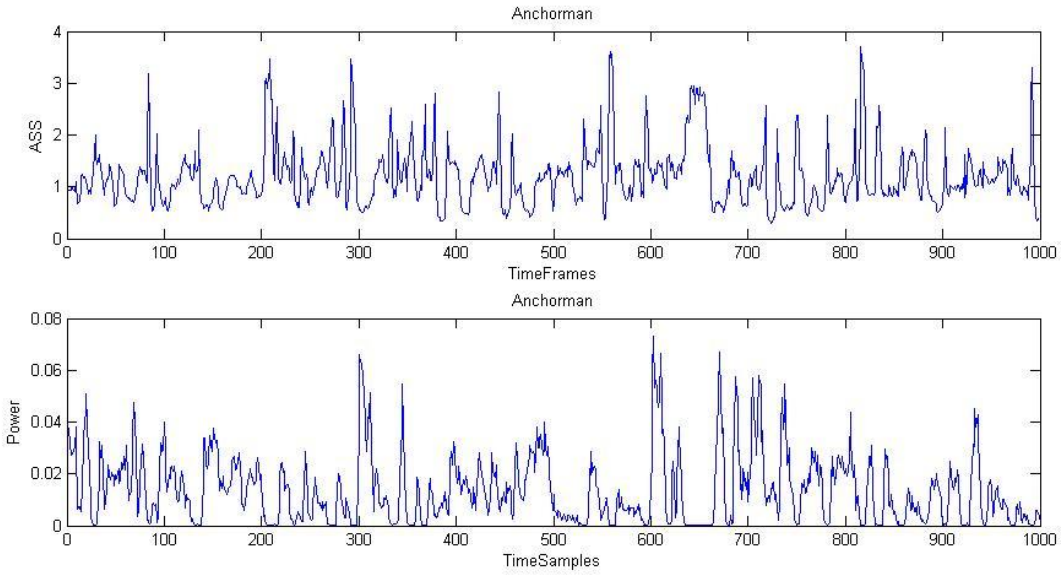

Figure 2.10: (a) ASS of 10 s of anchorman speech; (b) AP of the same segment.

In figure 2.10 (a) I have reported the temporal series of ASS values of 10 s of anchorman speech and below in the figure 2.10 (b) the AP of the same audio segments. Comparing the two graphs, we can note that only when we have piece of silence or noise-like the ASS assumes high values otherwise the spread remains rather low.

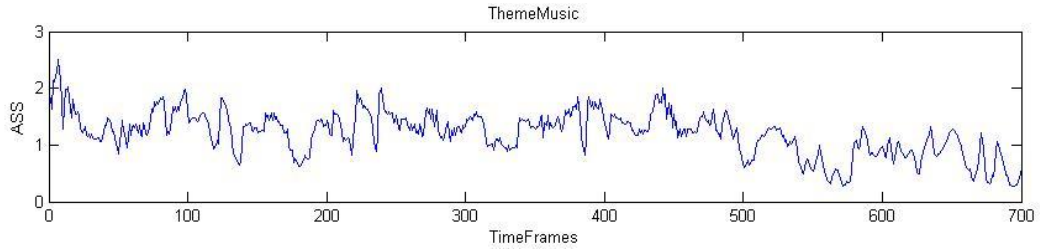

Figure 2.11: ASS of 7 s of thememusic

Figure 2.11 instead depicts the temporal series of ASS values of 7 s of thememusic and we can observe that respects to the ASS of speech signal this ASS has minus variations, because it doesn't have piece of silence or noise-like as the speech segments.

2.5.4 Audio Spectrum Flatness

The audio spectrum flatness (ASF) reflects the flatness properties of the power spectrum. More precisely, for a given signal frame, it consists of a series of values, each one expressing the deviation of the signal's power spectrum from a flat shape inside a predefined frequency band [9]. As such, it is a measure of how similar an audio signal is to white noise, or, vice versa, how correlated a signal is.

2.5.4.1 Extraction

In order to extract the Audio Spectrum Flatness I have used the $AudioSpectrumFlatnessD$ matlab code that is in the MPEG-7 Experimental Model [15].

At first, in the code, is performed a spectral analysis (windowing, DFT) of the input signal using the same procedure and parameters specified for the extraction of the $AudioSpectrumEnvelopeType$, but with the window length, l_w, corresponding to hop size (i.e. no overlap between subsequent calculations), which is recommended to be 30 ms in this case.

Within a $[loEdge, hiEdge]$ range, the spectrum is then divided into 1/4-octave-spaced log-frequency bands. These parameters must be distinguished from the $loEdge$ and $hiEdge$ edges used in the definition of the ASE descriptor. Here, the values of $loEdge$ and $hiEdge$ must be chosen so that the intervals separating them from 1 kHz are integer multipliers of a 1/4 octave. We thus have:

$$loEdge = 2^{\frac{1}{4}n} \times 1kHz \tag{2.26}$$

$$hiEdge = 2^{\frac{1}{4}B} \times loEdge \tag{2.27}$$

where n and B are integer parameters with the following meanings:

- The value of n determines the lower band edge. The minimum value for $loEdge$ is recommended to be 250 Hz (i.e. $n = -8$).
- B is the desired number of frequency bands. After $loEdge$ has been set, the value of B determines the higher band edge. The value of $hiEdge$ should not exceed a frequency limit beyond which no flatness features can be properly extracted. The most obvious

limitation to $hiEdge$ is the Nyquist frequency. Another limitation could be the bandwidth of the original signal. The choice of parameter B must be made accordingly within these limitations. For these reason in the code there is a control on the value of the $hiEdge$, if upper than the Nyquist frequency , then this value is set to the Niquist frequency.

The resulting frequency bands are proportional to those used in the definition of the ASE, thus ensuring compatibility among the different basic spectral descriptors. However, defining frequency bands with no overlap could make the calculation of ASF features too sensitive to slight variations in sampling frequency. Therefore, the nominal edge frequencies of equation 2.27 are modified so that the B frequency bands slightly overlap each other. Each band is thus made 10% larger in the following manner:

$$loF_b = 0.95 \times loEdge \times 2^{\frac{1}{4}(b-1)}$$
$$hiF_b = 1.05 \times loEdge \times 2^{\frac{1}{4}b} \qquad (1 \leq b \leq B) \qquad (2.28)$$

with loF_b and hiF_b being the lower and upper limits of band b. We denote as loK_b and hiK_b the corresponding bins in the power spectrum, obtained from equation (2.8).

Furthermore, in order to reduce computational costs and to adjust the frequency resolution of the spectrum to the log-frequency bands, the MPEG-7 standard specifies a method for grouping the power spectrum coefficients $P(k)$ in bands above the edge frequency of 1 kHz. The grouping is defined as follows:

- For all bands between 1 kHz and 2 kHz power spectrum coefficients $P(k)$ are grouped by pairs. Two successive coefficients $P(k)$ and $P(k+1)$ are replaced by a single average coefficient $(P(k) + P(k+1))/2$.
- This grouping procedure is generalized to the following intervals of 1 octave as follows. Within all bands between $2^n\ kHz$ and $2^{n+1} kHz$ (where n is an integer and $n \geq 1$), each group of 2^n successive power coefficients is replaced by a single coefficient equal to their arithmetic mean. Figure 2.10 illustrates the coefficient grouping procedure within two consecutive bands b (between $f = 2^{\frac{3}{4}}\ kHz \approx 1681.8Hz$ and $f = 2\ kHz$) and $b+1$ (between $f = 2\ kHz$ and $f = 2^{\frac{5}{4}}\ kHz \approx 2378\_4\ Hz$). As specified in equation 2.21 these nominal

edge frequencies are actually modified to introduce a 10% overlap represented on the schema[16].

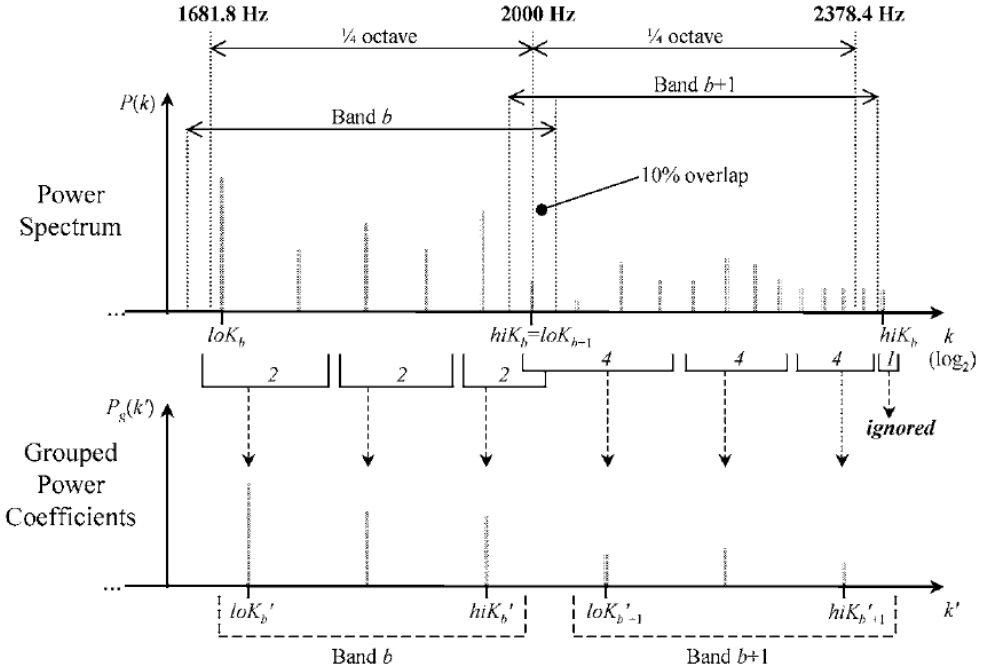

Figure 2.12: Power coefficient grouping within two consecutive bands around 2 kHz

- At the end of each band, the last group of coefficients may not contain the required number of values. If at least 50% of the required coefficients are available (i.e. 2^n coefficients for bands between 2^n kHz and 2^{n+1} kHz), the group is completed by using the appropriate number of coefficients at the beginning of the next band. Otherwise, no average coefficient is yielded; the power coefficients contained in the last group are simply ignored. In the example of figure 2.10, the last group of band $b+1$ only contain one coefficient, which is ignored in the calculation of the three grouped power coefficients finally associated to $b+1$.

This grouping procedure results in a new set of power coefficients $P_g(k')$. We call loK'_b and hiK'_b the new band edge indexes of frequency bands b in the modified power spectrum (see figure 2.10).

For each band b, a spectral flatness coefficient is then estimated as the ratio between the geometric mean and the arithmetic mean of the spectral power coefficients within this band:

$$ASF = \frac{\sqrt[hiK'_b - loK'_b + 1]{\prod_{k'=loK'_b}^{hiK'_b} P_g(k')}}{\frac{1}{hiK'_b - loK'_b + 1} \sum_{k'=loK'_b}^{hiK'_b} P_g(k')} \qquad (1 \le b \le B) \tag{2.29}$$

If no audio signal is present (i.e. the mean power is zero), a flatness measure value of 1 is returned.

For all bands under the edge of 1 kHz, the power coefficients are averaged in the normal way. In that case, for each band b, we have $P_g(k') = P(k')$ between $k' = loK'_b = loK_b$ and $k' = hiK'_b = hiK_b$. For all bands above 1 kHz, for which a power coefficient grouping was required, only the reduced number of grouped coefficients is taken into account in the calculation of the geometric and arithmetic means.

A flat spectrum shape corresponds to a noise or an impulse signal. Hence, high ASF coefficients are expected to reflect noisiness. On the contrary, low values may indicate a harmonic structure of the spectrum. From a psycho-acoustical point of view, a large deviation from a flat shape (i.e. a low spectral flatness measure) generally characterizes the *tonal* sounds.

In figure 2.13 I have reported the temporal series of the ASE and ASF vectors of 10 seconds of anchorman speech and 7 seconds of theme music. Each ASF vector is extracted from 24 frequency bands within a 6-octave frequency interval, between $loEdge = 250Hz$ and $hiEdge = 16\ kHz$ (chosen to be smaller than the 22.05 kHz Nyquist frequency). A lighter shade indicates a higher spectral flatness value, meaning that the tonal component is less present in the corresponding bands.

Comparing the two descriptors we can note that where the ASE has high values the ASF has low value in fact in these point the spectrum is far to be flat.

The spectral flatness coefficients may be used as a feature vector for robust matching between pairs of audio signals. It is also possible to reduce the spectral flatness features to a single scalar by computing the mean value across the frequency band coefficients *ASF(b)* for each frame. The resulting feature measures the overall flatness of a frame and can be used by an audio classifier [17].

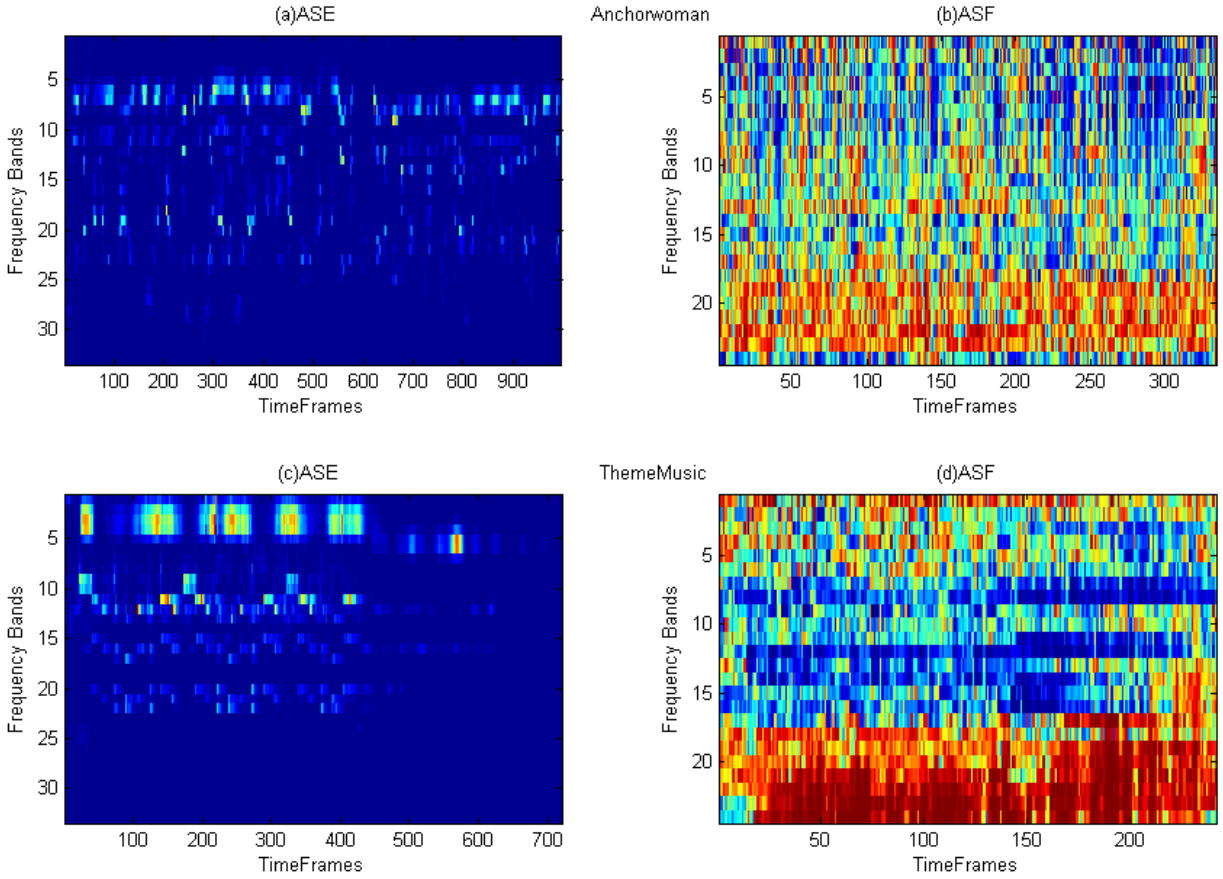

Figure 2.13: (a-b) ASE and ASF of 10 s of anchorwoman speech; (b-c)ASE and ASF of 7 s of thememusic

2.6 Basic signal parameters

The above-mentioned basic spectral LLDs give a smoothed representation of power spectra. They cannot reflect the detailed harmonic structure of periodic sounds because of a lack of frequency resolution. The following descriptors provide some complementary information, by describing the degree of harmonicity of audio signals.

2.6.1 Audio Harmonicity

The audio harmonicity (AH) descriptor provides two measures of the harmonic properties of a spectrum:

- the *harmonic ratio*: the ratio of harmonic power to total power.

- the $upper\ limit\ of\ harmonicity$: the frequency beyond which the spectrum cannot be considered harmonic.

They both rely on a standardized fundamental frequency estimation method, based on the local normalized autocorrelation function of the signal. This approach, widely used for local pitch estimation, is independent of the extraction of the audio fundamental frequency descriptor presented below [9].

2.6.1.1 Harmonic Ratio

The harmonic ratio (HR) is a measure of the proportion of harmonic components in the power spectrum. An HR coefficient is computed for each N_w sample frame of the original signal $s(n)$, with a hop of N_{hop} samples between successive frames. The extraction of an HR frame feature is standardized as follows. For a given frame index l, the normalized autocorrelation function of the signal is first estimated as:

$$\Gamma_l(\mathrm{m}) = \frac{\sum_{\mathrm{n}=0}^{\mathrm{N_w}-1} \mathrm{s_l(n)s_l(n-m)}}{\sqrt{\sum_{\mathrm{n}=0}^{\mathrm{N_w}-1} \mathrm{s_l(n)^2} \sum_{\mathrm{n}=0}^{\mathrm{N_w}-1} \mathrm{s_l(n-m)^2}}} \qquad (l \le m \le M; 0 \le l \le L-1) \tag{2.30}$$

where $s_l(n)$ is defined as $s(lN_{hop}+n)$, m is the lag index of the autocorrelation and L is the total number of frames in $s(n)$. In the definition of equation 2.30 autocorrelation values are computed at lags ranging from $m = 1$ to $m = M$.

The maximum lag M corresponds to the maximum fundamental period T_0(or equivalently the minimum fundamental frequency) that can be estimated:

$$M = T_0^{max} F_s = \frac{F_s}{f_0^{min}} \tag{2.31}$$

The default expected maximum period T_0^{max} is 40 ms, which corresponds to a minimum fundamental frequency of 25 Hz.

If the signal is purely periodic, the maximum values of $\Gamma_l(\mathrm{m})$ will be at lags m corresponding to multiples of T_0. At lags near $m = 0$ a high peak will appear which will very likely reach values

near to 1 for almost any type of audio signal, independently of its degree of periodicity. To obtain the HR, the autocorrelation is searched for the maximum, after having ignored the zero-lag peak:

$$HR = \max_{M_0 \leq m \leq M}\{\Gamma_l(m)\} \tag{2.32}$$

where M_0 denotes a lag immediately to the right of the zero-lag peak. One straightforward possibility is to define M_0 as the lag corresponding to the first zero crossing of the autocorrelation.

It should be noted that, in the MPEG-7 standard, the above equation is written as:

$$HR = \max_{l \leq m \leq N_{hop}}\{\Gamma_l(m)\} \tag{2.33}$$

It can be seen that, on the one side, the zero-lag peak is not ignored, which would result in HR values virtually always close to 1. On the other side, the rightmost limit corresponds only to a frame length, and not to the maximum lag M corresponding to the maximum fundamental period expected [16].

The lag that maximizes $\Gamma_l(m)$ corresponds to the estimated local fundamental period.

The HR values will be close to 0 for white noise and to 1 for purely periodic signals.

2.6.1.2 Upper Limit of Harmonicity

The upper limit of harmonicity (ULH) is an estimation of the frequency beyond which the spectrum no longer has any harmonic structure. It is based on the output/input power ratio of a time domain comb filter [18] tuned to the fundamental period of the signal estimated in the previous paragraph. The algorithm is performed as follows:

1. The comb-filtered signal is calculated as:

$$\tilde{s}_l(n) = s_l(n) - G_l s_l(n - \hat{m}) \quad (0 \leq n \leq N_w - 1) \tag{2.34}$$

where $\hat{m}$ is the lag maximizing the autocorrelation function $\Gamma_l(m)$ in equation 2.32, which corresponds to the estimated fundamental period of frame l. The G_l factor is the optimal gain of the comb filter:

$$G_l = \frac{\sum_{j=0}^{N_w-1} s_l(j)s_l(j-\hat{m})}{\sum_{j=0}^{N_w-1} s_l(j-\hat{m})^2} \tag{2.35}$$

2. The power spectra of the original and comb-filtered signals ($P'(k)$ and $P_c^{'}(k)$, respectively) are computed for each frame l as described in equation 2.22.
3. For each of the spectra $P'(k)$ and $P_c^{'}(k)$, all the power samples falling beyond a given frequency bin k_{lim} are summed. The ratio of the two sums is then taken as follows:

$$R(k_{lim}) = \frac{\sum_{k=k_{lim}}^{(N_{FT}/2)-k_{low}} P_c^{'}(k)}{\sum_{k=k_{lim}}^{(N_{FT}/2)-k_{low}} P^{'}(k)} \tag{2.36}$$

   The maximum frequency bin of the spectra $k_{max} = (N_{FT}/2) - k_{low}$ has been explained in equation (2.22).
4. The ratios $R(k_{lim})$ are computed sequentially, decrementing k_{lim} from $k_{min} = k_{max}$ down to the first frequency bin k_{ulh} for which $R(k_{lim})$ is smaller than a threshold of 0.5.
5. The corresponding frequency f_{ulh} is given by f($k_{ulh} + K_{low}$) as defined in equation 2.13, except if $k_{ulh} = 0$. In the later case, f_{ulh} is set to 31.25 Hz, conforming to the definition of the ASC in equation 2.24.
6. Finally, a ULH feature is computed for each signal frame as:

$$ULH = \log_2\left(\frac{f_{ulh}}{1000}\right) \tag{2.37}$$

   The conversion of the frequency limit f_{ulh} into an octave scale centred on1 kHz makes the ULH coherent with the definitions of the ASC and ASS descriptors in equations 2.24 and 2.25 [9].

The two AH features HR and ULH are designed to provide a compact description of the harmonic properties of sounds. They can be used for distinguishing between *harmonic sounds* (i.e. sounds whose spectra have a harmonic structure, like musical sounds and voiced speech segments) and *non-harmonic sounds* (i.e. sounds with non-harmonic spectra, like noisy sounds and unvoiced speech segments).

2.6.1.2.1 Extraction

In order to extract this descriptor I have used the $AudioHarmonicityD$ matlab code which is in the MPEG-7 Experimental Model [15], but before running it I have made same changes because in the code in some expression the matrix dimension are not agree.

This code considers frame with duration of 40 ms, and for each frame calculate:

$$r_k(j) = \frac{\sum_{i=1}^{n}[s(i+j)-s(i)]^2}{\sum_{i=1}^{n} s(i)^2+\sum_{i=1}^{n} s(i+j)^2} \tag{2.38}$$

The lag j varies from $j = 1$ till the end of the frame. After that the code chooses the minimum and subtracts it from 1:

$$H_k = 1 - \min(r(j)) \tag{2.39}$$

In this way the meaning of the H_k is the same of the HR which I have defined above. In fact this value is 1 for purely periodic signal, and it should be close to 0 for white noise.

The minimum is found using a parabolic approximation, in this way the first values close to zero are not considerate.

The duration of the frame is set to 40 ms because in this way the maximum lag T_0^{max} is 40 ms, which corresponds to a minimum fundamental frequency of 25 Hz.

This code doesn't consider H_k as the HR, but calls HR the ratio between the power spectra of the original and comb-filtered signals ($P'(k)$ and $P_c^{'}(k)$, respectively). For this reason I have changed the code saving H_k as the HR.

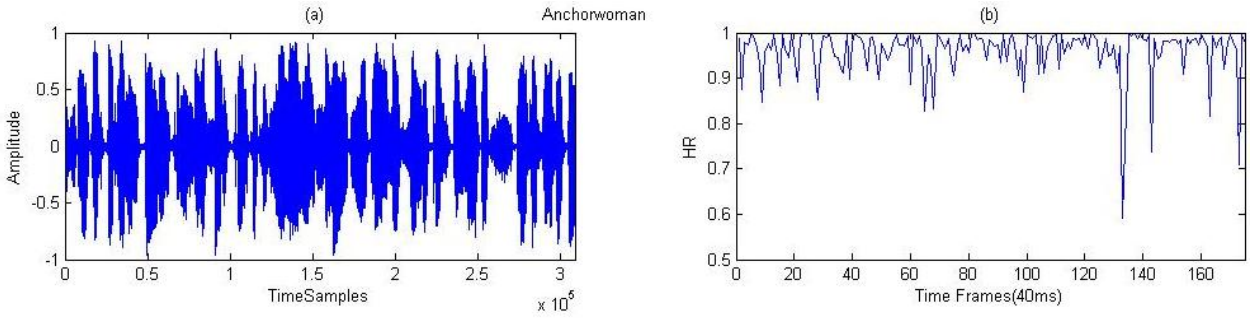

Figure 2.14: (a) 10 s of anchorwoman speech; (b) HR of the same segments

Figure 2.14 gives the temporal series of HR values extracted from 10 s of anchorwoman speech. If we compare the HR with its temporal waveform, we can note that the HR is near 1 when we have piece of speech and is lower when we have piece of silence or noise-like.

The algorithm used in the experimental model matlab code to compute the ULH feature is the same to the one which I have descripted above.

2.6.2 Audio Fundamental Frequency

The audio fundamental frequency (AFF) descriptor provides estimations of the fundamental frequency f_0 in segments where the signal is assumed to be periodic [9].

It is particularly useful to get an approximation of the pitch of any music or speech signals.

Numerous f_0 estimation algorithms are available in the literature. One of the most common approaches is the temporal autocorrelation (TA) method already described in the AH section, in equations 2.30 and 2.32.

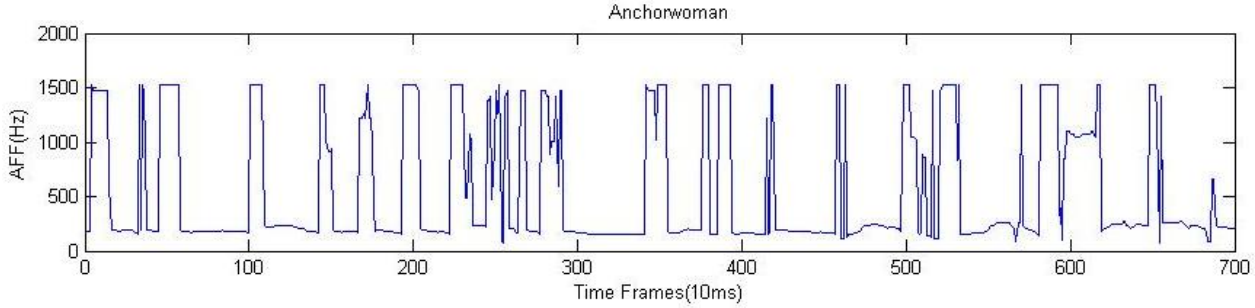

Figure 2.15: AFF of 7 s of anchorwoman speech

The standard does not specify any normative extraction method in order to promote choice of strategy. However in all cases shall be present that the limits of the search range shall be specified using $loLimit$ and $hiLimit$. The extraction method shall report a fundamental frequency for any signal that is periodic over the analysis interval with a fundamental within the search range.

In order to extract the audio fundamental frequency I have used the matlab code of the MPEG-7 experimental model $AudioFundamentalFrequencyD$ which uses the equation 2.30 and 2.32 and set the search range setting the $lofreq$ to 62.5 Hz and the $hifreq$ to 1500 Hz.

In figure 2.15 I have reported the AFF of 7 s of anchorwoman speech and we can note that the fundamental frequency ranges from 62.5 Hz and 1500 Hz as I have said before.

CHAPTER 3
FEATURE TRASFORMATION

The purpose of *sound classification* is to understand whether a particular sound belongs to a certain class.

Many classification systems can be partitioned into components such as the ones shown in the figure 3.1

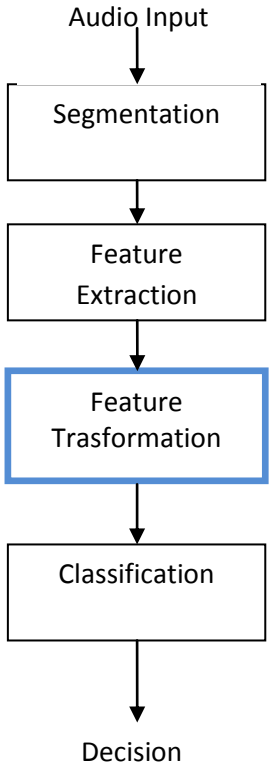

Figure 3 .1: General Sound Classification System

The first step is the segmentation in which the entire signal is divided into relevant sound segments. Then a feature extraction stage extracts properties of the sound that are useful for classification. It is vital that the feature vectors used are rich enough to describe the content of the sound sufficiently.

Spectrum-based features are often considered canonical for audio applications but the direct spectrum features are generally incompatible with classification applications due to their high dimensionality and their inconsistency. Each spectrum slice is an *n*-dimensional vector, with *n* being the number of spectral channels, therefore typical values of a linearly spaced spectrum are between 64 and 1024 dimensions. Probability classifiers require relatively low-dimensional data representations, preferably fewer than 10 dimensions. A logarithmically-spaced frequency spectrum, such as the one octave bandwidth power spectrum, reduces the dimensionality of the representation significantly, but

necessarily disregards much information due to the low frequency resolution. What is required is a representation that makes a compromise between dimensionality reduction and information loss. A well-known technique for reducing the dimensionality of data whilst retaining maximum information is to use data-derived basis functions, such as computed by *principal component analysis* (PCA) [19], *singular value decomposition* (SVD) [20] or *independent component analysis* (ICA) [21]. A spectrogram may be reconstructed using a set of de-correlated frequency basis functions derived using one of these methods. Fewer functions are required to reconstruct a given spectrogram than the total number of frequency channels, hence the possibility for dimensionality reduction.

Then the reduced dimension feature vector are uses by the classifier to assign the sound to a category. The sound classifiers are often based on statistical models. Examples of such classifiers include Gaussian mixture models (GMMs) [22], hidden Markov models (HMMs) [24], neural networks (NNs) [23] and support vector machines (SVMs) [25].

The classification problem can be seen as a batch process employing various stages independently. In practice many systems employ feedback from and to various stages of the process; for instance, when segmenting speech parts in an audio track it is often useful to perform classification at the same time [16]. The choice of the feature vector and the choice of the classifier are critical in the design of sound classification systems. Often prior knowledge plays a major role when selecting such features. In practice many of the feature vectors described in Chapter 2 may be combined to arrive at a "large" compound feature vector for similarity measure or classification. It is usually necessary to train the classifier based on sound data. The data collection can amount to a surprisingly large part of the costs and time when developing a sound classification system. The process of using a collection of sound data to determine the classifier is referred to as training the classifiers and choosing the model.

3.1 MPEG-7 Sound Classification

The MPEG-7 standard [9] has adopted a generalized sound recognition framework, in which dimension-reduced, de-correlated log-spectral features, called the audio spectrum projection (ASP), are used to train HMM for classification of various sounds. The feature extraction of the MPEG-7 sound recognition framework is based on the projection of a spectrum onto a low-dimensional subspace via reduced rank spectral basis functions called the audio spectrum basis

(ASB). To attain a good performance in this framework, a balanced trade-off between reducing the dimensionality of data and retaining maximum information content must be performed, as too many dimensions cause problems with classification while dimensionality reduction invariably introduces information loss.

The MPEG-7 sound recognition classifier is performed using three steps: audio feature extraction, training of sound models, and decoding. Figure 3.2 depicts the procedure of the MPEG-7 sound recognition classifier.

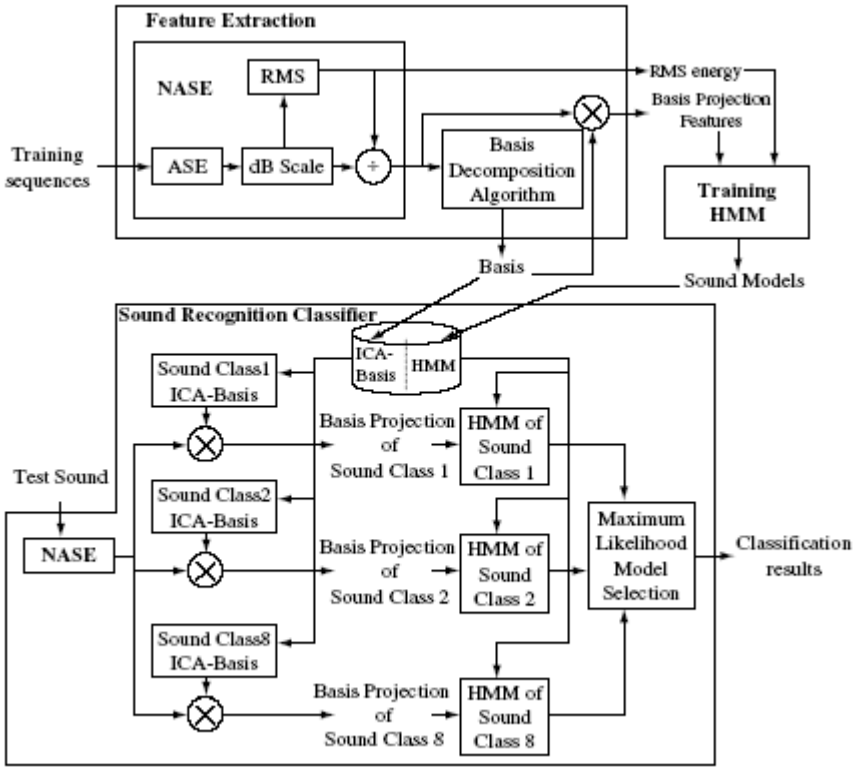

Figure 3.2: MPEG-7 sound recognition classifier

3.2 MPEG-7 Audio Spectrum Projection (ASP) Feature Extraction

The purpose of MPEG-7 feature extraction is to obtain from the audio source a low-complexity description of its content. The starting point is the calculation of the audio spectrum envelope (ASE) descriptor outlined in Chapter 2. Figure 3.2 shows the four steps of the feature extraction in the dimensionality reduction process [26]:

- ASE via short-time Fourier transform (STFT);
- normalized audio spectrum envelope (NASE);

- basis decomposition algorithm – such as SVD or ICA;
- basis projection, obtained by multiplying the NASE with a set of extracted basis functions.

First, the observed audio signal $s(n)$ is divided into overlapping frames. The ASE is then extracted from each frame. The resulting log-frequency power spectrum is converted to the decibel scale:

$$ASE_{dB}(l,f) = 10log_{10}(ASE(l,f)) \tag{3.1}$$

where f is the index of an ASE logarithmic frequency range, l is the frame index.
Each decibel-scale spectral vector is normalized with the RMS energy envelope, thus yielding a normalized log-power version of the ASE called NASE. The full-rank features for each frame l consist of both the RMS-norm gain value R_l and the NASE vector $X(l,f)$:

$$R_l = \sqrt{\sum_{f=1}^{F}(ASE_{dB}(l,f))^2}\ , \quad 1 \le f \le F \tag{3.2}$$

and :

$$X(l,f) = \frac{ASE_{dB}(l,f)}{R_l}\ , \quad 1 \le l \le L \tag{3.3}$$

where F is the number of ASE spectral coefficients and L is the total number of frames.
Much of the information is disregarded due to the lower frequency resolution when reducing the spectrum dimensionality from the size of the STFT to the F frequency bins of NASE.
To visualize the kind of information that the NASE vectors $X(l,f)$ convey, three dimensional (3D) plots of the NASE of 10 seconds of Anchorman and Anchorwoman speech signal respectively shown in Figures 3.3(a),3.3(c).In the example the frequency channels are spaced 1/4-octave bands.
We can note that recognizing the gender of the speaker by visual inspection of the plots is easy. Compared with the female speaker, the male speaker produces more energy at the lower frequencies and less at the higher frequencies.
The figure 3.3 (b), shows the NASE of 7 seconds of a GR theme music, we can note the harmonic nature of the music, observing the almost time-independent spectral peaks of the

NASE. In figure 3.3 (c), instead I have reported the NASE of 10 seconds of a male voice on the telephone in which we can easy note the effects of the a non ideal telephone line

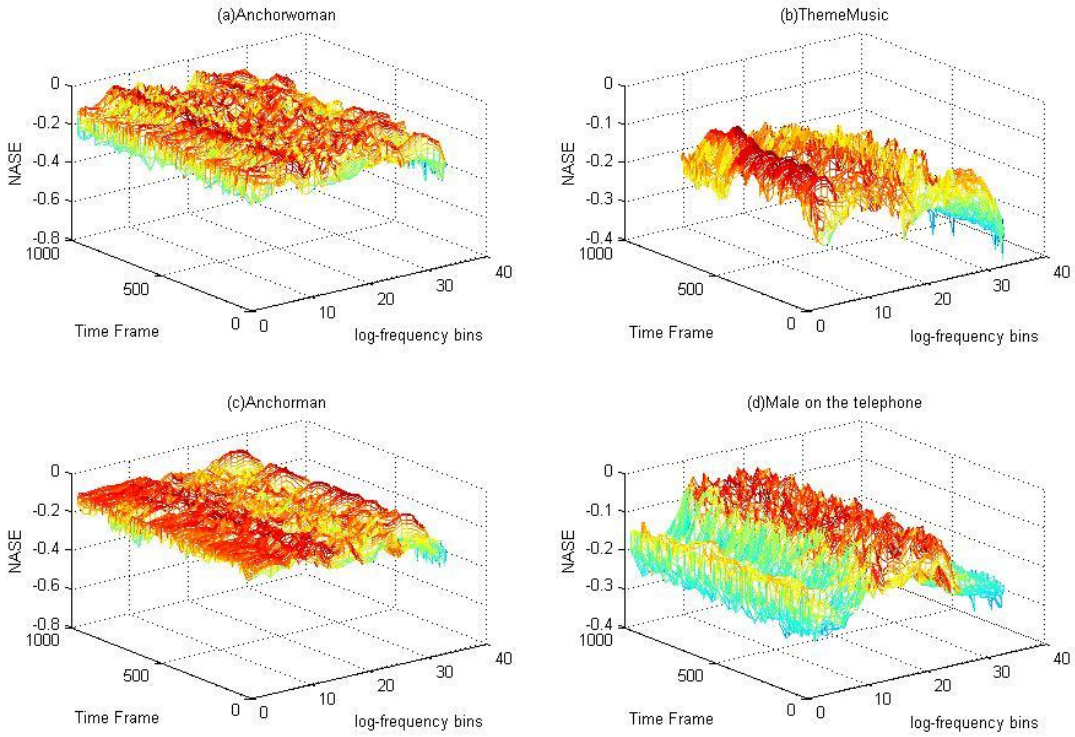

Figure 3.3: 3D NASE with 1/4-octave bands: (a) 10 seconds of anchorwoman voice; (b) 7 seconds of ThemeMusic;(c) 10 seconds of anchorman voice; (d) 10 seconds of a man on the telephone.

3.3 Dimensionality Reduction Using Basis Decomposition

In order to achieve a trade-off between further dimensionality reduction and information loss, the ASB and ASP of MPEG-7 low-level audio descriptors are used. To obtain the ASB, SVD or ICA may be employed. We have the NASE matrix X, in the form of an $L \times F$ time–frequency matrix, where the vertical dimension represents time (i.e. each row corresponds to a time frame index l ($1 \leq l \leq L$)) and the horizontal dimension represents the spectral coefficients (i.e. each column corresponds to a logarithmic frequency range index $f(1 \leq f \leq F)$).

SVD is performed on the feature matrix in the following way:

$$X = UDV^T \tag{3.4}$$

where X is factored into the matrix product of three matrices: the $L \times L$ row basis U matrix, the $L \times F$ diagonal singular value matrix D and the $F \times F$ transposed column basis functions V.

In order to perform dimensionality reduction, the size of the matrix V is reduced by discarding $F - E$ of the columns of V. The resulting matrix V_E has the dimensions $F \times E$. To calculate the proportion of information retained for E basis functions we use the singular values contained in matrix D:

$$I(E) = \frac{\sum_{i=1}^{E} D(i,i)}{\sum_{j=1}^{F} D(j,j)} \tag{3.5}$$

where $I(E)$ is the proportion of information retained for E basis functions and F is the total number of basis functions, which is also equal to the number of spectral bins. The SVD transformation produces de-correlated, dimension-reduced bases for the data, and the right singular basis functions are cropped to yield fewer basis.

3.4 Statistically Independent Basis

After extracting the reduced SVD basis V_E, ICA is employed for applications that require maximum de-correlation of features, such as the separation of the source components of a spectrogram.

ICA assumed the data to be linear mixtures of some unknown latent variables, and the mixing system is also unknown. The latent variables are assumed to be non-Gaussian and mutually independent. They are called the independent components of the observed data. These independent components, also called sources or factors, can be found by ICA.

ICA is a statistical method which not only de-correlates the second-order statistics but also reduces higher-order statistical dependencies. Thus, ICA produces mutually uncorrelated bases. The independent components of matrix X can be thought of as a collection of statistically independent bases for the rows (or columns) of X. The $L \times F$ matrix X is decomposed as:

$$X = WS + N \tag{3.6}$$

where S is the $P \times F$ source signal matrix, W is the $L \times P$ mixing matrix (also called the matrix of spectral basis functions) and N is the $L \times F$ matrix of noise signals. Here, P is the number of independent sources [16].

We are interested to found a matrix which transforms our data X in its independent component, so from the above expression this matrix is the pseudo inverse of W.

The above decomposition can be performed for any number of independent components and the sizes of W and S vary accordingly. To find a statistically independent basis using the basis functions, the well known ICA algorithms, such as INFOMAX, JADE [27] or FastICA [3], can be used.

The ICA basis is the same size as the SVD basis. The retained information ratio, *I(K)*, is equivalent to the SVD when using the given extraction method.[26]

The ICA basis are then stored in the columns of a matrix within the $AudioSpectrumBasisType$ descriptor. The following code example in the listing 3.1 shows an instance of the $AudioSpectrumBasis$ descriptor for a 32-dimensional spectrum using a subset of 4 basis functions:

```
<AudioDescriptor xsi:type="AudioSpectrumBasisType"
loEdge="62.5" hiEdge="8000" octaveResolution="1/4">
<SeriesOfVector totalNumOfSamples="1" vectorSize="32 4">
<Raw dim="32 4">
0.082 -0.026 0.024 -0.093
0.291 0.073 0.025 -0.039
0.267 0.062 0.030 -0.026
0.267 0.062 0.030 -0.026
0.271 -0.008 0.039 0.007
0.271 -0.008 0.039 0.007
0.269 -0.159 0.062 0.074
<-- more values here ... -->
0.010 -0.021 0.063 -0.103
</Raw>
</SeriesOfVector>
</AudioDescriptor>
```

Listing 3.1: Example code of an instance of the $AudioSpectrumBasisType$ descriptor.

The ASP Y is obtained by multiplying the NASE matrix with the basis vectors obtained after the SVD decomposition and the ICA transformation.

$$Y = XV_E W \tag{3.7}$$

The spectrum projection features and RMS-norm gain values are used as input to the classifier and are stored in the $AudioSpectrumProjection$ descriptor.

The elements of each $AudioSpectrumProjection$ vector shall represent, in order, the L2-norm value, R_l, of the ASE. This shall be followed by the inner product of the normalized spectral frame, $X(l,f)$ and each of the basis vectors, B_k. The resulting vector has $E+1$ elements, where E is the number of basis components, and it is defined by:

$$Y = [R_l \;\; X(l,f)B_1 \;\; X(l,f)B_2 \;\; \ldots \;\; X(l,f)B_E] \tag{3.8}$$

The next code example in the listing 3.2 shows an instance of the *AudioSpectrumProjection* descriptor representing features derived from the basis vectors in the previous example.

```
<AudioDescriptor xsi:type="AudioSpectrumProjectionType">
<SeriesOfVector hopSize="PT10N1000F" totalNumOfSamples="263"
vectorSize="4">
<Raw dim="263 4">
0.359 -0.693 0.345 -0.145
0.364 -0.690 0.308 -0.147
0.353 -0.656 0.382 -0.175
<-- more values here ... -->
0.998 -0.342 0.569 0.592
1.000 -0.324 0.562 0.601
</Raw>
</SeriesOfVector>
</AudioDescriptor>
```

Listing 3.2: Example code of an instance of the *AudioSpectrumProjection* descriptor

3.5 Plots of MPEG-7 Audio Spectrum Projection

To compute the ASP I have used the matlab codes $AudioSpectrumBasiD$ and $AudioSpectrumProjectionD$ that are in the MPEG-7 Experimental Model [15], instead to compute the ICA transformation I have download the Cardoso matlab code [28] which implements the jade algorithm.

To visualize the kind of information that the MPEG-7 ASP features convey, results of NASE and ASP descriptors of a male voice and of a male voice on the telephone are depicted in Figures 3.4–3.18.In the figure 3.4(a) is clear evident that the power at the higher and lower frequency are cut by the telephonic channel.

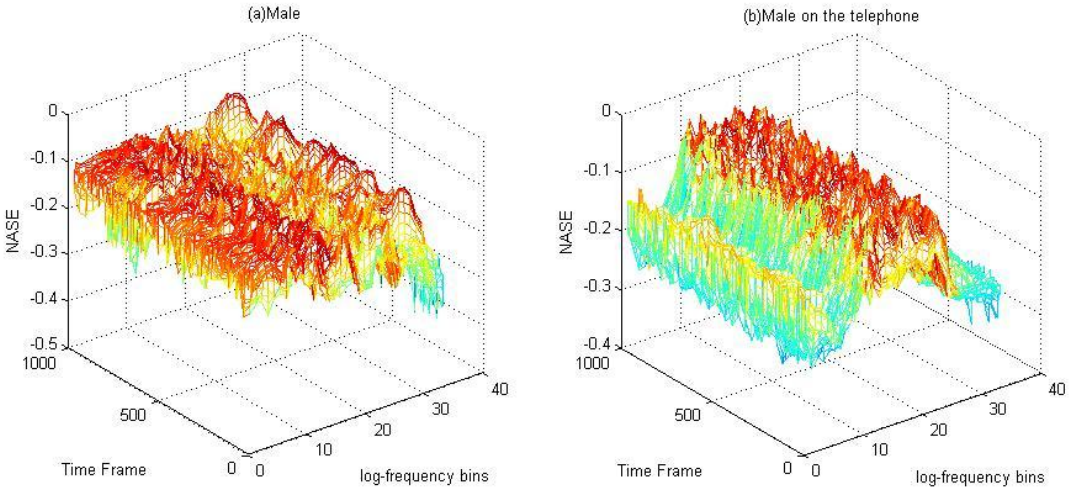

Figure 3.4: 3D NASE with 1/4-octave bands; (a)10 seconds of anchorman voice;
(b) 10 seconds of a man on the telephone.

So distinguishing between speech and speech on the telephone line .is easy by visual inspection of the NASE. While this visual interpretation of the NASE is rather easy, visual interpretation of the bases C_E in Figure 3.5 is not so straightforward. Each of these bases is a matrix, which can be thought as a linear transformation between a spectral domain containing correlated information (NASE) and ICA basis vectors, in which the correlations in the information are reduced. Since we do not know exactly how the correlations are being reduced in each case, the bases are difficult to interpret. However, we can note that the basis have peaks on average due to larger variances this is justified by the fact that the ICA algorithm uses a non-linear technique to de-correlate the NASE.

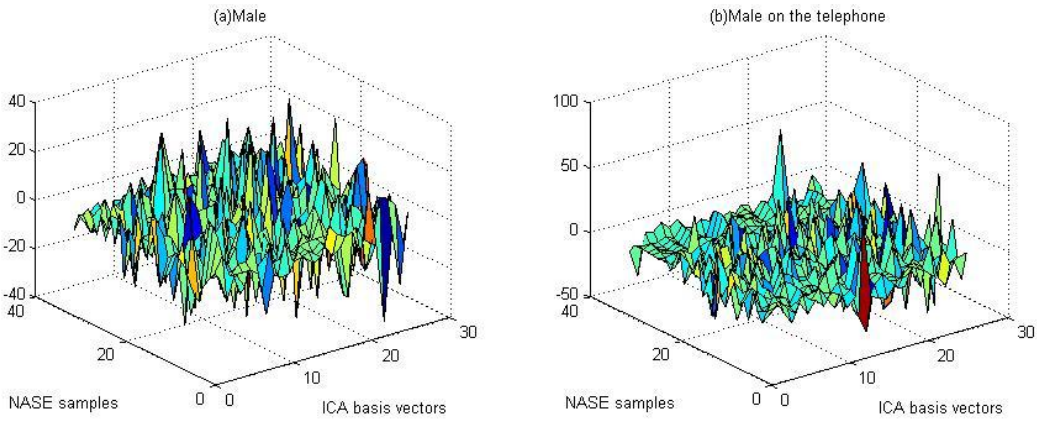

Figure 3.5: ICA basis of the: (a)10 seconds of anchorman voice;
(b) 10 seconds of a man on the telephone.

In Figure 3.6 I have reported the projections $Y = XC_E W$, which we can view as a versions of the NASE where the frequency information is scrambled by the basis. As can be verified is still possible distinguish the two signals. In fact in Figure 3.6 (b) the peaks of the ASP are concentrate in correspondence of the telephonic band.

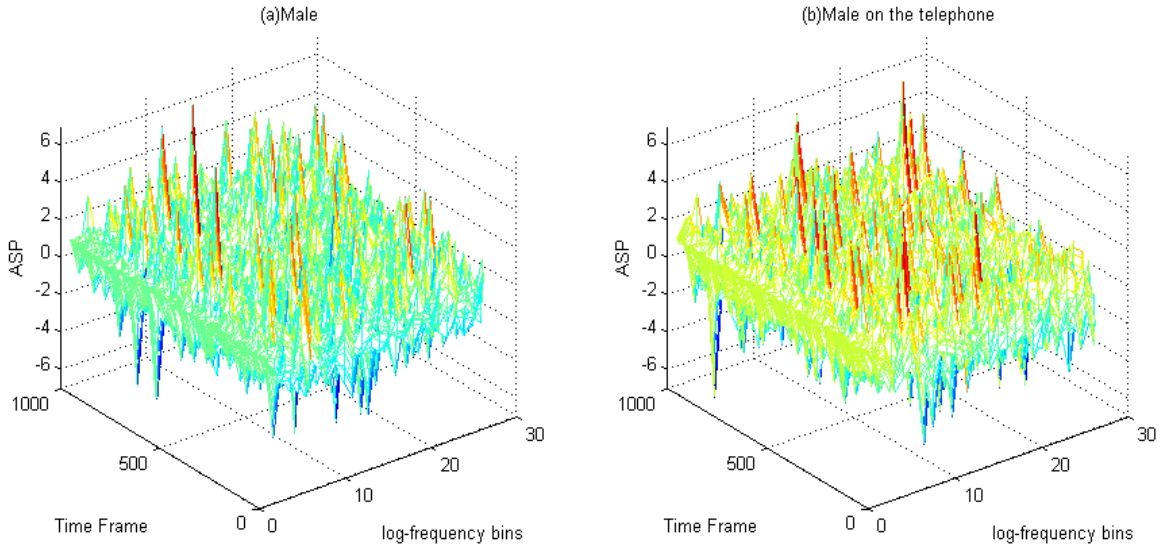

Figure 3.6: ASP with frequency dimension 25 of the: (a)10 seconds of anchorman voice; (b) 10 seconds of a man on the telephone.

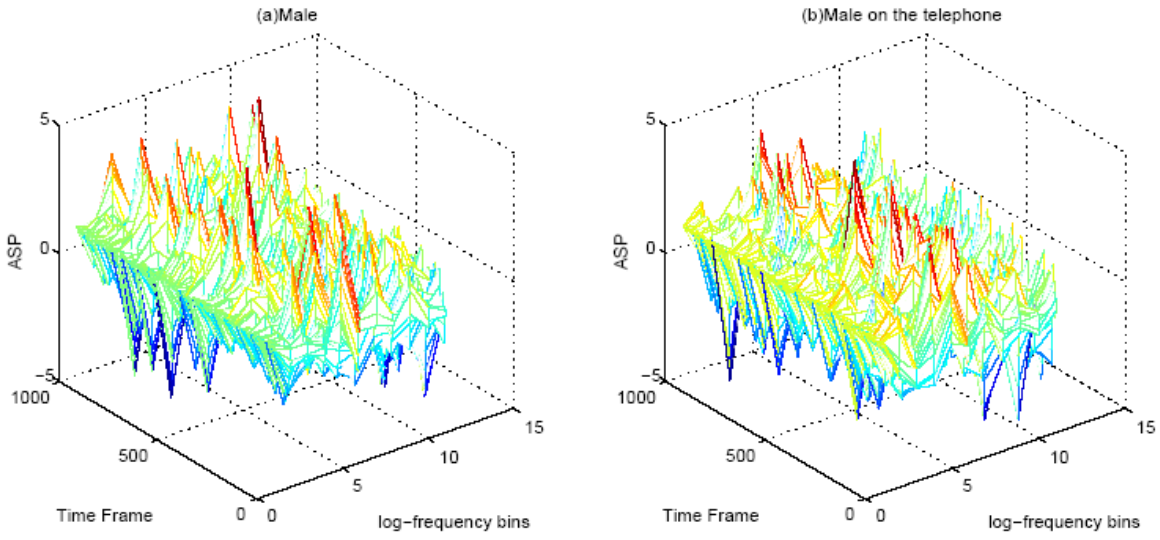

Figure 3.7 ASP with frequency dimensions 10 of the: (a)10 seconds of anchorman voice; (b) 10 seconds of a man on the telephone.

If we compare the ASP depicted in Figure 3.6, which have a frequency dimension of 25,with the ASP in Figure 3.7, which have a frequency dimension of 10, we can note that in Figure 3.7 the features of the two signals are more evident, in fact as we will view also in the next chapters the increase of the feature vector dimension doesn't assure the improvement of the performances this means that the optimum feature vector dimension must be choose after several simulation.

3.6 Reconstruction of an independent spectrogram frame

The reconstruction of an independent spectrogram frame, $X(j)$, is calculated by taking the outer product of the *j*th vector in $AudioSpectrumBasis$ and the j+1th vector in $AudioSpectrumProjection$:

$$X(j) = Y(j+1)pinv(B_j) \tag{3.9}$$

These frames are concatenated to form a new spectrogram. Any combination of spectrogram subspaces can be summed to obtain either individual source spectrograms or an approximation of the original spectrogram [9].
The salient features of a spectrogram may be efficiently represented with much less data than a full spectrogram using independent component basis functions. In The following figure I report the original full-bandwidth NASE of the speech on telephone segment analyzed above and **Errore. L'origine riferimento non è stata trovata.** shows a 6-component reconstruction of the same descriptor. The data ratio, *R*, between the reduced-dimension spectrum and the full-bandwidth spectrum is

$$R = K\left(\frac{1}{M} + \frac{1}{N}\right) \tag{3.10}$$

where *K* is the number of basis components, *M* is the number of frames in the spectrogram and *N* is the number of frequency bins. For example, a 5-component summary of 500 frames of a 64-bin spectrogram leads to a data reduction of 11:1.

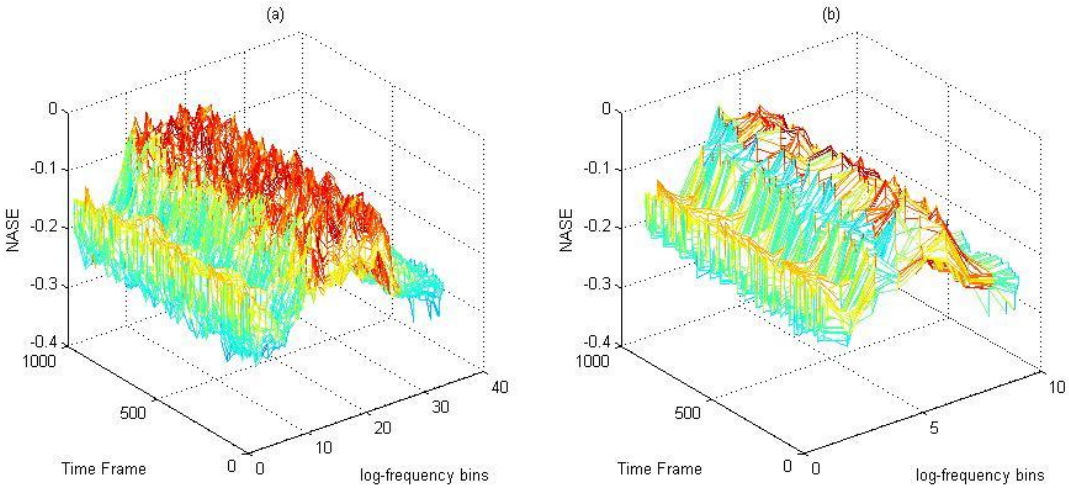

Figure 3.8: 3D NASE with 1/4-octave bands of 10 seconds of a man on the telephone: (a) full dimension (b) Reconstruct with only 6 basis function

CHAPTER 4
THE CLASSIFIER

Once feature vectors are generated from audio clips, and if required reduced in dimension, these are fed into classifiers. The model-based classifiers that have been most often used for audio classification include Gaussian mixture model (GMM) classifiers, neural network (NN) classifiers, hidden Markov model (HMM) classifiers, and support vector machines (SVMs).
The classifier proposed in the MPEG-7 standard is the Hidden Markov Models, so in the following paragraphs I will analyze this model in detail.

4.1 The Hidden Markov Model

An HMM is a statistical method, widely used in the pattern classification field.
Very successful applications based on HMM include speech recognition, speaker verification and handwriting recognition. HMMs are used to model processes with time varying characteristics.
An HMM can be described as:

- A set of N_S states $\{S_i\}$
- A set of state transition probabilities $\{a_{ij}\}$, where a_{ij} is the probability of transition from state S_i to state S_j.
- A set of d-dimensional probability density functions $b_j(x)$, where b_j is the density function of state S_j .
- A set of initial state probabilities $\{\pi_i\}$, where π_i is the probability that S_i is the initial state.

The system starts at time 0 in a state S_i with a probability π_i. When in a state S_i at time l the system moves at time $l+1$ to state S_j with a probability a_{ij} and so on, generating a sequence of L observation vectors x_l. An HMM is completely specified by the three sets $\{a_{ij}\}\{b_j\}$ and $\{\pi_i\}$. Continuous HMMs generally set $b_j(x)$ to a multivariate Gaussian distribution with mean μ_j and covariance matrix Σ_j, giving $b_j = \{\mu_j, \Sigma_j\}$ for each state.
In the following paragraphs I will use the following notation:
N = number of states in the model
M= total number of distinct observation symbol

T = length of observation sequence i.e. the number of symbols observed

i_t denotes the state in which we are at time t

$V = \{V_1, \dots, V_M\}$ the discrete set of possible observation symbols

$\pi = \{\pi_i\}$, $\pi_i = P(i_1 = i)$, the probability of being in state i at the beginning of the experiment i.e. at $t = 1$

$A = \{a_{ij}\}$ where $a_{ij} = P(i_{t+1} = j | i_t = i)$ the probability of being in state j at time t+1 given that

we were in state i at time t. We assume that a_{ij} are independent of time.

$B = \{b_j(k)\}, b_j(k) = P(i_{t+1} = j | i_t = i)$, the probability of observing the symbol v_k given that we are in state j.

O_t will denote the observation symbol observed at instant t

$\lambda = (A, B, \pi_i)$ will be used as a compact notation to denote an HMM.

4.2 The Three Problems for HMMs

Most applications of HMMs are finally reduced to solving three main problems. These are:

Problem 1: Given the model $\lambda = (A, B, \pi)$ how do we compute $P(O|\lambda)$ the probability of occurrence of the observation sequence $O = O_1, O_2, \dots, O_T$.

Problem 2: Given the model $\lambda = (A, B, \pi)$ how do we choose a state sequence $I = i_1, i_2, \dots, i_T$ so that $P(O, I|\lambda)$, the joint probability of the observation sequence $O = O_1, O_2, \dots, O_T$ and the state sequence given the model is maximized.

Problem 3: How do we adjust the HMM model parameters $\lambda = (A, B, \pi)$ so that $P(O|\lambda)$ or $P(O, I|\lambda)$ is maximized.

Problems 1 and 2 can be viewed as analysis problems while Problem 3 is a typical synthesis (or model identification or training) problem.

4.3 Solutions to the Problem 1

A most straightforward way to determine $P(O|\lambda)$ is to find $P(O|I, \lambda)$ for a fixed state sequence

$I = i_1, i_2, \dots, i_T$ then multiply it by $P(I|\lambda)$ and then sum up over all possible I's. We have

$$P(O|I, \lambda) = b_{i_1}(O_1) b_{i_2}(O_2) \dots b_{i_T}(O_T) \tag{4.1}$$

$$P(I|\lambda) = \pi_{i_1} a_{i_1 i_2} a_{i_2 i_3} \dots a_{i_{T-1} i_T} \tag{4.2}$$

Hence we have:

$$P(O|\lambda) = \sum_I P(O|I,\lambda)\, P(I|\lambda) \tag{4.3}$$

$$= \sum_I \pi_{i_1} b_{i_1}(O_1) a_{i_1 i_2} b_{i_2}(O_2) \dots a_{i_{T-1} i_T}\, b_{i_T}(O_T) \tag{4.4}$$

where $I = i_1, i_2, \dots, i_T$

From (4.4) we see that the summand of this equation involves $2T - 1$ multiplications and there exists N^T distinct possible state sequences I. Hence a direct computation from (4.4) will involve of the order of $2TN^T$ multiplications. Even for small values, $N = 5$ and $T = 100$ this means approximately 10^{72} multiplications.

Hence we see that a more efficient procedure is required to solve Problem 1; such a procedure exists and is called the *forward-backward* procedure.

4.3.1 Forward-Backward Procedure

Consider the forward variable $\alpha(t_i)$ defined as:

$$\alpha_t(i) = P(O_1, O_2, \dots, O_t, i_t = i|\lambda) \tag{4.5}$$

i.e. the probability of the partial observation sequence up to time t and the state i at time t given the model λ, $\alpha_t(i)$ can be computed inductively as follows:

1. $\alpha_1(i) = \pi_i b_i(O_1)\,, \qquad 1 \le i \le N$ (4.6)
2. for $t = 1,2, \dots, T-1,\ 1 \le j \le N$

   $\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j(O_{t+1})$ (4.7)
3. then we have:

   $P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i)$ (4.8)

In Step 2 we want to compute the probability of partial observation sequence up to time t +1 and state j at time $t + 1$, state j can be be reached (with probability a_{ij}) independently from any of the N states at time t; The summation in Eq. 4.7 refers to this fact. Also the summand

gives observation sequence up to time t; hence the $b_j(O_{t+1})$ outside the brackets. In Step 3 we just sum up all possible (independent) ways of realizing the given observation sequence.
Now let us examine the number of computations involved in this algorithm. Step 1 involves N multiplications. In Step 2 the summation involves N multiplications plus one for the out of bracket $b_j(O_{t+1})$ term-this has to be done for $j = 1$ to N and $t = 1$ to $T - 1$ making the total number of multiplications in Step 2 as $(N + 1)N(T - 1)$. Step 3 involves no multiplications. Hence total number of multiplications is $N + N(N + 1)(T - 1)$ i.e. of the order of N^2T as compared to $2T \cdot N^T$ required for the direct method. For N=5 and T=100 we need about 3000 computations for the forward method as compared to 10^{72} required by the direct method-a saving of about 69 orders of magnitude.

4.3.2 Backward Procedure

In a similar manner we may define a backward variable $\beta_t(i)$ as:

$$\beta_t(i) = P(O_{t+1}, O_{t+2}, \dots, O_T | i_t = i, \lambda) \tag{4.10}$$

i.e. the probability of the observation sequence from $t + 1$ to T given the state i at time t and the model λ Note that here $i_t = i$ has already been given (it wasn't in the case of the forward variable). This distinction has been made to be able to combine the forward and the backward variables to produce useful results as we shall see soon. We can easily solve for $\beta_t(i)$ as done for $\alpha_t(i)$:

1. $\beta_T(i) = 1, \quad 1 \le i \le N$ (4.11)

2. for $t = T - 1, T - 2, \dots, 1, \quad 1 \le i \le N$
   $\beta_t(i) = \sum_{j=1}^{N} a_{ij}\, b_j(O_{t+1})\beta_{t+1}(j)$ (4.12)

3. $P(O|\lambda) = \sum_{i=1}^{N} \pi_i b_i(O_1)\beta_1(i)$ (4.13)

The computation of $P(O|\lambda)$ using $\beta_t(i)$ also involves of the order of N^2T calculations. Hence both the forward as well as the backward method is equally efficient for the computation of $P(O|\lambda)$. This solves Problem 1.

4.4 Solutions to the Problem 2

Here we have to find a state sequence $I = i_1, i_2, \dots, i_T$ such that probability of occurrence of the observation sequence $O = O_1, O_2, \dots O_T$ from this state sequence is greater than that from any other state sequence. In other words, our problem is to find *I* that will maximize $P(O, I|\lambda)$. There is a famous algorithm to do this called the *Viterbi Algorithm* [30]. It is an inductive algorithm in which at each instant you keep the best (i.e. the one giving maximum probability) possible state sequence for each of the *N* states as the intermediate state for the desired observation sequence $O = O_1, O_2, \dots, O_T$. In this way you finally have the best path for each of the *N* states as the last state for the desired observation sequence. Out of these we select the one which has highest probability. In order to get an idea of the *Viterbi algorithm* as applied to the optimum state estimation problem a simple reformulation of the problem will be useful

Consider the expression for $P(O, I|\lambda)$; from (4.1)-(4.2)we have:

$$\begin{aligned} P(O, I|\lambda) &= P(O|I, \lambda)P(I|\lambda) \\ &= \pi_{i_1} b_{i_1}(O_1) a_{i_1 i_2} b_{i_2}(O_2) \dots a_{i_{T-1} i_T} b_{i_T}(O_T) \end{aligned} \tag{4.13}$$

Now we define

$$U(i_1, i_2, \dots, i_T) = -\left[\ln\left(\pi_{i_1} b_{i_1}(O_1)\right) + \sum_{t=2}^{T} \ln(a_{i_{t-1} i_t} b_{i_t}(O_t))\right] \tag{4.14}$$

then it is easily seen that

$$P(O, I|\lambda) = e^{-U(i_1, i_2, \dots, i_T)} \tag{4.15}$$

consequently the problem of optimal state estimation, namely,

$$\max_{\{i_t\}_{t=1}^{T}} P(O, i_1, \dots i_T|\lambda) \tag{4.16}$$

becomes equivalent to

$$\min_{\{i_t\}_{t=1}^{T}} U(i_1, i_2, \dots, i_T) \tag{4.17}$$

This reformulation now enables us to view terms like $-\ln(a_{i_j i_k} b_{i_k}(O_t))$ as the cost associated in going from state i_j to state i_k at time t. The *Viterbi algorithm* to find the optimum state sequence can now be described as follows: Suppose we are currently in state i and we are considering visiting state j next. We shall say that the weight on the path from state i to state j is $-\ln(a_{ij}\, b_j\,(O_t))$ (i.e. the negative of logarithm of probability of going from state i to state j and selecting the observation symbol O_t in state j) where O_t is the observation symbol selected after visiting state j.

When the initial state is selected as state i the corresponding weight is $-\ln(\pi_i b_i(O_1))$ we shall call this the initial weight.

We define the weight of a sequence of states as the sum of the weights on the adjacent states. Note that this actually corresponds to multiplying the corresponding probabilities. Now finding the optimum sequence is merely a matter of finding the path (i.e. a sequence of states) of minimum weight through which the given observation sequence occurs [29].

Note that the *Viterbi Algorithm* is essentially a dynamic programming approach for minimizing $U(i_1, i_2, \dots i_T)$.

We shall illustrate with an example on how the minimization is done. Consider a three state HMM. Figure 4.1 shows the weights assigned to various paths as described above. The numbers in circles show the initial weights assigned to the corresponding states. To simplify understanding of the algorithm we assume that the weights are symmetrical (i.e. the weight from state i to state j is same as the weight from state j to state i). Also we assume the weights do not change with time which in general will not be true but then once this simplified case is understood the extension to practical case is straight forward.

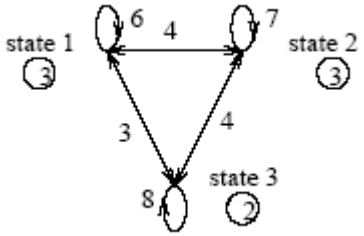

Figure 4.1: The state machine.

In Figure 4.2 I reported an example of how the *Viterbi* Algorithm works if we have an observation sequence $O_1, \ldots, O_4$ of length four. At time $t = 2$ state 1 can be visited from any of the three states that we had at time $t = 1$. We find out the weights on each of these paths and add corresponding weights to the initial weights (we shall call this cumulative weight at state 1 as the cost at state 1 at time $t = 2$). Thus going from state 1 to state 1 the cost at state 1 is 3+6=9. Similarly the cost at state 1 in going from state 2 to state 1 is 3 + 4 = 7 and the cost in going from state 3 to state 1 is $2 + 3 = 5$. Of these the cost 5 is minimum. Hence we retain this cost at state 1 for further calculations. The minimum cumulative weight paths are shown in the figure by arrowed lines. The cumulative weights have been shown in bold alongside the respective states at each time instant.We repeat the above procedure again for $t = 3$ but now using the costs calculated above for each state rather than the initial weights. And the same procedure is repeated for $t = 4$. Now select out the state which has minimum cost of all the states. We see that state 1 is the required state with a cost of 11. Back tracing the sequence of states through which we got at state 1 at time t=4 gives the required sequence of states through which the given observation sequence has highest probability of occurrence. As can be seen from the figure this state sequence is state 3, state 1, state 3, state 1. This sequence has been shown in bold in the figure.

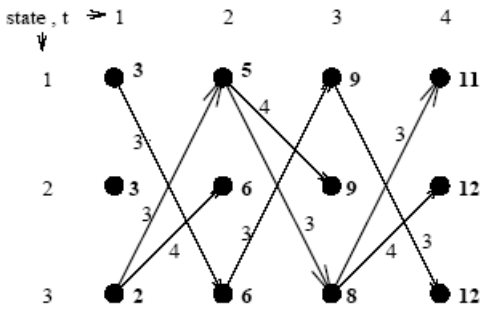

Figure 4.2 How a path of minimum cost is traced out using the *Viterbi algorithm*.

Now that we know how the *Viterbi* Algorithm works here is how it can be implemented [30]: $\delta_t(i)$ denotes the weight accumulated when we are in state i at time t as the algorithm proceeds and $\psi_t(j)$ represents the state at time $t-1$ which has the lowest cost corresponding to the state transition to state j at time t. For example in Figure 4.2:

$$\delta_3(1) = 9,\ \delta_3(2) = 9,\ \delta_3(3) = 8 \tag{4.18}$$

and

$$\psi_4(1) = 3,\ \psi_4(2) = 3,\ \psi_4(3) = 1 \tag{4.19}$$

1. Initialization

   For $1 \le i \le N$

   $$\delta_1(i) = -\ln(\pi_i) - \ln(b_i\,(O_1)) \tag{4.20}$$

   $$\psi_1(i) = 0 \tag{4.21}$$

2. Recursive computation

   For $2 \le t \le T$ for $1 \le j \le N$

   $$\delta_t(j) = \min_{1\le i\le N}\left[\delta_{t-1}(i) - \ln(a_{ij})\right] - \ln b_j\,(O_t)) \tag{4.22}$$

   $$\psi_t(j) = arg\,\min_{1\le i\le N}\left[\delta_{t-1}(i) - \ln(a_{ij})\right] \tag{4.23}$$

3. Termination

   $$P^* = \min_{1\le i\le N}[\delta_T(i)] \tag{4.24}$$

   $$q_T^* = \arg\min_{1\le i\le N}[\delta_T(i)] \tag{4.25}$$

4. Tracing back the optimal state sequence

   For $t = T-1,\ T-2, \dots, 1$

   $$q_t^* = \psi_{t+1}(q_{t+1}^*) \tag{4.26}$$

Hence

e^{-P^*}

gives the required state-optimized probability, and

$$Q^* = \{q_1^*, q_2^*, \dots, q_T^*\} \tag{4.27}$$

is the optimal state sequence.

A little reflection over the above steps will show that computationally the *Viterbi Algorithm* is similar to the forward (-backward) procedure except for the comparisons involved for finding the maximum value. Hence its complexity is also of the order of N^2T this solves Problem 2.

4.5 Solutions to the Problem 3

The third, and by far the most difficult, problem of HMMs is to determine a method to adjust the model parameters (A, B, π) to maximize the probability of the observation sequence given the model. There is no known way to analytically solve for the model which maximizes the probability of the observation sequence. In fact, given any finite observation sequence as training data, there is no optimal way of estimating the model parameters. We can, however, choose $\lambda = (A, B, \pi)$ such that $P(0|\lambda)$ is locally maximized using an iterative procedure such as the *Baum-Welch method*. (or equivalently the EM (expectation-modification) method [31]), or using gradient techniques [32]. In the following paragraph I will describe the *Baum Welch Algorithm* [29].

4.5.1 The *Baum-Welch* re-estimation Formulas

This method assumes an initial HMM model which is improved upon by using the formulas which maximize $P(O|\lambda)$. An initial HMM can be constructed in any way. This algorithm maximizes $P(O|\lambda)$ by adjusting the parameters of λ. This optimization criterion is called the *maximum likelihood criterion*. The function $P(O|\lambda)$ is called the *likelihood function.*

Before we get down to the actual formulas we shall introduce some concepts and notations that shall be required in the final formulas.

Consider $\gamma_t(i)$ defined as follows:

$$\gamma_t(i) = P(i_t = i|O, \lambda) \tag{4.28}$$

i.e. the probability of being in state i at time t given the observation sequence $O = O_1, O_2, \dots, O_T$ and the model $\lambda = (A, B, \pi)$. By Bayes law we have:

$$\gamma_t(i) = \frac{P(i_t=i, O|\lambda)}{P(O|\lambda)} = \frac{\alpha_t(i)\beta_t(i)}{P(O|\lambda)} \tag{4.29}$$

where $\alpha_t(i)$ and $\beta_t(i)$ are defined by (4.5)and (4.10) respectively $\alpha_t(i)$ accounts for $O_1, O_2, \dots O_t$ and state i at time t and $\beta_t(i)$ accounts for $O_{t+1}, O_{t+2}, \dots O_T$ given state i at time t.

We also define $\xi_t(i,j)$ as:

$$\xi_t(i,j) = P(i_t = i, i_{t+1} = j | O, \lambda) \tag{4.30}$$

i.e. the probability of being in state i at time t and making a transition to state j at time $t+1$ given the observation sequence $O = O_1, O_2, \dots, O_T$ and the model $\lambda = (A, B, \pi)$. Using Bayes law

it is seen that

$$\xi_i(i,j) = \frac{P(i_t=i, i_{t+1}=j, O|\lambda)}{P(O|\lambda)} \tag{4.31}$$

But $O = O_1, O_2, \dots, O_T$. Hence

$$\begin{aligned} The\ Numerator &= P(i_t = i, O_1, \dots, O_t, O_{t+1}, \dots, O_T, i_{t+1} = j\ |\lambda) \\ &= P(i_t = i, O_1, \dots, O_t|\lambda) P(O_{t+1}, \dots, O_T, i_{t+1} = j\ |\lambda) \end{aligned} \tag{4.32}$$

(*since the Markov chain is causal*)

Consider the second term. The observation symbol at time $t+1$ is O_{t+1} at which time we are required to be in state j; this means that the observation symbol O_{t+1} has to be picked up from state j. Hence we have

$$\begin{aligned} P(O_{t+1}, \dots, O_T, i_{t+1} = j\ |\lambda) &= P(\ i_{t+1} = j, O_{t+1}|\lambda) P(O_{t+1}, \dots, O_T\ |i_{t+1} = j\ , \lambda) \\ &= a_{ij}\, b_j(O_{t+1}) \beta_{t+1}(j) \end{aligned} \tag{4.33}$$

(*since $i_t = i$ is known from (4.32))*

Hence combining (4.5), (4.31), (4.32) and (4.33) we get

$$\xi_t(i,j) = \frac{\alpha_t(i) a_{ij}\, b_j(O_{t+1}) \beta_{t+1}(j)}{P(O|\lambda)} \tag{4.29}$$

Here $\alpha_t(i)$ accounts for $O_1 \dots O_t$, a_{ij} accounts for the transition to state j, $b_j(O_{t+1})$ picks up the symbol O_{t+1} from state j and $\beta_{t+1}(j)$ accounts for the remaining observation sequence $O_{t+2}, \dots, O_T$.

If we sum up $\gamma_t(i)$from $t+1$ to T we get a quantity which can be viewed as the expected number of times state i is visited or if we sum up only up to $T-1$ then we shall get the expected number of transitions out of state i (as no transition is made at $t = T$). Similarly if $\xi_t(i,j)$ be summed up from t=1 to $T-1$, we shall get the expected number of transitions from state i to state j. Hence

$$\sum_{t=1}^{T-1} \gamma_t(i) = Expected\ number\ of\ transitions\ from\ state\ i \quad (4.30)$$

$$\sum_{t=1}^{T-1} \xi_t(i,j) = Expected\ number\ of\ transitions\ from\ state\ i\ to\ state\ j \quad (4.31)$$

Now we are prepared to present the Baum-Welch re-estimation formulas:

$$\hat{\pi}_i = \gamma_t(i), \ \ 1 \le i \le N \quad (4.32)$$

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \quad (4.33)$$

$$\hat{b}_j(k) = \frac{\sum_{t=1_{O_t=k}}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)} \quad (4.34)$$

The re-estimation formula for π_i is simply the probability of being in state i at time t. The formula for $a_{ij}(i)$ is the ratio of expected number of time of making a transition from state i to state j to the expected number of times of making a transition out of state i. The formula for $b_k(i)$ is the ratio of the expected number of times of being in state j and observing symbol O_k to the expected number of times of being in state j. Note that the summation in the last formula goes up to $t = T$ [29]. If we denote the initial model by λ and the re-estimation model by$\hat{\lambda}$ consisting of the parameters estimated above then it can be shown that either:

1. The initial model λ is a critical point of the likelihood function in which case $\hat{\lambda} = \lambda$, or
2. $P(O|\hat{\lambda}) > P(O|\lambda)$, i.e. we have found a better model from which the observation sequence $O = O_1, O_2, \dots O_T$ is more likely to be produced.

Hence we can go on iteratively computing until $P(O|\lambda)$ is maximized. This solves Problem 3.

4.6 Types of HMMs

Until now we have only considered the special case of ergodic or fully connected HMMs in which every state of the model could be reached (in a single step) from every other state of the model. (Strictly speaking, an ergodic model has the property that every state can be reached from every other state in a finite number of steps.)

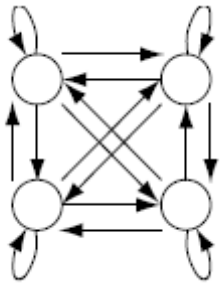

Figure 4.3: Hergodig HMM

As shown in figure 4.3 for an $N = 4$ state model, this type of model has the property that every a_{ij} coefficient is positive. Hence for the example of figure 4.3 we have:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}$$

For some applications other types of HMMs have been found to account for observed properties of the signal being modeled better than the standard ergodic model. One such model is shown in figure 4.4 This model is called a left-right model or a Bakis model [29]

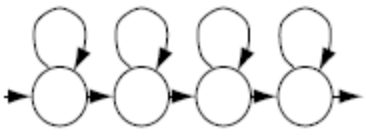

Figure 4.4: Left-right HMM

because the underlying state sequence associated with the model has the property that as time increases the state index increases (or stays the same), i.e., the states proceed from left to

right. Clearly the left-right type of HMM has the desirable property that it can readily model signals whose properties change over time e.g., speech. The fundamental property of all left-right HMMs is that the state transition coefficients have the property

$$a_{ij} = 0 \quad j < i \tag{4.35}$$

i.e., no transitions are allowed to states whose indices are lower than the current state. Furthermore, the initial state probabilities have the property

$$\pi_i = \begin{cases} 0, & i \neq 1 \\ 1, & i = 1 \end{cases} \tag{4.36}$$

since the state sequence must begin in state 1 (and end in state *N).*

Often, with left-right models, additional constraints are placed on the state transition coefficients to make sure that large changes in state indices do not occur;

hence a constraint of the form

$$a_{ij} = 0, \; j > i + \Delta \tag{4.37}$$

is often used. In particular, for the example of figure 4.4, the value of Δ is 1**,** i.e., no jumps of more than 1 states are allowed [29]. The form of the state transition matrix for the example of is thus:

$$\begin{bmatrix} a_{11} & a_{12} & 0 & 0 \\ 0 & a_{22} & a_{23} & 0 \\ 0 & 0 & a_{33} & a_{34} \\ 0 & 0 & 0 & a_{44} \end{bmatrix} \tag{4.38}$$

Although we have dichotomized HMMs into ergodic and left-right models, there are many possible variations and combinations possible. By way of example, figure 4.5 shows the forward and backward HMM.

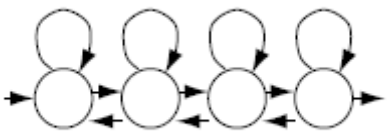

Figure 4.5: forward and backward HMM

4.7 Continuous Observation Densities in HMMs

All of our discussion, to this point, has considered only the case when the observations were characterized as discrete symbols chosen from a finite alphabet, and therefore we could use a

discrete probability density within each state of this model. The problem with this approach, at least for some applications, is that the observations are continuous signals (or vectors). Although it is possible to quantize such continuous signals via codebooks, etc., there might be serious degradation associated with such quantization. Hence it would be advantageous to be able to use HMMs with continuous observation densities.

In order to use a continuous observation density, some restrictions have to be placed on the form of the model probability density function (pdf) to insure that the parameters of the pdf can be re-estimated in a consistent way. The most general representation of the pdf, for which a re-estimation procedure has been formulated [33] [34], is a finite mixture of the form

$$b_j(O) = \sum_{m=1}^{M} c_{jm}\, \mathcal{M}\left[O, \mu_{jm}, \Sigma_{\mathrm{jm}}\right], \quad 1 \le j \le N \tag{4.39}$$

Where O is the vector being modeled, c_{jm} is the mixture coefficient for the mth mixture in state j and $\mathcal{M}$ is any log-concave or elliptically symmetric density [29] (e.g., Gaussian), with mean vector μ_{jm} and covariance matrix Σ_{jm} for the mth mixture component in state j. Usually a Gaussian density is used for $\mathcal{M}$.

In the case of a single Gaussian the re-estimation formulas are more closed to those obtained in the discrete case. In fact we have that the re-estimation formula for a_{ij} is identical to the one used for the discrete observation densities while the re-estimation formulas for the coefficient of the density i.e. $\mu_j, \Sigma_{\mathrm{j}}$ are of the form:

$$\mu_j = \frac{\sum_{t=1}^{T} \gamma_t(j) \cdot O_t}{\sum_{t=1}^{T} \gamma_t(j)} \tag{4.40}$$

$$\Sigma_{\mathrm{j}} = \frac{\sum_{t=1}^{T} \gamma_t(j) \cdot (O_t - \mu_j)(O_t - \mu_j)'}{\sum_{t=1}^{T} \gamma_t(j)} \tag{4.41}$$

4.8 Sound probability Models

Spectral features of a sound vary in time and it is this variation that gives a characteristic fingerprint for classification. Sound recognition models divide the sound feature space into a number of states and each state is defined by a continuous probability distribution [11].

The states are labeled 1 to *k*, and sound is indexed by the most probable sequence of states for a given sound model. Figure 4.6 shows the model of the Anchorwoman speech that I have

obtained using an HMM with 4 states and in order to plot it I have used a basis of only one dimension. The sound trajectory in the space corresponds to a sequence of spectral frames projected into an $AudioSpectrumProjection$ descriptor.

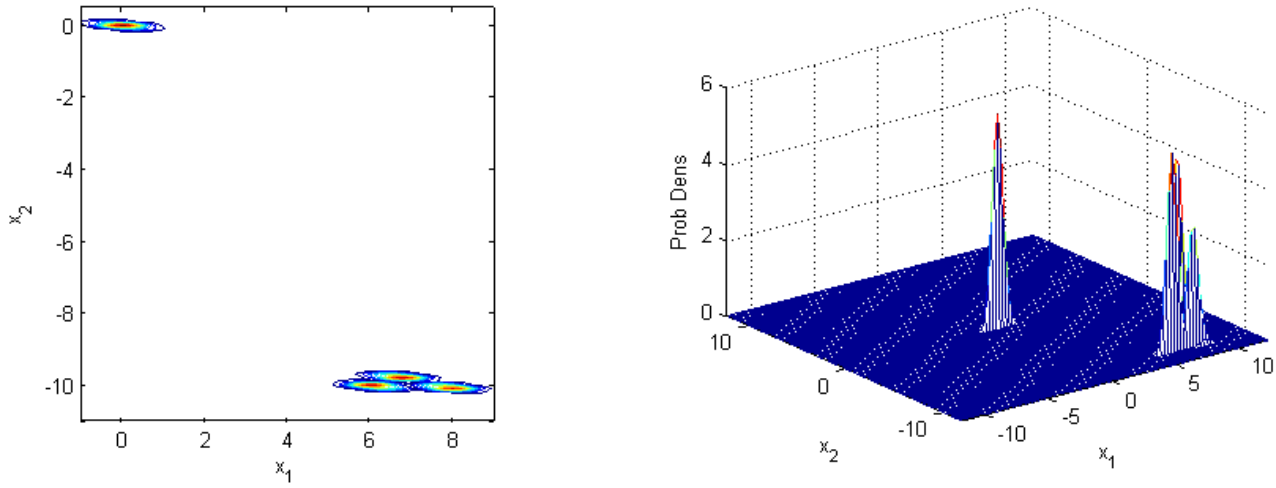

Figure 4.6: Four probability model states in a two-dimensional vector space of the class Anchorwoman.

4.9 Train the HMM using the Experimental Model

In order to train a statistical model on the basis projection features for each audio class I have used and adapted to our case the matlab code that are in the MPEG-7 Experimental Model [15].

The code for training the HMM is $SoundClassificationModelDS$ it requires as input the root directory for training data that contains sub-directory of individual class data, the number of initial model states for each HMM, and the number of basis functions to extract.

During training, the parameters for each state of an audio model are estimated by analyzing the feature vectors of the training set. Each state represents a similarly behaving portion of an observable symbol sequence process. At each instant in time, the observable symbol in each sequence either stays at the same state or moves to another state depending on a set of state transition probabilities. Different state transitions may be more important for modeling different kinds of data.

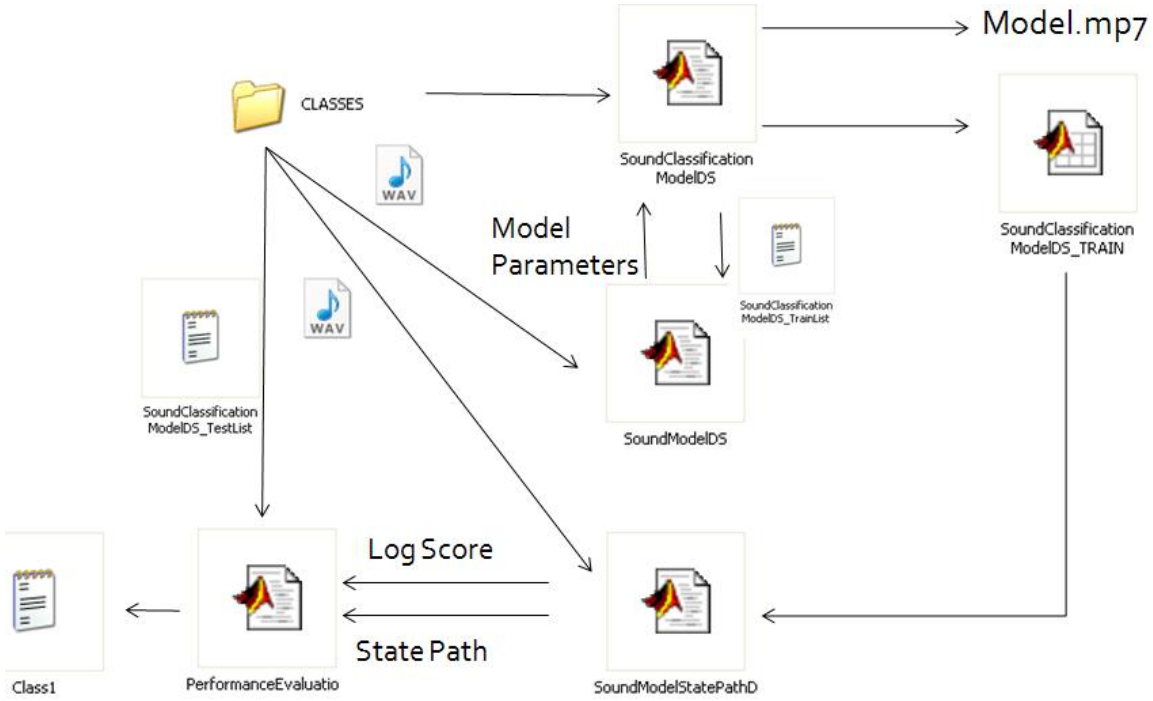

Figure 4.7: The classification system based on the Experimental Model

The *SoundClassificationModelDS* code at beginning verifies if in the folder of each class with the wav segments there are the text-files with the list of the files respectively to train the model and to test it. If there aren't it create the two file text splitting randomly the available data in 70% for training and 30% for testing the model. The train list with the number of states, the dimension of the feature vector and other optional parameters are then passed to the code *SoundModelDS* as we can see in the schema in figure 4.7.

This code for each file in the list computes the ASE as describe in Chapter 2 and concatenates all these matrices in a unique matrix which is then pass to codes that computes the basis and the projection of the feature vector in the lower dimensional subspace. This projection is then used to train the HMM.

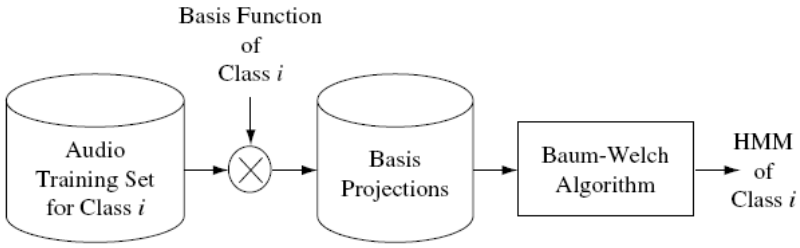

Figure 4.8: HMM training for a given sound class i

Figure 4.8 illustrates the training process of an HMM for a given sound class i [11]. The HMM parameters are then obtained using the Baum–Welch algorithm. As described in the above paragraphs the procedure starts with random initial values for all of the parameters and optimizes the parameters by iterative re-estimation. Each iteration runs through the entire set of training data in a process that is repeated until the model converges to satisfactory values. Often parameters converge after three or four training iterations. With the Baum–Welch re-estimation training patterns, one HMM is computed for each class of sound that captures the statistically most regular features of the sound feature space [26].

For defining state densities is used the multidimensional Gaussian distribution. Gaussian distributions are parameterized by a $1 \times n$ vector of means $\boldsymbol{\mu}$ and an $n \times n$ covariance matrix $\boldsymbol{\Sigma}$, where n is the number of features (columns) in the observation vectors. The $GaussianDistribution\ DS$ stores the *inverse* covariance matrix and the *determinant* of the covariance matrix along with the vector of means for each state [11]. Therefore, the expression for calculating probabilities for a random vector $\boldsymbol{x}$, given a mean vector $\boldsymbol{\mu}$, covariance matrix inverse and the covariance matrix determinant is:

$$f_x(\boldsymbol{x}) = \frac{1}{\left(\sqrt[2]{2\pi}\right)^2 \sqrt[2]{|\boldsymbol{\Sigma}|}} e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})} \tag{4.42}$$

In the end the $SoundModelDS$ returns to the $SoundClassificationModel$ the following parameters for each model:

- T=State transition matrix;
- S=Initial State probability vector;
- M=Stacked means matrix (1 vector per row)
- C=Stacked inverse covariances
- V=$AudioSpectrumBasis$ vectors
- $maxEnv$= scaling parameter for model decoding
- P=Training cycle likelihoods

After that the $SoundClassificationModel$ stores them in a struct variable and then saves this variable in a mat file which in figure 4.7 is called *SoundClassificationModelDS_train*.

The training procedure ends when the Markov model parameters and the $AudioSpectrumBasis$ functions are written in the $SoundModelClassification$ description

scheme. The $SoundModelClassification$ description scheme is derived from the $ContinuousHiddenMarkovModel$ DS defined in MDS (Model.mp7 in figure 4.7).

4.10 Automatic Audio Classification using Experimental Model

Automatic audio classification finds the best-match class for an input sound by presenting it to a number of HMMs and selecting the model with the highest *likelihood* score. A combination of HMMs used in this way is called a *classifier*. To build a classifier, a set of individual $SoundModel\ DSs$ are trained, one for each class in a classification scheme, and combined into a $SoundClassificationModel$ as I have described in the previous paragraph. Given a query sound, the spectrum envelope is extracted and the result is presented to each sound model.

The spectrum is projected against the model's basis functions producing a low-dimensional feature representation represented by the $AudioSpectrumProjection$ descriptor. The *Viterbi* algorithm is then used to compute the $SoundModelStatePath$ and likelihood score and the HMM with the maximum likelihood score is selected as the representative class for the sound. The $SoundModelStatePath$ descriptor contains the dynamic state path of a sound through a HMM model.

Figure 4.9 shows a spectrogram of an anchorwoman speech with the state path through the an Anchorwoman HMM of four states.

The state path is an important method of description since it describes the evolution of a sound with respect to physical states.

The state path shown in the figure indicates physical states for the Anchorwoman speech; there are clearly delimited the state of unvoiced and voiced. This is true of most sound classes; the individual states within the class can be inspected via the state path representation and a useful semantic interpretation can often be inferred.

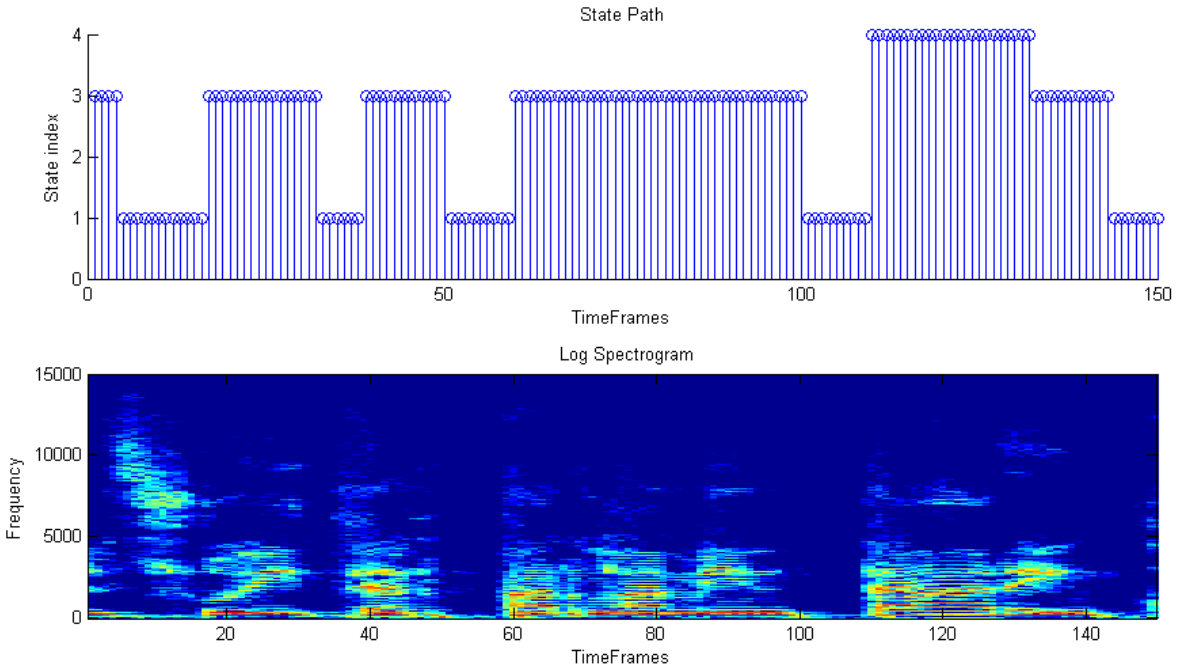

Figure 4.9 Anchorwoman speech spectrogram and the state path through Anchorwoman continuous hidden Markov model

Figure 4.10 illustrates the method for automatic audio classification descript above [11].

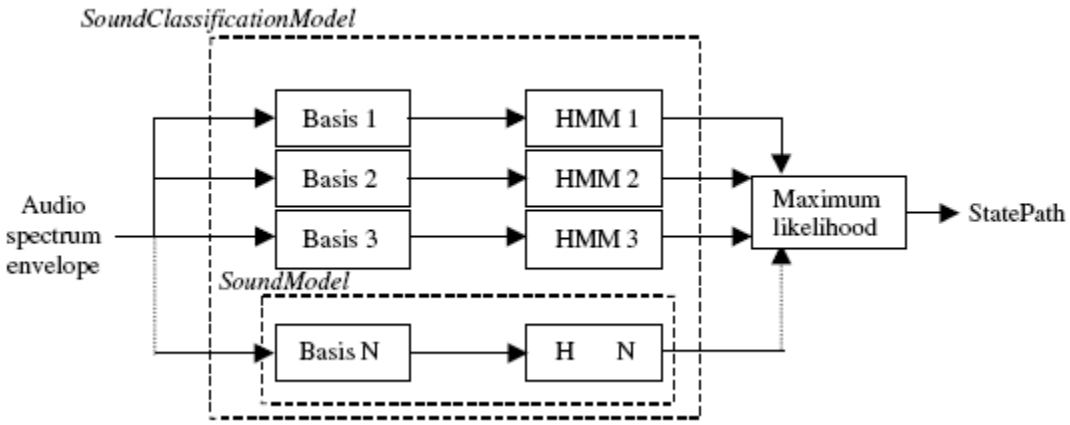

Figure 4.10: Automatic audio classification method

4.11 Testing the Model

In order to test the Model I have written a matlab code $Performance Evaluation$ which has as input the folder that contains the wav segments of the class that we want test (the same folder used to create the model as we can see in figure 4.7).

At first the code load the .mat file that contains, all the HMM and the basis functions, for each class. Then it reads the text file with the list of the wav segments for testing the model, and

presents each segment in the list to every sounds model using the matlab code $SoundModelStatePathD$ that is in the MPEG-7 Experimemetal Model [15]. At the beginning this code computes the ASE and then projects it in the lower dimension feature space using the basis function of the model stored in the mat file. This projection is passed with the HMM parameters to the function $viterbi$ which computes the *Viterbi Algorithm* as described in the paragraph 4.4 and returns the path and the log likelihood score of the sequence.

I my code I store this score for each model and I choose the model that produces the maximum score.

In the end I store the results of every wav segments in the list in a text file, and I use it to build the confusion matrix.

4.12 Confusion Matrix

A confusion matrix contains information about actual and predicted classifications done by a classification system. Performance of such systems is commonly evaluated using the data in the matrix. The following table shows the confusion matrix for a two class classifier.

The entries in the confusion matrix have the following meaning in the context of our study:

- *a* is the percentage of **correct** predictions that an instance is **negative**,
- *b* is the percentage of **incorrect** predictions that an instance is **positive**,
- *c* is the percentage of **incorrect** of predictions that an instance **negative**, and
- *d* is the percentage of **correct** predictions that an instance is **positive**.

		Predicted	
		Negative	Positive
Actual	Negative	***a***	**b**
	Positive	**c**	**d**

Table 4.1: Confusion Matrix

CHAPTER 5
SIMULATIONS RESULTS AND DISCUSSION

In order to train and test sounds model a data base of speech and music was collected. The speech contains a wide range of different radio newscast. The samples are chosen to reflect many different kinds of typical speech, ranging from anchor speakers in almost perfect conditions to conversations between multiple speakers. Also, narrow-band telephone interviews are present.

.

Number	GR	Date	Time	Duration (min)	Anchor
1	RAI GR1	10/11/2007	13.00	30	Anchorman1: G. Trevisi Anchorman2: A. Biciocchi
2	RAI GR1	11/11/2007	13.00	30	Anchorman1: G. Trevisi Anchorman2: A. Biciocchi
3	RAI GR1	17/11/2007	13.00	30	Anchorman1: G. Trevisi Anchorman2: A. Biciocchi
4	RAI GR1	26/11/2007	13.00	30	Anchorman1: G. Trevisi Anchorman2: A. Biciocchi
5	RAI GR1	02/12/2007	13.00	30	Anchorman1: G. Trevisi Anchorman2: A. Biciocchi
6	RAI GR2	05/11/2007	19.30	28	Anchorwoman1: L. Scardini Anchorwoman2: V.Montanari
7	RAI GR2	08/11/2007	19.30	28	Anchorwoman1: L. Scardini Anchorwoman2: A. Fiori
8	RAI GR2	12/11/2007	19.30	28	Anchorwoman1: L. Scardini Anchorwoman2: A. Fiori
9	RAI GR2	14/11/2007	19.30	28	Anchorwoman1: L. Scardini Anchorwoman2: A. Fiori
10	RAI GR2	21/11/2007	19.30	28	Anchorwoman1: L. Scardini Anchorwoman2: A. Fiori
11	RAI GR3	06/11/2007	8.45	20	Anchorwoman: A. Pizzato
12	RAI GR3	07/11/2007	8.45	20	Anchorwoman: A. Pizzato
13	RAI GR3	08/11/2007	8.45	20	Anchorwoman: A. Pizzato
14	RAI GR3	09/11/2007	8.45	20	Anchorwoman: A. Pizzato
15	RAI GR3	10/11/2007	8.45	20	Anchorwoman: A. Pizzato

Table 5.1: List of the GR wav files present in the data base

Some of the speech clips contain speech from reporters speaking from noisy environments. 15 Giornali Radio RAI (GRR) was recorded from the RAI web page http://www.radio.rai.it/grr/ with Freecoder, a software tools for recording internet audio.

The sources of the clips are listed in table 4.1. The clips were collected in November and December 2007. The GRR are chosen in order to have the same anchor for the same GRR
The GRRs are then segmented manually, using WaveSurfer [35], into different classes.

5.1 WaveSurfer Tool for the Manual Segmentation

WaveSurfer is an Open Source tool for sound visualization and manipulation [35]. WaveSurfer has a simple and logical user interface that provides functionality in an intuitive way and which can be adapted to different tasks. It can be used as a stand-alone tool for a wide range of tasks in speech research and education. Typical applications are speech/sound analysis and sound annotation/transcription. WaveSurfer can also serve as a platform for more advanced/specialized applications. This is accomplished either through extending the WaveSurfer application with new custom plug-ins or by embedding WaveSurfer visualization components in other applications.
As shown in Figure 4.1 when WaveSurfer is first started, it contains an empty sound. You can load a sound file from disk.

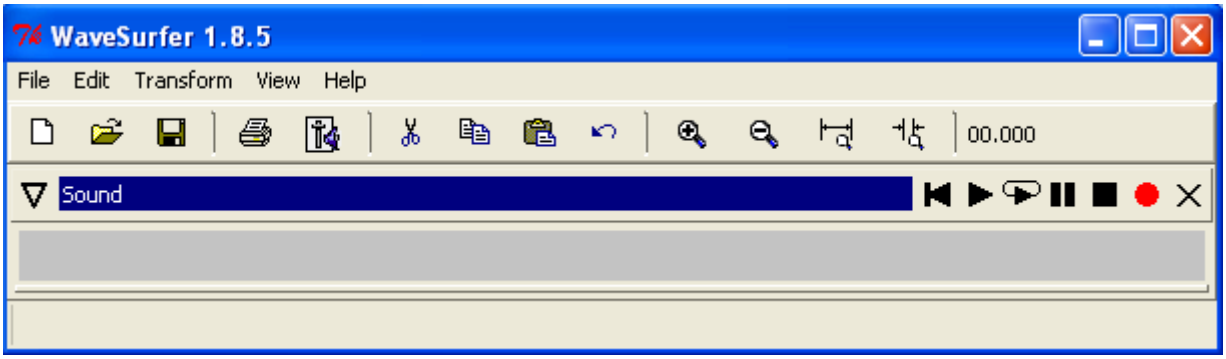

Figure 5.1: WaveSurfer Interface

To allow for different sophisticated tasks WaveSurfer gives the possibility of adding panes. A pane is a window stacked on top of the WaveBar that can contain for example a waveform, a spectrogram, a pitch-curve, a time axis or a transcription or something else. Unlike the WaveBar, a pane will not necessarily display the whole sound. Rather it will display a portion of the sound that is specified in the WaveBar. Think of the WaveBar as an overview and the pane as a variable magnifying glass.
WaveSurfer can read a number of sound file formats including WAV, AU, AIFF, MP3, CSL, and SD. It can also save files in several formats, including WAV, AU, and AIFF. There are separate plug-ins to handle Ogg/Vorbis and NIST/Sphere files. WaveSurfer can be used to

visualize and analyze sound in several ways. The standard analysis plug-in can display Waveform, Spectrogram, Pitch, Power or Formant panes.

WaveSurfer has many facilities for transcribing sound files. Transcription is handled by a dedicated plug-in and it's associated pane type. In the Figure 4.2 is shown WaveSurfer interface when the the configuration Transcription is chosen.

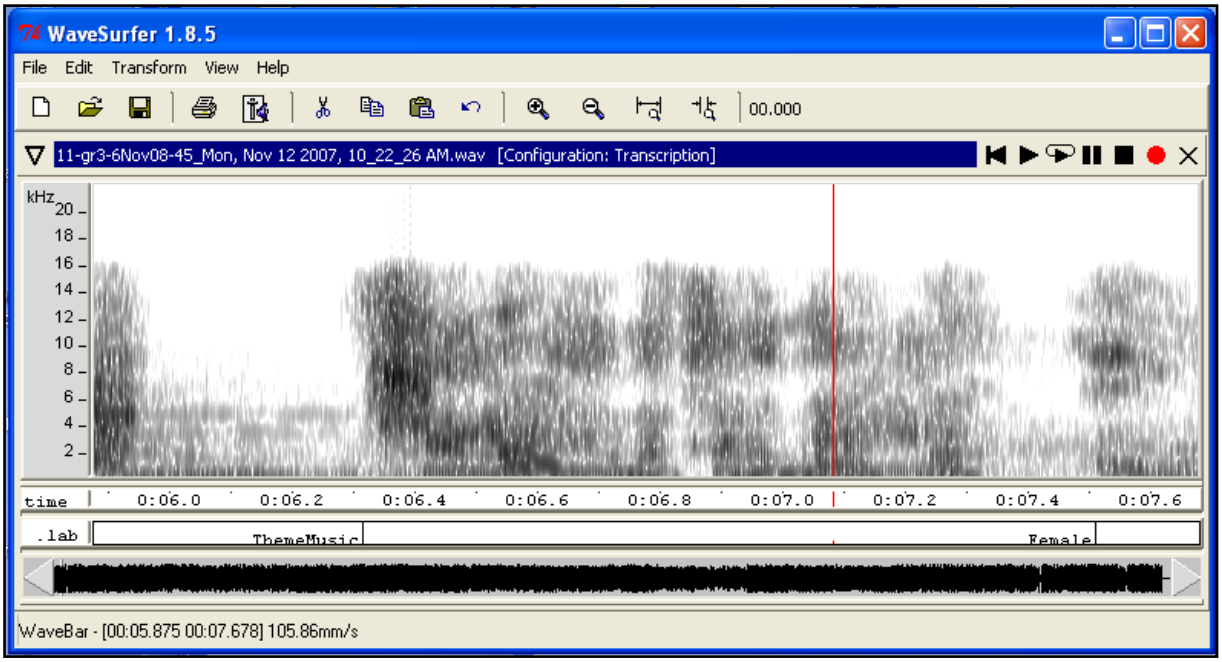

Figure 5.2: WaveSurfer interface when the transcription configuration is chosen.

The properties-dialog can be used to specify which label file that should be displayed in a transcription pane. Unicode characters are supported in order to keep the binary versions small. The transcription plug-in is used in combination with format handler plug-ins which handles the conversion between file formats and the internal format used by the transcription plug-in. The standard popup menu has additional entries for transcription panes. *Popup | Load Transcription* and *Popup | Save Transcription* are used to load and save transcription files.

In the Listing 4.1 is shown an example of a lab file that WaveSurfer has as output.

WaveSurfer allows splitting sound on labels, but every label should have a different name so we wrote a matlab code that modify the lab file, produced with WaveSurfer, adding an increasing number to each name label.

```
0.0000000 0.4425000 Silence
0.4425000 0.8650000 Female
0.8650000 1.1450000 Silence
1.1450000 1.3575000 Female
1.3575000 1.5150000 Silence
1.5150000 1.6700000 Female
1.6700000 1.9950000 Silence
1.9950000 2.2525000 Female
2.2525000 2.4200000 Silence
2.4200000 3.5900000 Female
3.5900000 4.1650000 Silence
4.1650000 6.3150000 ThemeMusic
6.3150000 7.5100000 Female
7.5100000 11.3925000 Male+Music
```

Listing 5.1: Piece of a lab file

5.2 Data Base Description

In order to have an idea of the amount of the seconds of each class I wrote a matlab code that reads the file lab of WaveSurfer and analysing it returns for each class the total seconds.
In the following tables are reported the amount of seconds for each class.

CLASSES	DURATION (h:m:s:ms)
Speech	4:48:27:231
Music	0:16:12:139
Silence	0:11:23:472
Other	1:17:9:121
Total	6:33:11:963

Table 5.2: Duration of the classes: Speech, Music, Silence, Other

CLASSES	DURATION (h:m:s:ms)
Male	3:4:22:348
Female	1:44:4:883
Total	4:48:27:231

Table 5.3: Duration of the subclasses Male and Female of the class Speech

CLASSES	DURATION (h:m:s:ms)
Anchorman	0:20:15:627
Male	1:37:7:879
MaleSpot	0:5:20:382
MaleTel	1:1:38:459
Total	3:4:22:348

Table 5.4: Duration of the subclasses: Anchorman, Male, MaleSpot, MaleTel of the class Male

CLASSES	DURATION (h:m:s:ms)
Anchorwoman	0:40:21:574
Female	0:56:51:503
FemaleSpot	0:3:41:117
FemaleTel	0:3:10:688
Total	1:44:4:883

Table 5.5 Total duration of the wav files of the subclasses: Anchorwoman, Female, FemaleSpot, FemaleTel of the class Female

CLASSES	DURATION (h:m:s:ms)
Anchorman1	0:10:8:774
Anchorman2	0:10:6:853
Total	0:20:15:627

Table 5.6 Total duration of the wav files of the subclasses: Anchorman1, Anchorman2 of the class Anchorman.

CLASSES	DURATION (h:m:s:ms)
Anchorwoman1	0:13:36:137
Anchorwoman2	0:8:54:940
Anchorwoman3	0:1:38:625
Anchorwoman	0:16:11:873
Total	0:40:21:574

Table 5.7 Total duration of the wav files of the subclasses: Anchoworman1, Anchoworman2, Anchorwoman3, and Anchorwoman of the class Anchorwoman.

CLASSES	DURATIONS (h:m:s:ms)
OtherMusic	0:3:29:218
ThemeMusic	0.12:42:921
Total	0:16:12:139

Table 5.8: Total duration of the wav files of the subclasses: OtherMusic, ThemeMusic of the class Music.

CLASSES	DURATIONS (h:m:s:ms)
Speech+Music	0:56:37:544
Speech+Music	0:18:20:299
Environmental	0:2:11:278
Total	1:17:9:121

Table 5.9: Total duration of the wav file of the subclasses: Speech+Music, Speech+Noise, Environmental of the class Other.

In order to obtain a compact view of the available audio materials for each class the matlab code also returns the pie diagrams, reported in figure 5.3 and 5.4, that represent statistical information of the data base created.

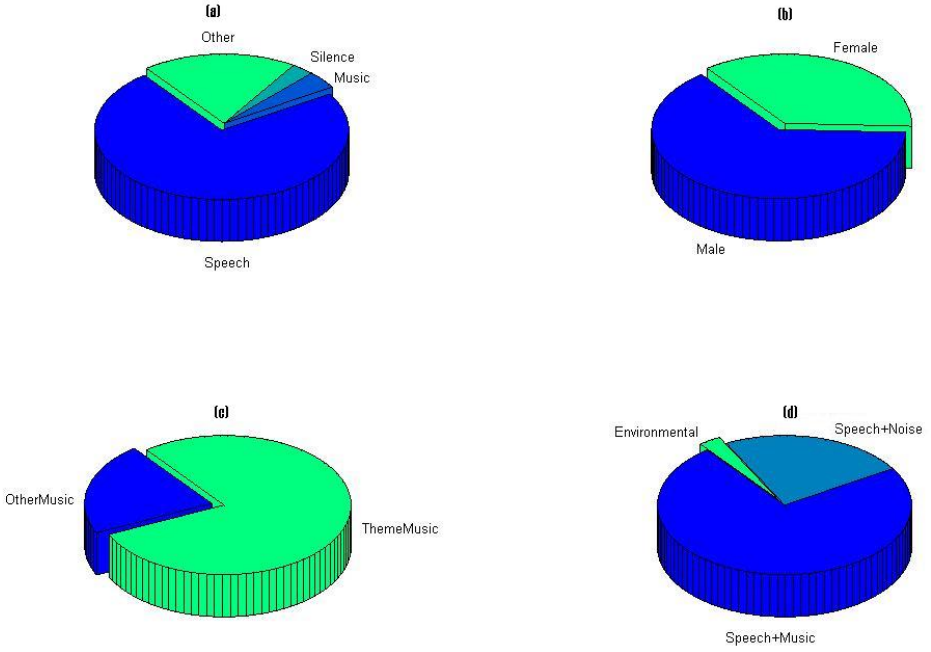

Figure 5.3: a) Pie diagram of the classes: Speech, Music, Other, Silence; b) Pie diagram of the subclasses Male, Female of the class Speech; c) Pie diagram of the subclasses OtherMusic, ThemeMusic of the class Music; d) Pie diagram of the subclasses Speech+Noise ,Speech+Music, Enviromental of the class Other.

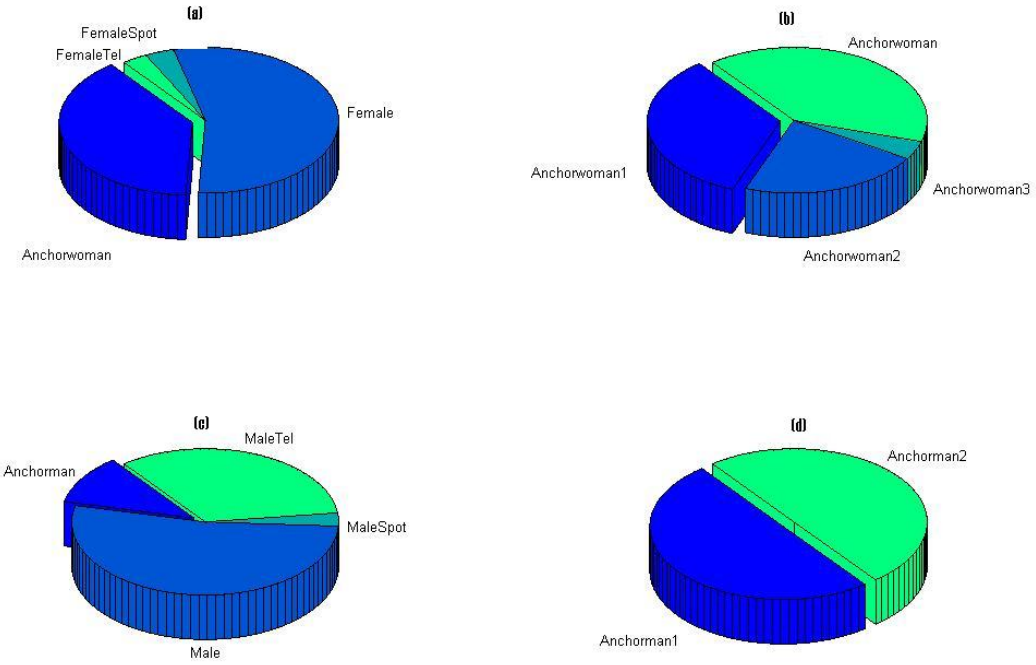

Figure 5.4: a) Pie diagram of the subclasses: Anchorwoman, Female, FemaleSpot, FemaleTel of the class Female; b) Pie diagram of the subclasses Anchorwoman1, Anchorwoman2, Anchorwoman3, Anchorwoman of the class Anchorwoman; c) Pie diagram of the subclasses Anchorman, Male ,MaleSpot, MaleTel of the class Male; d) Pie diagram of the subclasses Anchorman1, Anchorman2, of the class Anchorman.

Analyzing the segments obtained using Wavesurfer I noted that they present a large variation in duration from few ms to half minute so I have written a matlab code which analyzes the lab file and when it meets a segments which have a duration major than 15 s it discards the first 5 s and stores as good segment for training only the next 10 seconds.

After that I have create for each radio newscast a folder for each class which contain the audio segments compute as described earlier.

In order to test the classification system I trained some experiments that I will report in the following. In all these experiments I have trained for each class an HMM with 10 states and a feature dimension of 23 as made in [36] those values are good but aren't optimized for the specific case. I performed an optimization step only for the final classification model.

5.3 Experiment 1: *Gender identification*

The target of this first experiment is the gender identification, so in order to realize it I created 2 sound classes Male and Female using the audio segments respectively of the class

Anchorman and of the class anchorwoman. The numbers of the training testing sound clips for the two classes are reported in the table 5.10.

	Training	Test
Male	92	38
Female	100	43

Table 5.10: Number of the wav files used for train and test the model

In the table 5.11 is reported the confusion matrix of this experiment which indicates that every segments that belongs to the class Female are classified as female and the same for the segments that belong to the class male.

	Female	Male
Female	100	0
Male	0	100

Table 5.11: Confusion Matrix

5.4 Experiment 2: *Discrimination between Speech and Music Sound*

Automatic discrimination of speech and music sound is important in many multimedia applications, i.e. (1) radio receivers for the automatic monitoring of the audio content of FM radio channels, (2) disabling the speech recognizer during the non-speech portion of the audio stream in automatic speech recognition of broadcast news, (3) distinguishing speech from music for low-bit-rate audio coding.

Before to reach good results I have made some experiment and I have noted that the speech model is well formed only if I use for train the HMM only one kind the speech for example the results reported below are obtained using the speech of the Anchor.

During those experiments I have also noted that if we use for the testing of the model very short audio clip ranging from few ms to 1.5 s the test fails. So in this experiment instead of using the automatic generation of the text files with the lists of the audio segments for training and testing the model (see paragraph 4.9) I create manually those 2 lists putting the shorter segments in the train list and the other in the test list.

The numbers of the training testing sound clips for the two classes are reported in the table 5.12,

	Train	Test
Music	65	20
Speech	60	40

Table 5.12: Number of the wav files used for train and test the model

and in the table 5.13 there is the confusion matrix of this experiment which shows that only the 15% of 20 music segments are wrongly assigned to the class speech.

	Music	Speech
Music	85	15
Speech	0	100

Table 5.13: Confusion Matrix

To verify that the two models are robust I test them with wav files that belong to the classes reported in following table:

	Test
FemaleSpot	33
MaleSpot	52
Speech+Music	39
SpeechMale	68
SpeechFemale	58

Table 5.14: Number of the wav files used for test the models

and in the following table I report the results of this test:

	Music	Speech
FemaleSpot	6	94
MaleSpot	0	100
Speech+Music	10	90
SpeechMale	0	100
SpeechFemale	0	100

Table 5.15: Results of the tests

We can note that the two models are well formed, because every kind of speech are classified as *Speech*. I have test the two models also with segments that belong to the class *Speech+Music* and we can note that this segments are classified as *Speech*, perhaps because in a radio newscast the piece of sound, where we have speech over music the speech component is prevalent. This observation is most important for the design of the final classification model.

5.5 Experiment 3: *Speech-Telephone Speech*

The target of this experiment is distinguishing speakers speaking in an ideal environment from speakers that speak on a non ideal telephone line.

In the table 5.16 I report the number of segments used in this experiment in order to train and to test the model.

	Train	Test
Speech	37	30
Telephone Speech	60	40

Table 5.16: Number of the wav files used for train and test the model

In the table 5.17 is reported the confusion matrix of this experiment which indicates that every of the 70 segments using for testing the two models are correctly assigned to the respective classes.

	Speech	Telephone Speech
Speech	100	0
Telephone Speech	0	100

Table 5.17: Confusion Matrix

5.6 Experiment 4: *Speaker Identification*

Speaker recognition attempts to recognize a person from a spoken phrase, useful for radio and TV broadcast indexing.

In order to perform this experiment I created two models for the two anchormans of the GR1 and other two models for the anchorwoman of the GR2 and of the GR3.

In the following table are reported the numbers of the wav files used for train and test the model.

	Training	Test
AnchormanGR1_1	53	30
AnchormanGR1_2	35	20
AnchorwomanGR3	31	15
AnchorwomanGR2	56	20

Table 5.18: Number of the wav files used for train and test the model

In the table 5.19 is reported the confusion matrix of this experiment. We can note that the results are satisfactory because also the gender identification is completely reaches. However we have to consider that we are in an ideal case in which we have a model for each speaker, and this hypothesis are not verify in general when we have to analyze a generic piece of a radio newscast. In fact in this case how we will see later the results are worst.

	AnchormanGR1_1	AnchormanGR1_2	AnchorwomanGR2	AnchorwomanGR3
AnchormanGR1_1	90	10	0	0
AnchormanGR1_2	5	95	0	0
AnchorwomanGR3	0	0	100	0
AnchorwomanGR2	0	0	5	95

Table 5.19: Confusion Matrix

5.7 Experiment 5: *Hierarchical classification model*

The purpose of this experiment is the design of a hierarchical classification model that can be used to classifier the segments extract from an entire radio newscast. In the following I report the analysis of the four models that belongs to the final classification model. For the creation of the following three levels I have made some experiments to investigate the correlation between different classes. In those experiment if two classes are confused each other then I joint this classes and I move their classification in the layer below.

5.7.1 Parameters Selections

The other advantage of using a hierarchical approach is that in this way is possible design each layer in an independent way, in fact in the experiments that I have made to individuate the class of each layer I used HMMs with 10 states and a feature dimension of 23, and then separately for each layer I performed an optimization step in which I chosen the optimal feature dimension and the optimal number of states.

The parameter with the most drastic impact turned out to be the feature vector dimension, in fact if this is too small, the basis vectors reduce the data too much, and the HMM do not receive enough information. However, if the feature vector dimension is too large, then the extra information extracted is not very important and is better ignored. The optimal value of this parameter is different for different experiments [36]. One needs to be careful about choosing this and to test empirically to find the optimal value. As said before the other

important parameter is the number of states, so after the setting of the dimension of the feature vector should be chosen the number of the states, always in an empirical way.

5.7.2 First Layer: *Sound*

In this layer in addition to the class *Speech* and the class *Music*, I considerate also the class *Telephone Speech*, because when I consider a system with only the classes *Speech* and *Music*, as in the Experiment 2, and I test it with wav files that belong to the class *Telephone Speech* these are classified as *Music* and isn't reasonable joint the two classes.

In the following table are reported the numbers of the wav files used for train and test the models.

	Train	Test
Music	65	20
Speech	60	40
Telephone Speech	37	30

Table 5.20: Number of the wav files used for train and test the model

The results of this experiment are reported in the confusion matrix in the table 5.21 in which we have a recognition rate of 95%.

	Music	Speech	Telephone Speech
Music	85	15	0
Speech	0	100	0
Telephone Speech	0	0	100

Table 5.21: Confusion Matrix

Then I have verified that all kind of speech segments are classified as speech testing the models with wav files that belongs to the classes reported in the follow table:

	Test
FemaleSpot	33
MaleSpot	52
Speech+Music	39
Speech Male	68
Speech Female	58
Telephone Female	10

Table 5.22: Number of the wav files used for test the models

As we can see in the table 5.21 in this test I considerate also the class *Telephone Speech*. The reason of this choice is that in our database we have very few segments which belong to this

class, so I have verify, and the results in the table 5.22 confirms this, that the class *Telephone Speech* obtained using the segments of the class *Telephone Male* is well formed also for the segments in which a female speaker speaks on an non ideal telephonic line.

	Music	Speech	Telephone Speech
FemaleSpot	6	94	0
MaleSpot	0	100	0
Speech+Music	10	90	0
SpeechMale	0	100	0
SpeechFemale	0	100	0
Telephone Female	0	10	90

Table 5.23: Results of the tests

The results reported in the table 5.23 confirm that the three models are well formed and so I performed the optimization step choosing the optimal value for the feature dimension maintaining fix the number of states to 10. In figure are reported the values obtained.

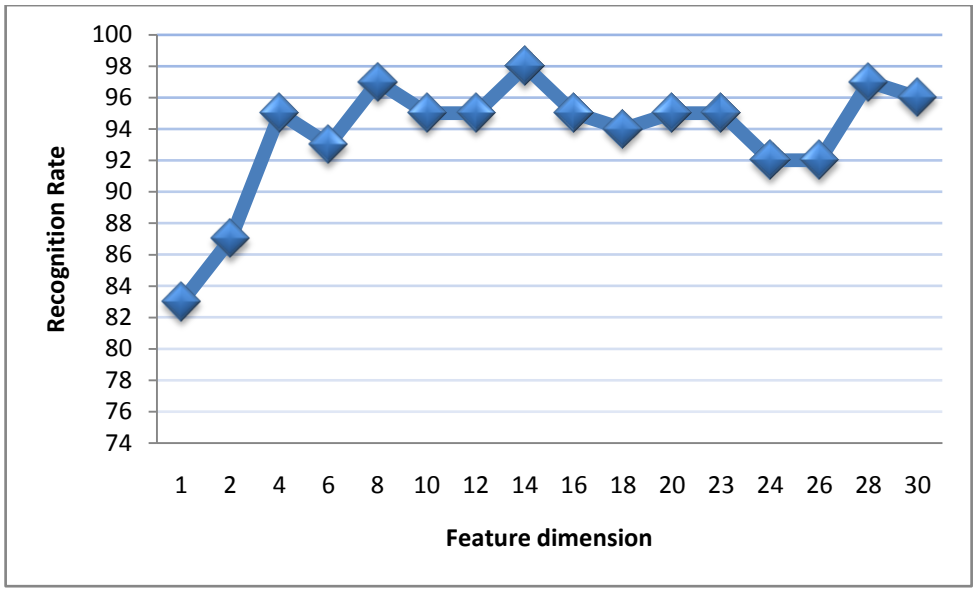

Figure 5.5 Recognition rate vs feature dimension

From figure 5.5 we can see that the optimal feature vector dimension is 14 to which corresponds a recognition rate of 98%. Hence with this value I tuned on the number of states and I have obtained the results reported in figure 5.6

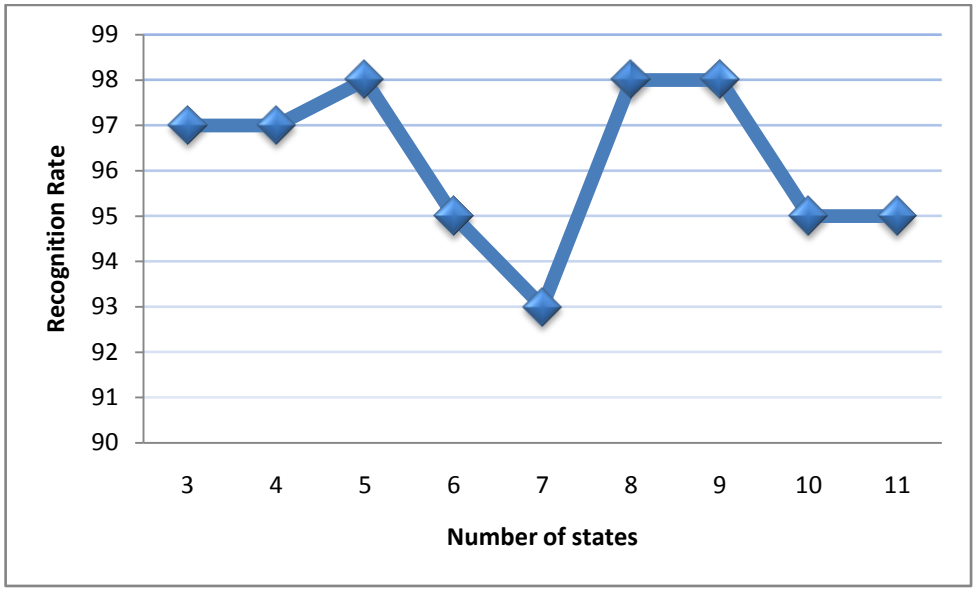

Figure 5.6 Recognition rate vs Number of states

Observing the graph in figure 5.6 we can see that varying the number of states we cannot improve the recognition rate but we can reduce the number of states from 10 to 5, maintaining the same performance.

5.7.3 Second Layer: *Speech*

In this layer I have individuated five classes: the classes *FemaleSpot* and *MaleSpot* that are composed by segments in which a female or a male voice speaks in a spot, the class *Speech+Music* in which I have put segments with speech over music, and in the end the two classes *Anchorman* and *Anchorwoman* that are composed by the voices of the anchors. I am not able to obtain a model whit the classes *Other Male* and *Other Female* perhaps because to this model belong a lot of kind of speech.

	Train	Test
FemaleSpot	23	10
MaleSpot	32	20
Speech+Music	29	20
SpeechFemale	55	30
SpeechMale	60	30

Table 5.24: Number of the wav files used for train and test the model

In the table 5.25 are reported the results obtained and analyzing it we can see that in this experiment we have reached a recognition rate of 86%.

	FemaleSpot	MaleSpot	Speech+Music	Anchorwoman	Anchorman
FemaleSpot	80	0	10	10	0
MaleSpot	5	90	5	0	0
Speech+Music	15	10	70	5	0
Anchorwoman	0	0	0	100	0
Anchorman	0	10	0	0	90

Table 5.25: Confusion Matrix

The results are satisfactory, but in this way we don't classify the segments that belong to the classes *Other Male* and *Other Female,* so I tested this model with segments that belong to these classes.

	Test
Other Female	83
Other Male	75

Table 5.26: Number of the wav files used for test the models

	FemaleSpot	MaleSpot	Speech+Music	Anchorwoman	Anchorman
Other Female	17	7	3	72	0
Other Male	9	75	0	6	10

Table 5.27: Results of the tests

The results reported in the table 5.27 show that the 72% of 83 wav files that belong to the class *OtherFemale* are classified as Anchorwoman so we can consider a unique class *SpeechFemale.* The 75% of the 75 segments that belong to the class *OtherMale* are instead classify as *MaleSpot,* so I have made an experiment with only the two classes *OtherMale* and *MaleSpot,* but I was not able to generate a well formed model with the segments that composed our data base. For this reason in the final classification I haven't made a distinction between the two classes.

From the table 5.27 we can also note that only 10% of the segments of the class *OtherMale* are classify as *Anchorman* perhaps because we have over trained this classes. This can be seen as an advantage because in this way in the third level we can easily distinguish the two anchormans. After having individuate the classes I searched to improve the performance with the optimization of the parameters starting from the feature dimension remaining fix to 10 the number of states and I obtain the graph reported in figure 5.7

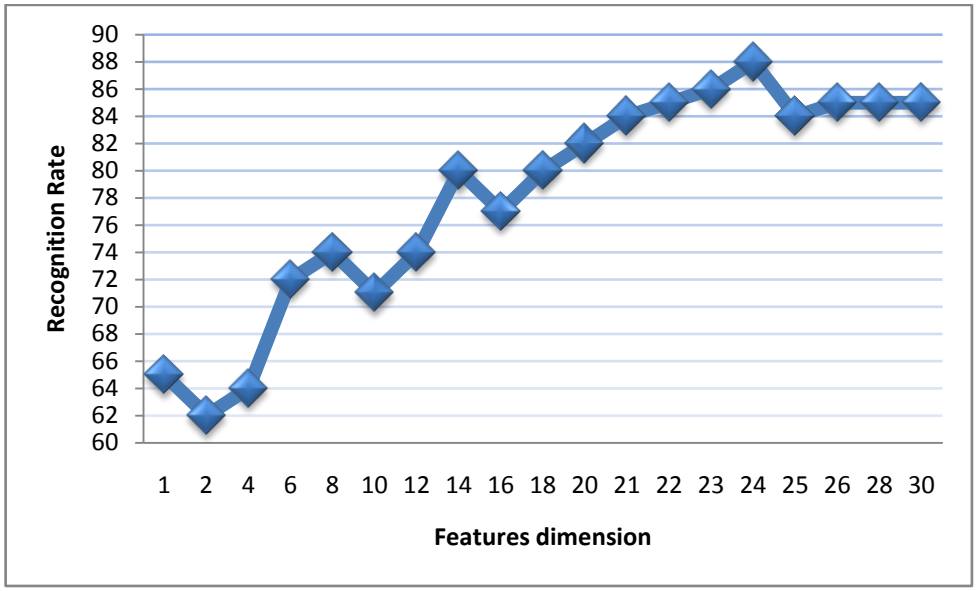

Figure 5.7 Recognition rate vs. feature dimension

Observing the graph we can see that the best recognition rate 88% is reached with a feature dimension of 24. Hence fixing the feature dimension to 88% and varying the number of states I have obtained the following graph:

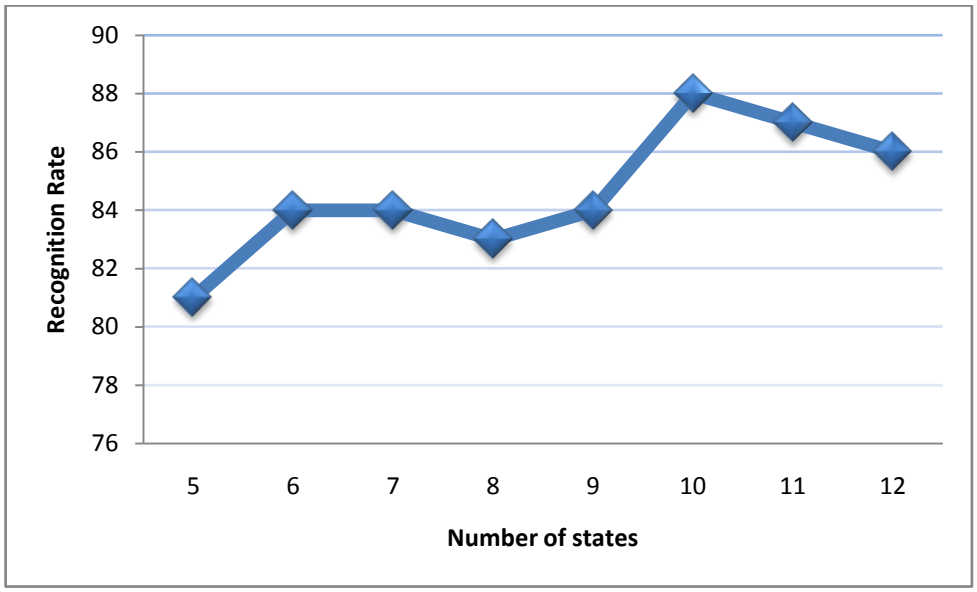

Figure 5.8 Recognition rate vs. number of states

Also in this case we couldn't improve the recognition rate varying the number of states, so the optimal value to this parameter remains 10.

5.7.4 Third Layer: *Anchorman Identification*

In the previous layer we are able to distinguish piece of GR in which speaks generic male voice, from piece of GR in which speaks an anchorman, so in this layer I performed the anchorman identification

	Training	Test
Anchorman_1GR1	46	30
Anchorman_2GR1	30	25

Table 5.28: Number of the wav files used for train and test the model

This experiment is very similar to the experiment 4, and also the results, which are in the table 5.29, are quite the same with a recognition rate of 93%.

	Anchorman_1GR1	Anchorman_2GR1
Anchorman_1GR1	90	10
Anchorman_2GR1	4	96

Table 5.29: Confusion Matrix

Also for this simple case with only two classes I searched the optimal parameters for the model and in the following graph I reported the results obtained in the case of the feature dimension:

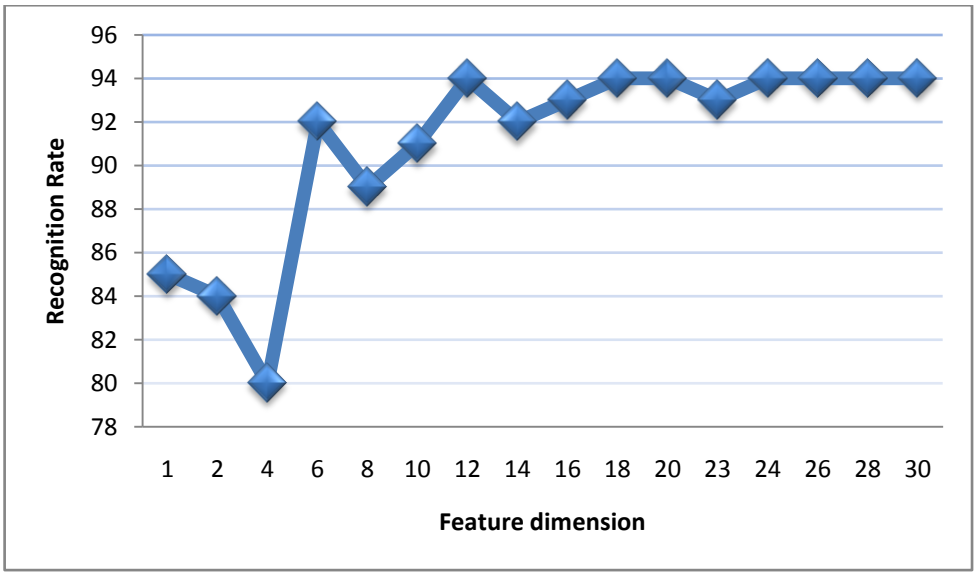

Figure 5.9: Recognition rate vs. feature dimension

The best recognition rate is 94% which is reached when we have a feature dimension of 12. And observing the graph in figure 5.10 we can see that also varying the number of the states 94% remains the highest value.

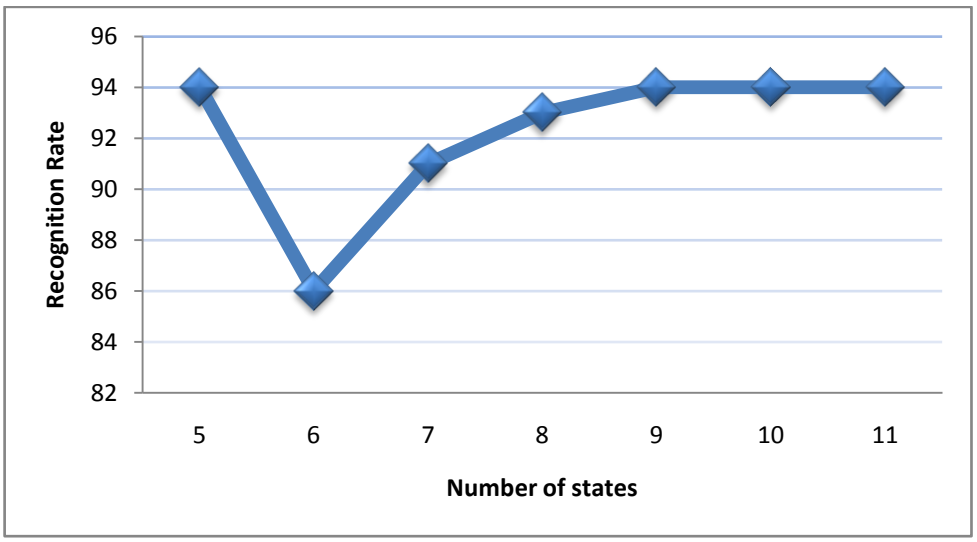

Figure 5.10: Recognition rate vs. number of states

5.7.5 Third Layer: *Anchorwoman GR3 Identification*

Differently to the anchorman identification in this case we have to consider also the class *Other Female* and so we haven't a model for each speaker in the experiment, this problem makes worst the performance. In this experiment I didn't consider the anchorwoman of the GR2 because is correlated with the class *Other Female*, in fact in some experiment which I trained the segments belong to this class are classify as *Other Female*.

	Training	Test
AnchorwomanGR3	31	15
Other Female	83	20

Table 5.30: Number of the wav files used for train and test the model

Analyzing the confusion matrix reported in figure 5.31 we can compute the recognition rate which in this case is of 90%

	AnchorwomanGR3	Other Female
AnchorwomanGR3	100	0
Other Female	20	80

Table 5.31: Confusion Matrix

In this case, as we can see in figure 5.11, to improve the performance we have to use a feature dimension of 30, which is a very high value, in fact the research of the optimal number of states that I made fixing the feature dimension to this value has request more time respect to the previous layer.

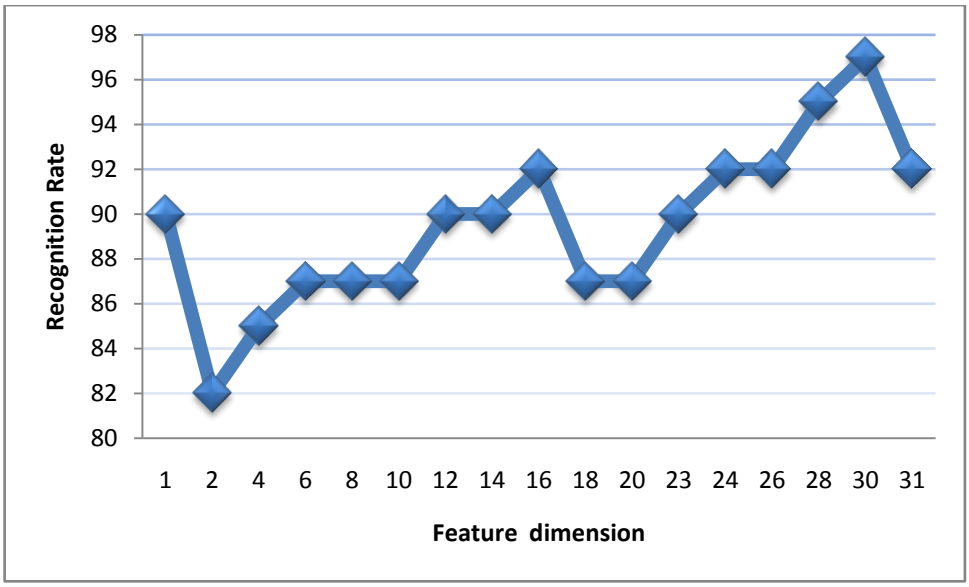

Figure 5.11: Recognition rate vs. feature dimension

However also in this last case the tuning on the number of states can't increase the recognition rate that remains to the 97% confirming that the feature dimension has the major impact on the performance of the classification system.

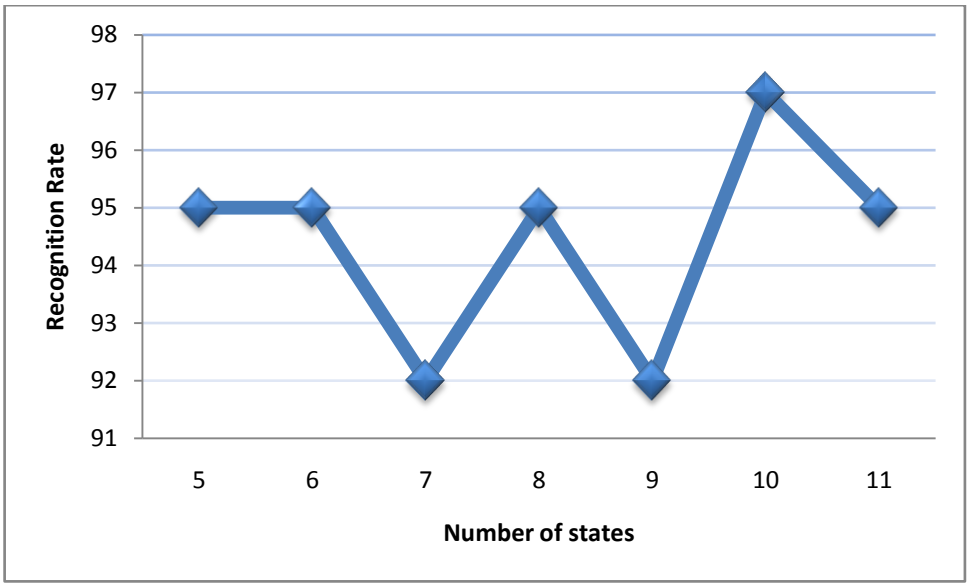

Figure 5.11: Recognition rate vs. number of states

5.7.6 Final hierarchical classification System

From the analysis of the experiments presented above I designed the classification system reported in figure 5.12.

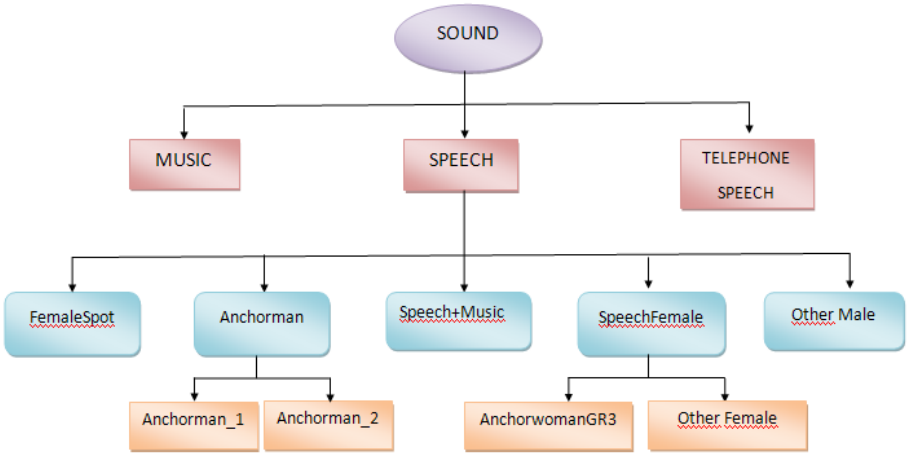

Figure 5.12: Final hierarchical classification system

Then in order to test this system with all the 9 classes individuated I wrote a matlab code, which require as input the name of the folder which contains the audio segments for testing the system, and projects each segment first again the models of the first layer and then according to the result if necessary project it to the models of the next layer and the code ends writing the final results in a text file.

	Test
Anchorman_1GR1	30
Anchorman_2GR1	25
AnchorwomanGR3	15
Other Female	65
FemaleSpot	10
Other Male	68
MaleSpot	20
Music	20
Speech+Music	20
Telephone Speech	30

Table 5.32: Number of the wav files used for test the hierarchical model

As we can see in the table 5.32 I have also tested the model with the classes *Other Male* and *MaleSpot* to show that the model *Other Male* is well formed for both.

	Anch_1GR1	Anch_2GR1	Anch GR3	Other Female	Female Spot	Other Male	Music	Speech+ Music	Telephone Speech
Anch_1GR1	67	7	0	0	0	27	0	0	0
Anch_2GR1	0	80	0	0	0	20	0	0	0
Anch GR3	0	0	100	0	0	0	0	0	0
Other Female	0	0	12	60	15	6	0	3	0
FemaleSpot	0	0	0	0	90	0	0	10	0
Other Male	0	0	0	0	8	79	0	10	0
MaleSpot	0	0	0	0	5	75	15	5	0
Music	0	0	0	0	0	0	95	5	0
Speech +Music	0	0	5	0	15	10	20	50	0
Telephone Speech	0	0	0	0	0	0	0	0	100

Table 5.33: Confusion Matrix

The results are a little worse respect to the results obtained in the experiment above because in those experiment we needed of a priori knowledge here instead no, in fact here we have to add the penalty of the above layer.

The worst results are reached for the class *Speech+Music* here the percentage of segments classified as Music are more high respect to the experiment made to analyze the second layer (paragraph 5.7.3) this is because in that experiment I have used more segments for the testing while now I can use only the segments that wasn't used for the training of the second layer. The part of segments classified as *FemaleSpot* and *OtherMale* (*MaleSpot*) are justified because more of the segments with speech over music are piece of spot.

5.8 Conclusion

A hierarchical approach has permitted me to classify in different layers classes which are correlated each other and to design each layer in an independently way.

Only in this way I could be able to classify 9 different sound classes with a recognition rate of 80%.

CHAPTER 6
SEGMENTATION AND CLASSIFICATION

In order to be able to create for an entire piece of GR an XML document MPEG-7 compliant for describing some audio events (i.e. a reporter speaks on a telephonic line) that occurs in the GR, in this last part of my thesis I join the classification system described in the previous chapter to the automatic segmentation system for radio newscast developed by Vincenzo Dimattia in his thesis [37].

A schema of the final system obtained is reported in figure 6.1.

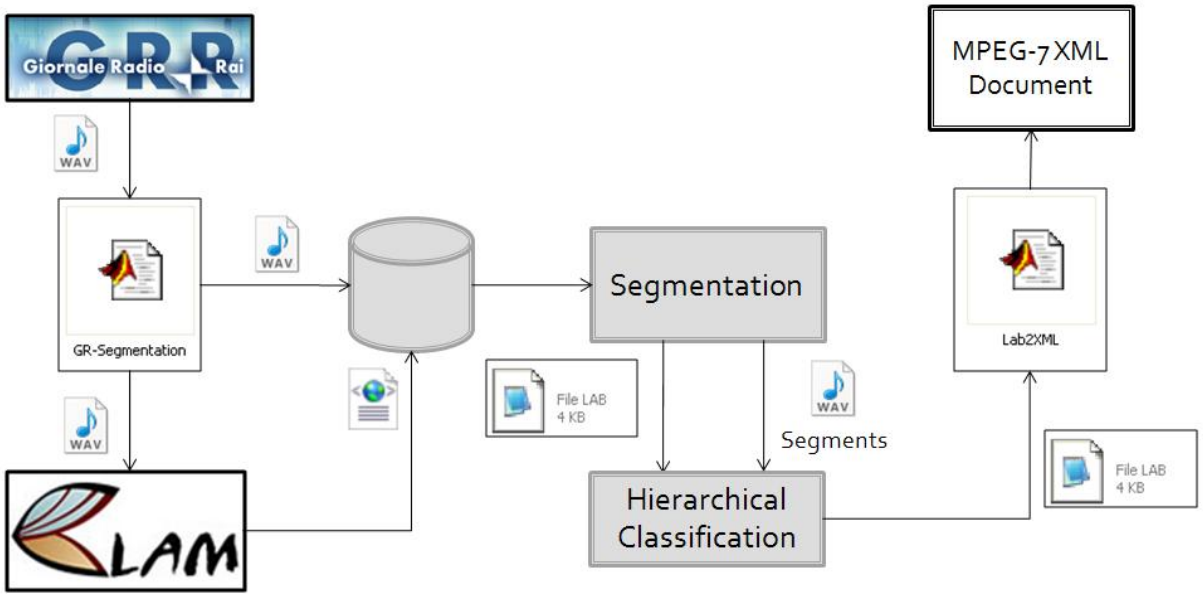

Figure 6.1: The final segmentation and classification system

The GR coming from the web site of the RAI are first segmented by the GR Segmentation matlab code into overlapping segments with duration of 10 minutes and an overlap of 1 minute, because segment too large cannot be processed by the next steps.

This macro segments are then passed to the CLAM which create an XML document with some audio descriptors. This document together with his wav file is stored in a folder.

If ones would segment a GR, which is already stored in the database, can use the matlab code of the segmentation system which require as input only the name of the piece of GR which he would analyze.

The output of the segmentation system, based on MFCC (Mel Frequency Cepstral Coefficient), is a lab file, which contains for each segment the initial and the last sample. This

lab file is used to generate a file wav for each segment, which are then stored in a folder. After that the lab file is modified changing the initial and last sample of each segment in the initial and last second.

The classification system analyzes the segments in the folder, as described in the previous chapter projecting each segment against the models of the hierarchy, and modifies file lab coming from the segmentation system adding to each segments the label assigned by the classification procedure.

Then I implemented a matlab code that converts the file lab in an XML document MPEG-7 compliant. In the following sections I describe more in detail the MPEG-7 Multimedia Descriptor Schemes (MDSs) introduced in the Chapter 1.

6.1 MPEG-7 Multimedia Description Schemes

MPEG-7 Multimedia Description Scheme (MDS) comprises the set of Description Tools (Ds and DSs) dealing with multimedia entities. MDS contains, among others, which we can see in figure 6.2, the following areas; Basic Elements, Content Management and Content Description [38].

- *Basic Elements*: address specific needs of audiovisual content description, such as the description of time, persons, places and other textual annotation.
- *Content Management*: describes different aspects of creation and production of the process such as: title, creators, locations, dates and media information of the audiovisual content (storage media, coding format and compression) to adjust to different network environments.
- *Content Description*: describes the structure, segmentation of the content and semantics (entities, events, relationships) of the audiovisual content. Thus, it allows attaching audio, video, annotation and content management to the multimedia segments, to depict them in detail.

As the target of this section of my thesis is obtain an XML MPEG-7 document, which describes for each GR all the segments automatically compute, and their temporal location in the wav file, I'm interested only in Content Description and Basic Elements.

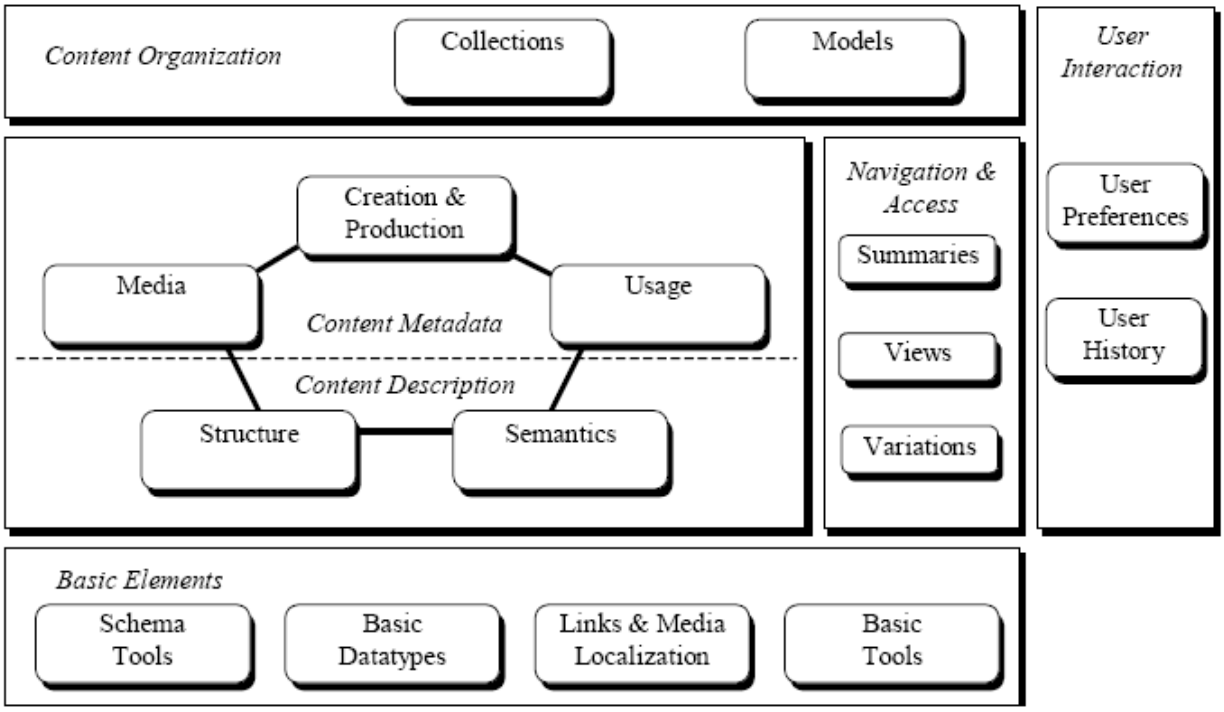

Figure 6.2: Overview of the MPEG-7 Multimedia DSs

6.1.1 Content Description

The description schemes for content description, describe the Structure (region, video frames and audio segments) and Semantics (objects, events and abstract notions). The functionality of each of these classes of description schemes is given as follows:

- *Structural aspects*: description schemes describe the multimedia content from the viewpoint of its structure. The description is built around the notion of Segment description scheme that represents a spatial, temporal or spatial temporal portion of the multimedia content. The Segment description scheme can be organized into a hierarchical structure to produce a Table of Content for accessing or an Index for searching the multimedia content. The Segments can be further described on the basis of perceptual features using MPEG-7 Descriptors for color, texture, shape, motion, audio features and so forth, as well as semantic information using Textual Annotations.
- *Conceptual aspects*: description schemes describe the multimedia content from the viewpoint of real-world semantics and conceptual notions. The Semantic description schemes involve entities such as objects, events, abstract concepts and relationship. The Segment description schemes and Semantic description schemes are related by a

set of links that allows the multimedia content to be described on the basis of both content structure and semantics together [38].

In this project I am interesting only to give a structural description of the segments and so I used the Segment description schemes as we can see in the listing 6.1.

```
- <Audio xsi:type="AudioSegmentType">
  + <MediaTime>
  - <TemporalDecomposition>
    + <AudioSegment>
    + <AudioSegment>
    + <AudioSegment>
    [...]
    + <AudioSegment>
    + <AudioSegment>
    </TemporalDecomposition>
  </Audio>
```

Listing 6.1: AudioSegmentType

6.1.2 Basic Elements

As we can see from figure 6.2 the set of description tools called the MPEG-7 *Basic Elements* is subdivided into four groups:

- *Basic data types*, which represent mathematical constructs useful for multimedia description, such as matrices and vectors.
- *Linking and localization tools,* which are used to specify references within description, to link MPEG-7 description to media, to identify and locate media and to describe time.
- *Basic tools* that address common aspects of multimedia content description. This includes graphs for structuring complex multimedia content descriptions, text annotations, descriptions of people and places, specifications of effective response and description ordering.
- *Schema tools* compared with other basic elements, Schema tools have a different functionality because they not target the description of the content but are used to create valid descriptions and to manage them [13].

In the following table I report the four groups with their description tool

Schema tools	*Basic datatypes*	*Link & localization tools*	*Basic tools*
Base types	Integer, Real	References	Graphs & relations
Root element	Matrices, Vectors	Media Locators	Textual annotation
Top-level tools	Region, Country	Time	Classification schemes
Multimedia content entity tools	Currency		Terms
Package tool			Agents
Description metadata tool			Places
			Affective description
			Ordering key

Table 6.1: Overview of the MPEG-7 Basic Elements

6.2 Automatic generation of an XML Document for the description of the Segments

To create MPEG-7 descriptions of any multimedia content, the first requirement is to build a wrapper for the description using the *Schema Tools*, which should include the header information reported in the listing 6.2.

```
- <Mpeg7 xmlns="urn:mpeg:mpeg7:schema:2001" xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="urn:mpeg:mpeg7:schema:2001 Mpeg7-2001.xsd">
    [...]
  </Mpeg7>
```

Listing 6.2: Root element

The root type provides metadata about the description as well as information that is common to the description, such as the language of the text and the convention for specifying time. The root element provides a choice of elements for creating either a complete description or a description unit, which are defined as follows:

- *Complete Description*: describes multimedia content using the top-level types. For example, the description of an image is a complete description.
- *Description Unit*: describes an instance of a D, DS, or header. A description unit can be used to represent partial information from a complete description. For example, the description of a shape or color is a description unit [13].

In this thesis as we are interested to the description of an entire GR I choose the complete description that describes multimedia content using the top-level types of the *Schema Tools.* The top-level types are used in complete descriptions to describe multimedia content and metadata related to content management. Each top-level type contains the description tools

that are relevant for a particular description task. For describing multimedia content entities such as audio, we have to use the top-level type *ContentEntityType*, as shows in the listing 6.3.

```
- <Description xsi:type="ContentEntityType">
  + <MultimediaContent xsi:type="AudioType">
  </Description>
```

Listing 6.3: ContentEntityType

In order to give an identifier to each segment and to describe its temporal location we have to use the References tool of the linking and localization tools.

The Reference data type provides three basic reference mechanisms:

- *Idref*: References a description element within the same description document. The target is identified by its ID attribute, which is unique within the description document.
- *Xpath:* References a description element using a subset of XML Path Language (XPath). An XPath expression identifies a reference target by its position within the description tree.
- *href*: References a description or description element using a Uniform Resource Identifier (URI). Unlike the idref and xpath mechanism, href references can refer to an element in another description document [11].

In this work I have use the first mechanism as show in the listing 6.4

```
<AudioSegment id="S178">
```

Listing 6.4: *Idref*

MPEG-7 can represent two different kinds of time: (1) media time, which is time measured or stored within the media and (2) generic time, which is time measured in the world. Both and world time use the same representation, except that the data types for world time also contain time zone (TZ) information [11]. Here I describe only the media time data types.

The MPEG-7 media time data types are compatible with time specification used in common multimedia formats such as MPEG-4 and are based on the ISO 8601 standard. The media time data types represent time periods using a start time point (*mediaTimePoint* data type) and

a duration (*mediaDuration* data type). The *mediaTimePoint* data type uses the following syntax:

-YYY-MM-DThh:mm:ss:nFN

which includes the year (Y), month(M), days(D), a separator T, hours(h), minutes(m) and seconds (s), 1/N is a fraction of one second and n the number of those fractions.

For example the *MediaTimePoint* in the listing 6.5 indicates 19 minutes 59 seconds and 957 milliseconds.

```
<MediaTimePoint>T00:19:59:957F1000</MediaTimePoint>
```

Listing 6.5: MediaTimePoint

The *mediaDuration* data type uses the following format:

(-) PnDTnHnMnSnNnF

In this format, each part of the duration contains a count followed by a letter indicating the unit being counted: P is the separator indicating the start of a duration, days are indicated by D, T separates time from days, H indicates hours, M minutes, S seconds and the sub second part uses the same representation as *mediaTimePoint*, with N indicating the counted fractions and F the fractions of one second [11].

For example the *MediaDuration* in the listing 6.6 indicates 2 seconds and 267 milliseconds.

```
<MediaDuration>PT0M2S267N1000F</MediaDuration>
```

Listing 6.6: MediaDuration

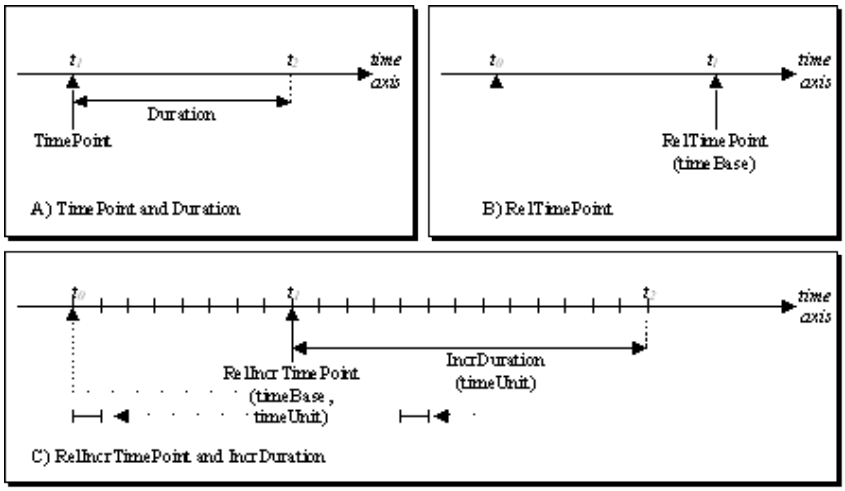

Figure 6.3: Kinds of media time representation

On the top of the *mediaDuration* and *mediaTimePoint* data types, MPEG-7 builds three kinds of media time representation:

- *Simple Time:* The basic representation of an absolute time point (figure 6.2a).
- *Relative time*: Specifies a media time point relative to a time base. Useful if a media segment, such as a story in a news sequence, is placed dynamically. To update the story's description, only the time base (t_0) needs to be changed (figure 6.2b).
- *Incremental time:* Specifies a time period by counting predefined time units (figure 6.2c)

In order to indicate to each segment its label I used the text annotation tool of the basic tool (see table 6.1), as show in the listing 6.7.

```
<TextAnnotation>
  <FreeTextAnnotation>MaleSpot</FreeTextAnnotation>
</TextAnnotation>
```

Listing 6.7: TextAnnotation

6.3 Example of a XML Document for the Description of the Segments

As I said earlier we implemented a matlab code that converts the lab files into a XML document MPEG-7 compliant.

In the code we employed an XML utility in order to create the node of the XML document. In the listing 6.8 I reported the XML document created from the lab file of a GR3.

The code is composed by four functions: the main function reads the lab file and controls if already exists an xml document for this lab file, if yes then calls the function *CreateXML* passing it the entire duration of the GR, the starting point of the first segment and its duration.

The function *CreateXML*, starting from the root element, creates all the nodes of the XML three as instances of the class *docNode* using the method *createElement*, and sets the attributes of each node using the method *setAttribute*. After that the code puts all the nodes in the correct position in the three using the method *appendChild*. This function ends with the writing in an XML document, the wrapper and the description of the first segment

For the second segment the XML document already exists and so instead to call the function *CreateXML* calls the function *addXMLAudioSegment*, with the same parameters, which creates all the nodes necessary for the description of a segment in the same way of the

function *CreateXML* and append this node in the XML three of the document created early. This function ends rewriting the XML document with the new XML three.

The code works in the same way for the other segments till the end of the lab file.

```
<?xml version="1.0" encoding="utf-8" ?>
- <Mpeg7 xmlns="urn:mpeg:mpeg7:schema:2001" xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="urn:mpeg:mpeg7:schema:2001 Mpeg7-2001.xsd">
 - <Description xsi:type="ContentEntityType">
  - <MultimediaContent xsi:type="AudioType">
   - <Audio xsi:type="AudioSegmentType">
    - <MediaTime>
       <MediaTimePoint>T00:00:00</MediaTimePoint>
       <MediaDuration>PT9M59S980N1000F</MediaDuration>
      </MediaTime>
    - <TemporalDecomposition>
     - <AudioSegment id="Segment_1.wav">
      - <TextAnnotation>
         <FreeTextAnnotation>Female</FreeTextAnnotation>
        </TextAnnotation>
      - <MediaTime>
         <MediaTimePoint>T00:00:00</MediaTimePoint>
         <MediaDuration>PT0M0S766N1000F</MediaDuration>
        </MediaTime>
       </AudioSegment>
            [...]
     - <AudioSegment id="Segment_47.wav">
      - <TextAnnotation>
         <FreeTextAnnotation>SpeechTel</FreeTextAnnotation>
        </TextAnnotation>
      - <MediaTime>
         <MediaTimePoint>T00:09:46:257F1000</MediaTimePoint>
         <MediaDuration>PT0M7S570N1000F</MediaDuration>
        </MediaTime>
       </AudioSegment>
     - <AudioSegment id="Segment_48.wav">
      - <TextAnnotation>
         <FreeTextAnnotation>MaleSpot</FreeTextAnnotation>
        </TextAnnotation>
      - <MediaTime>
         <MediaTimePoint>T00:09:53:827F1000</MediaTimePoint>
         <MediaDuration>PT0M6S153N1000F</MediaDuration>
        </MediaTime>
       </AudioSegment>
     </TemporalDecomposition>
    </Audio>
   </MultimediaContent>
  </Description>
</Mpeg7>
```

Listing 6.8: Example of an XML Document MPEG-7 compliant created from a lab file of a GR3

In order to test our XML document we validated it using the NIST MPEG-7 Validation Service [65]. In the listing 6.9 I report the output of the validation.

```
NIST MPEG-7 Validation Result ...

<Mpeg7Validation file="11-gr3_labels.xml">
  <Total warning="0" error="0" fatal="0"/>
  <Processed element="1094" attribute="368" time="25092ms"/>
</Mpeg7Validation>
```

Listing 6.9: NIST MPEG-7 Validation Result

6.4 Graphical Representation

In order to quickly evaluate the performance of the segmentation and calassification system and to obtain a graphical representation of the output this system I wrote a Matlab code that require as input the two lab files of the manual and the automatic segmentation, and the wave file of the GRR. In the figure is reported the mesh of the two vectors created downsampling the wave track and substituting the value of each sample with an integer number according to the specific class to whom the sample belongs.

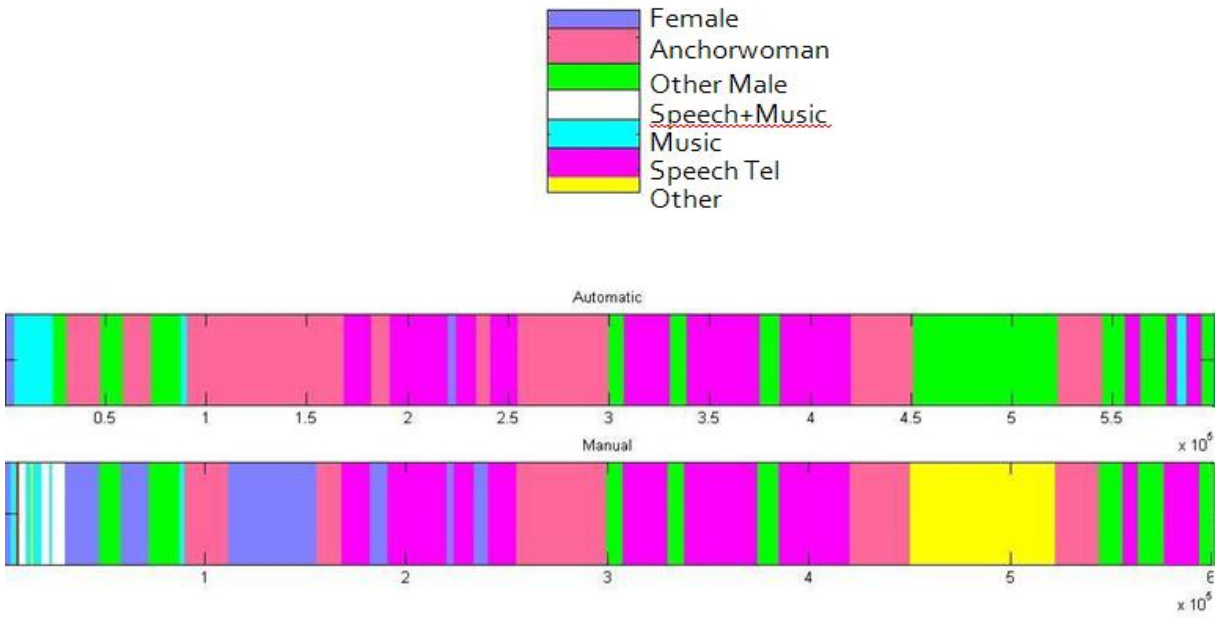

Figure 6.4: Output of the segmentation and classification system for the first 10 minutes of a GR3

In figure 6.4 we can note that the performance of the system is satisfactory. The segmentation is quite perfect only at the beginning of the track very short segments aren't detected. The classifier works also well in fact all the telephone speech are classified correctly and it only confuses the anchorwoman speech with the female speech.

As in the manual segmentation we have classes that aren't considered in the classifier we labelled these segments as *Other*. So in correspondence of the yellow segment in the figure 6.4 the classifier doesn't make an error.

REFERENCES

[1] NIST. Rich transcription task. http://nist.gov/speech/tests/rt/, 2005.

[2] J.-L. Gauvain, L. Lamel, and G. Adda. The limsi broadcast news transcription system. Speech Communication, 37(1-2):89–108, 2002. ISSN 0167-6393.

[3] S. E. Johnson, P. Jourlin, K. S. Jones, and P. C. Woodland. Information retrieval from unsegmented broadcast news audio. International Journal of Speech Technology, 4:251– 268, 2001.

[4] J. Saunders. Real-time discrimination of broadcast speech/music. In IEEE International Conference on Acoustics, Speech, and Signal Processing. (ICASSP'96). Conference Proceedings., volume 2, pages 993–996, Atlanta, GA, USA, May 1996. IEEE.

[5] M. F. McKinney and J. Breebaart. Features for audio and music classification. Proc. Of ISMIR, pages 151–158, 2003.

[6] T. Jehan. Creating Music by Listening. PhD thesis, Massachusetts Institute of Technology, 2005.

[7] MPEG-7: Overview of MPEG-7 Description Tools, José M. Martínez, Copyright © 2002 IEEE. Reprinted from IEEE Computer Society, July-September 2002.

[8] Manjunath, Salembier, Sikora, "Introduction to MPEG-7 Multimedia Content Description Interface" John Wiley & Sons, LTD

[9] Information technology - Multimedia content description interface - Part 4: Audio. ISO/IEC 15938-4:2002. International Organisation for Standardisation, 2002.

[10] world wide web consortium (w3c)'s xml homepage: http://www.w3.org/xml/.

[11] B.S. Manjunath, P. Salembier, and T. Sikora, editors. Introduction to MPEG 7: Multimedia Content Description Language. John Wiley and Sons, Ltd, West Sussex, England, 2002.

[12] World Wide Web Consortium (W3C)'s XML-Schema homepage, http://www.w3.org/XML/Schema.

[13] ISO/IEC FDIS 15938-5:2003, INternational Standard Document, 2003

[14] José M. Martínez MPEG-7 Overview (version 10) ISO/IEC JTC1/SC29/WG11N6828 Palma de Mallorca, October 2004

[15] http://mpeg7.doc.gold.ac.uk/

[16] Hyoung-Gook Kim, Nicolas Moreau, Thomas Sikora, "MPEG-7 Audio and Beyond: Audio Content Indexing and Retrieval" Wiley, October 2005.

[17] Burred J. J. and Lerch A. “Hierarchical Automatic Audio Signal Classification”, *Journal of the Audio Engineering Society*, vol. 52, no. 7/8, pp. 724–739.

[18] Moorer J. “The Optimum Comb Method of Pitch Period Analysis of Continuous Digitized Speech”, *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 22, no. 5, pp. 330–338.

[19] Jollife I. T. (1986) *Principal Component Analysis*, Springer-Verlag, Berlin.

[20] Golub G. H. and Van Loan C. F. (1993) *Matrix Computations*, Johns Hopkins UniversityPress, Baltimore, MD.

[21]Cardoso, J.F. and Laheld, B.H., Equivariant adaptive source separation. *IEEE Trans. On Signal Processing*, 4:112-114, 1996.

[22]Reynolds D. A. (1995) Speaker Identification and Verification Using Gaussian Mixture Speaker Models, *Speech Communication*, pp. 91–108.

[23] Haykins S. (1998) *Neural Networks*, 2nd Edition, Prentice Hall, Englewood Cliffs, NJ.

[24] Rabiner L. R. and Jung B. (1993) *Fundamentals of Speech Recognition*, Prentice Hall, Englewood Cliffs, NJ.

[25] Cortes C. and Vapnik V. (1995) “Support Vector Networks”, *Machine Learning*, vol. 20, pp. 273–297.

[26] Casey M. A. (2001) “MPEG-7 Sound Recognition Tools”, *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 6, pp. 737–747.

[27] J.F. Cardoso and A. Souloumiac,"Blind beamforming for non Gaussian signals",IEEE Proceedings-F, 140, 362–370, dec. 1993

[28] http://www.tsi.enst.fr/~cardoso/Algo/Jade/jade.m

[29] L. R. Rabiner and B.H. Juang, “An introduction to hidden Markov models,” IEEE ASSP Mag., pp 4-16 Jun. 1986.

[30] G.D. Forney Jr., “The Viterbi Algorithm,” Proc. IEEE, vol. 61 no.3, pp. 263-278, Mar. 1973.

[31] A. P. Dempster, N. M. Laird, and D. B. Rubin, “Maximum likelihood from incomplete data via the EM algorithm, ”Roy. Stat. Soc., vol. 39, no. 1, pp. 1-38, 1977.

[32] S. E. Levinson, L. R. Rabiner, and M. M. Sondhi, “An introduction to the application of the theory of probabilistic functions of a Markov process to automatic speech recognition,” Bell Syst. Tech. j., vol. 62, no. 4, pp. 1035-1074, Apr. 1983.

[33] B. H. Juang, ”Maximum likelihood estimation for mixture multivariate stochastic observations of Markov chains,” AT&T Tech. j., vol. 64, no. 6, pp. 1235-1249, July-Aug. 1985.

[34] B. H. Juang, S. E. Levinson, and M. M. Sondhi, “Maximum likelihood estimation for multivariate mixture observations of Markov chains, ”IEEE Trans. lnformat. Theory, vol. IT-32, no. 2, pp. 307-309, Mar. 1986.

[35] http://www.speech.kth.se/wavesurfer/

[36] Kim H.-G. and Sikora T. (2004a) “Comparison of MPEG-7 Audio Spectrum Projection Features and MFCC applied to Speaker Recognition, Sound Classification and Audio Segmentation”, *Proceedings IEEE ICASSP 2004*, Montreal, Canada, May.

[37] Vincenzo Dimattia “An Automatic Audio Classification System for Radio Newscast” TSC UPC Terrassa March 2008.

[38] Van Beek, P., A.B., Heuer, J., Martinez, J., Salembier, P., Smith, J. and Walker T. MPEG-7: Multimedia Description Schemes, ISO/IEC FDIS 15938-5:2001, INternational Standard Document, 2001.