Proiect cofinanțat din Fondul Social European prin Programul Operațional Sectorial Dezvoltarea Resurselor Umane 2007-2013
Investește în oameni!

Proiect InnoRESEARCH - POSDRU/159/1.5/S/132395
Burse doctorale și postdoctorale în sprijinul inovării și competitivității în cercetare

UNIVERSITATEA **POLITEHNICA** DIN BUCUREŞTI
Facultatea: Electronică, Telecomunicaţii şi Tehnologia Informaţiei
Departamentul: Dispozitive, Circuite și Aparate Electronice

Nr. Decizie Senat 238 din 30.09.2015

TEZĂ DE DOCTORAT

OPTIMIZĂRI ÎN RECUNOAŞTEREA LIMBAJULUI VORBIT

OPTIMIZATIONS IN SPOKEN LANGUAGE RECOGNITION

Autor: Ing. Alexandru CARANICA
Conducător de doctorat: Prof. Dr. Ing. Corneliu Burileanu

COMISIA DE DOCTORAT

Președinte	**Prof. Dr. Ing. Gheorghe Brezeanu**	de la	**Univ. POLITEHNICA din București**
Conducător de doctorat	**Prof. Dr. Ing. Corneliu Burileanu**	de la	**Univ. POLITEHNICA din București**
Referent	**Prof. Dr. Ing. Daniela Tărniceriu**	de la	**Univ. "Gheorghe Asachi" din Iași**
Referent	**Prof. Dr. Ing. Corneliu Rusu**	de la	**Universitatea Tehnică din Cluj-Napoca**
Referent	**Prof. Dr. Ing. Cristian Negrescu**	de la	**Univ. POLITEHNICA din București**

București 2015

ACKNOWLEDGEMENTS

I wish to express my sincere appreciation and gratitude to those who have contributed to this thesis and supported me in one way or the other during this journey. I am indebted to many people for making the time spent working on my Ph.D. an unforgettable experience.

First of all, I would like to express my special appreciation and thanks to my PhD coordinator, prof. dr. eng. Corneliu Burileanu, for giving me the opportunity to work on this thesis and for his continuous guidance and support. I would like to thank him for encouraging me to do research, for allowing me to improve my skills in a very strong team that he coordinates, the *Speech and Dialogue Research Laboratory*.

I also want to express my gratitude to the evaluation committee members: prof. dr. eng. Daniela Tărniceriu, prof. dr. eng. Corneliu Rusu and prof. dr. eng. Cristian Negrescu for their helpful comments and suggestions.

I also owe a great debt of gratitude to dr. eng Horia Cucu, for his valuable insight on time-management and task prioritization. I would have never started "real" work without his supervision. My special thanks also go to dr. eng. Andi Buzo, for guiding me in the final stages of writing, and introducing me into the world of unsupervised speech processing. I am also thankful to Bogdan Ludusan, from *Laboratoire de Sciences Cognitives et Psycholinguistique (ENS)*, for a fruitful collaboration in our Spoken Term Discovery approach.

Finally, I would like to thank my family, friends and Elena, for their infinite support through everything.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

AI	Artificial Intelligence
ANN	Artificial Neural Nets
ASR	Automatic Speech Recognition
BIC	Bayesian Information Criterion
CLR	Cross-Likelihood Ratio
Cnxe	Normalized Empirical Cross Entropy
PAM	Pulse Amplitude Modulation
PCM	Pulse Code Modulation
PWM	Pulse Width Modulation
DET	Detection Error Tradeoff
DFT	Discrete Fourier Transform
DSP	Digital Signal Processing
DCT	Discrete cosine transform
DTW	Dynamic Time Warping
EM	Expectation Maximization
FFT	Fast Fourier Transform
FSG	Finite State Grammar
GMM	Gaussian Mixture Model
HAC	Hierarchical Agglomerative Clustering
HMM	Hidden Markov Model
IDFT	Inverse Discrete Fourier Transform
IR	Information Retrieval
LPC	Linear Prediction Coding
LVCSR	Large vocabulary continuous speech recognition
MFCC	Mel-frequency Cepstral Coefficients
ML	Maximum Likelihood
MLP	Multi-layer Perceptron
MTWV	Maximum Term Weighted Value
NLP	Natural Language Processing

NN	Neural Network
PDF	Probability Density Function
PLP	Perceptual Linear Predictive Analysis
PNCC	Power Normalized Cepstral Coefficients
SER	Sentence Error Rate
SLM	Statistical Language Model
SLU	Spoken Language Understanding
SNR	Signal to noise ratio
STFT	Short-tine Fourier Transform
TRAP	Temporal Patterns structure of neural network
VAD	Voice Activity Detection
WER	Word Error Rate
ZCR	Zero Crossing Rate

CHAPTER 1

INTRODUCTION

1.1 THESIS MOTIVATION

Over the past decades, “machine learning” has become one of the main stays in computer technology and with that, a rather central part of our daily life, when we are interacting with “smart” devices around us. Despite of the rather sci-fi name, machine learning is the science of getting computers to act without being explicitly programmed to do so. It uses huge amount of data to improve the program's own understanding, detecting patterns in data and adjusting its learning actions accordingly. For example, Facebook's News Feed changes according to the user's personal interactions with other users. If a user frequently tags a friend in photos, writes on his wall or "likes" his links, the News Feed will show more of that friend's activity in the user's News Feed, due to statistically presumed closeness. In recent years, machine learning has given us self-driving cars, practical continuous speech recognition, effective and instant web search plus a vastly improved understanding of the human genome. Machine learning is so widely used today that we probably interact with its algorithm’s a dozen times a day without even knowing. Many researchers also think it is the best way to make progress towards human-level Artificial Intelligence (AI) [Standford, 2015].

Nowadays, learning techniques can also be successfully applied to Speech Recognition, in the field of Digital Signal Processing (DSP). Engineers and scientists have studied the phenomenon and production of speech communication, with an eye on creating more and more efficient and effective systems for human-computer interaction. Hands free applications increased in usage, especially in consumer based hardware. Applications such as Apple`s Siri or Google

Now offer speech commands, as a direct interface to the phones operating system. According to Jurafsky [Jurafsky, 2008], while many tasks are better solved with visual or pointing interfaces ("keyboard" or "touch"), speech has the potential to be a better human-computer interface than the keyboard for tasks where full natural language communication is useful, in which touch panels are also not appropriate. This can include hands-busy or eyes-busy applications, such as where the user has objects to manipulate or equipment to control, and his attention cannot be detained from the job at hand. This can further be explained by the fact that speech is the most natural communication method used by humans to exchange information, and the human user is not required to have any additional skills to be able to use a speech enabled device.

Automatic speech recognition (ASR) addresses the problem of mapping an acoustic signal to a sequence of words, and it`s been a hot topic in the international scientific community for over twenty years now. Development of resources and methods led eventually to high-performance commercial systems for most of the internationally spoken languages, such as English, French, Chinese, etc. Automatic speech recognition is still an unsolved topic for many languages, mainly because there is a lack of acoustic and linguistic resources needed for development (it is the case of so-called under-resourced languages) and the scientific research community is not stimulated by any national or international evaluation campaigns (as opposed to languages such as English, French or Chinese). With resources and methods no longer a challenging problem for international languages, research is now focused on advanced problems such as noise robustness, multi-speaker systems, emotional classifiers, or recognition performance improvement for specific scenarios like conversational and spontaneous speech, non-native accents and output intelligibility of an ASR system.

However, for the Romanian language, recent breakthroughs have been achieved by members of the Speech and Dialogue research group [SpeeD, 2015], by launching one of the first Large Vocabulary Continuous Speech Recognition (LVCSR) system for Romanian [Cucu, 2011a; Cucu, 2011b]. LVCSR is a subclass of ASR, which aims at transcribing most words in a specific language or at least a broad sub-domain of it containing thousands or hundreds of thousands of words. The automatic speech recognition system developed by the SpeeD group is continuously improved and upgraded. Recently, significant improvements where reported (between 30% and 35% relative word error rate reductions), obtained with increasing the number of speech, text corpus resources and to the implementation of noise robust speech features [Cucu, 2014]. Although getting closer to ASR systems available for other common languages, there is still much work to be done, especially in the Natural Language Processing area and transcription prost-processing. The output of an Automatic Speech Recognition (ASR) system consists of raw text, often in lowercase format and without any punctuation information. The transcript is intended to be as close as possible to the speech content of the audio file [Buzo, 2014]. This may be useful for a wide range of applications, such as database indexing and classification, where a machine uses this information in search related algorithms. For other tasks, where humans need to easily read and understand the text (e.g. subtitling, dictation and broadcast news transcription), post-processing the output raw text through diacritics (where applicable), capitalization and punctuation restoration greatly improves the readability of automatic speech transcripts. Apart from the insertion of punctuation marks and capitalization, enriching speech recognition covers other activities, such as detection and filtering of disfluencies, sentence segmentation, etc.

The applications of speech recognition in daily life are multiple, and truly there are no limits to the use cases of this technology: from niche applications like medical interfaces and industrial command and control systems to consumer applications, where modern operating systems offer speech interfaces to interact with the system. As ASR systems get better and better, users expect more interaction and understanding from their "smart" device, regardless of whether that device is a car, smartphone or PC. Microsoft Cortana, Google Now, Apples Siri, all push the boundaries of natural language processing and spoken language recognition to "understand" and extract as

much information as possible from the speech signal before returning it`s interaction results to its users. With the increasing availability of spoken documents in different languages, some of those languages, as stated above, even considered under-resourced in the speech community, there is a growing need for unsupervised methods of information extraction. An appropriate method for these types of task, spoken term discovery and detection systems identify recurring speech fragments from raw speech, without any knowledge of the language at hand [Park, 2008]. Applications employing automatically discovered terms have quickly appeared, having a wide focus, ranging from topic segmentation [Malioutov, 2007] to document classification [Dredze, 2010] or spoken document summarization [Harwath, 2013]. This is where unsupervised learning comes into play, with its immediate applications it can have in languages with little or no resources.

In this alive context, this doctoral thesis is particularly concerned with research in the area of increasing the output intelligibility of an ASR system and with unsupervised multi-language methods of information extraction, in the context of under resourced languages.

1.2 DEFINING THE PROBLEM OF SPOKEN LANGUAGE RECOGNITION

Speech is a versatile mean of communication. It conveys linguistic (e.g., message and language), speaker (e.g., emotional, regional, and physiological characteristics of the vocal apparatus), and environmental (ex., where the speech was produced and transmitted) information. Even though such information is encoded in a complex form, humans can relatively decode most of it. This human ability has inspired researchers to develop systems that would emulate such ability. From phoneticians to engineers, researchers have been working on several fronts to decode most of the information from the speech signal. Some of these fronts include tasks like identifying speakers by voice, detecting the language being spoken, transcribing speech, translating and understanding speech. Despite the human ability, researchers learned that extracting information from speech is not a straightforward process. The variability in speech due to linguistic, physiologic, and environmental factors challenges researchers to reliably extract relevant information from the speech signal [Clark, 2010].

From a DSP point of view, ASR is the process of speech-to-text transcription: the transformation of an acoustic signal into a sequence of words, without necessarily understanding the meaning or intent of what was spoken. When the input acoustic signal contains speech uttered by different speakers, the ASR task can be regarded as a two-step process: speaker diarization (who spoke when?) and speech-to-text transcription (what did he say?). But the speech waveform signal contains much more information, like carrying information about the timing, intonation, and voice quality of the speaker. These paralinguistic aspects convey information about the speaker's emotion and physiology, as well as disambiguating between different possible meanings. The various sources of speech variability make the general task a very challenging one. Nevertheless, in many practical situations, the variability is restricted. For example, there may be a single, known speaker, or the speech to be recognized may be carefully dictated text rather than a spontaneous conversation, or the recording environment may be quiet and non-reverberant. In speech-to-text transcription, a distinction is made between parts addressing acoustic variability (acoustic modeling), and parts addressing linguistic uncertainty (language modeling).

There are several sources of variability which can be clustered in four main areas:

a) speech task domain
b) speaking style
c) speaker characteristics
d) recognition environment

The speech "task domain" is another important factor that influences the difficulty of the speech transcription process. Aspects of the specific speech recognition task which affect the difficulty of the speech transcription process include the language and the size of the vocabulary to be recognized, and whether the speech comes from a limited domain. Different languages present different challenges for a speech recognizer. For a large number of languages there are very few speech and text resources available. These so called low resourced languages are spoken by a large number of people, but no prior work of collecting and organizing speech and/or text resources has been done. Other languages "suffer" from a complex morphology. For example rich morphology languages such as French and Romanian have larger vocabularies than poor morphological languages such as English. The size of the vocabulary is an important factor because it is obvious that a command and control task (with a limited vocabulary) is much simpler than a spontaneous telephone speech recognition task (with a 64k words vocabulary).

Nevertheless, larger vocabularies do not always mean a more difficult ASR task. The linguistic uncertainty of the possible speech utterances also plays a significant role. For example, a tourism-specific ASR task with a 64k words vocabulary which mostly contains proper names (places, restaurants, hotels, etc.) is not as difficult as a spontaneous telephone speech recognition task with an equal-size vocabulary. The low linguistic uncertainty (perplexity) of the first task makes it less difficult.

Another important factor which influences the difficulty of the speech process is the "speaking style". The speaking style refers to how fluent, natural or conversational the speech is. Obviously, isolated words speech recognition, in which each word is surrounded by some sort of pause, is much easier than recognizing continuous speech in which words run into each other and have to be segmented. In fact, in the early days of automatic speech recognition, systems solved the problem of where to locate word boundaries by requiring the speaker to leave pauses between words: the pioneering dictation product Dragon Dictate [Baker, 1989] is a good example of a large-vocabulary isolated words recognition system. One way to deal with this variability is through the construction of "speaker dependent" speech recognition systems, but this demands a new system to be constructed for each speaker. "Speaker independent systems", on the other hand, are more flexible in that they are designed to recognize any speaker. In the early days of automatic speech recognition, systems solved the problem of where to locate word boundaries by requiring the speaker to leave pauses between words. However, this is an unnatural speaking style and most research in speech recognition is now focused on continuous speech recognition, in which word boundary information is not easily available. The problem of continuous speech recognition thus involves segmentation into words, as well as labeling each word.

Finally, "speaker characteristics" have also a significant impact on the accuracy of a speech recognizer. Although human beings can understand quite well non-native speech, the automatic speech recognition systems exhibit very limited robustness when they are required to recognize this type of speech, thus non-native or accented speech recognition is still an open issue in a high number of studies that have been published in the past few years on this subject [Tan, 2007; Oh, 2007; Tan, 2008; Sam, 2010].

From a theoretical point of view, most successful speech recognition systems are based on statistical frameworks which brings us four problems that must be addressed:

a) The acoustic processing problem, i.e., to decide what acoustic data X is going to be estimated. The goal is to find a representation that reduces the model complexity (low dimensionality) while keeping the linguistic information (discriminability), despite the effects from the speaker, channel or environmental characteristics (robustness). In general, the speech waveform is transformed into a sequence of acoustic feature vectors, and this process is commonly referred to as "feature extraction".

b) Let $P(W|X)$ denote the probability that the words W were spoken given that the acoustic evidence X was observed. Then the recognizer should select the best sequence of words W^* satisfying $W^* = \underset{W}{\operatorname{argmax}} P(W|X)$. This is the "acoustic modeling" problem, to decide on how $P(X|W)$ should be computed. Thus, several acoustic models are necessary to characterize how speakers pronounce the words of W given the acoustic evidence X. The acoustic models are highly dependent of the type of application (fluent speech, dictation, commands). In general, several constraints are made so that the acoustic models are computationally feasible.
c) The "language modeling" problem, to decide on how to compute the prior probability $P(W)$ for a sequence of words. The most popular model is based on a Markovian assumption that a word in sentence is conditioned on only the previous N-1 words. Such statistical modeling method is called an n-gram.
d) The search problem, to find the best word transcription W^* for the acoustic evidence X, given the acoustic and language models.

Figure 1.1 offers a quick overview of how the above components interact to obtain an ASR system. More details about the mathematical methods behind HMMs and some of the methods used for signal processing and feature extraction are described in Chapter 2 and 3.

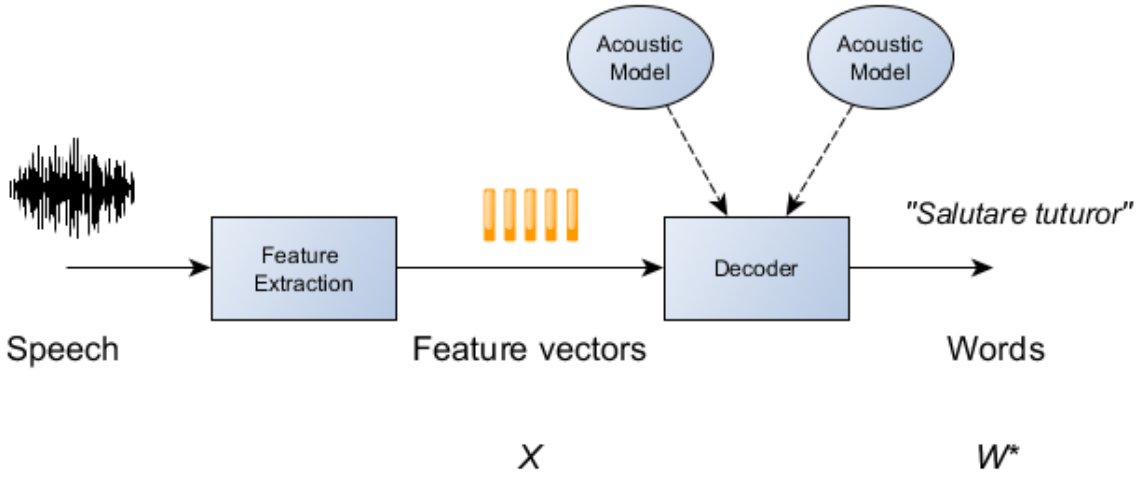

Figure 1.1 Quick overview of an ASR core architecture

Apart from the automatic speech recognition core, most modern ASR systems comprises of a speech pre-processing frontend, which are responsible with voice activity detection and speaker diarization, and a transcription post-processing framework. Voice activity detection is needed in order to split the raw audio signal into segments comprising speech and segments comprising music, noise, silence, etc. Obviously, only the speech segments will be further processed. Speaker diarization is the process of segmenting a speech signal based on the speakers that uttered the corresponding signals. Speaker diarization practically answers the questions "who spoke when?" by generating speech segments associated with speaker information (speaker ids). This information is used in the post-processing framework to associate speech transcriptions with the corresponding speakers. The speaker diarization block also preserves the timing information associated with the speech segments [Buzo, 2014]. As research presented in this thesis will show, this post-processing framework can be further improved to optimize an ASR system. Because most of the time the output of an ASR system consists of raw text, in lowercase format, with no diacritics, capitalization or punctuation marks, these can be restored using statistical linguistic information and unformatted transcriptions organized into paragraphs. Moreover, the post-processing framework formats numbers and dates (converts numbers written with words into

numbers written with digits) creating a more intelligible transcription. This greatly improves readability and intelligibility of the system, as Table 1.1 shows.

Table 1.1 Comparison of an ASR output with / without post-processing

Raw ASR output
iată ce spun telespectatorii noștri pe facebook în continuare îi rog să ne trimită propuneri pentru guvernul ponta
ASR output with post-processing
Iată ce spun telespectatorii noștri pe Facebook, în continuare îi rog să ne trimită propuneri pentru guvernul Ponta.
Ideal ASR output
Iată ce spun telespectatorii noștri, pe Facebook. În continuare, îi rog să ne trimită propuneri pentru Guvernul Ponta.

As mentioned earlier, current state-of-the-art paradigm for continuous speech recognition is the hidden Markov model (HMM), in particular, the HMM-based acoustic model used in conjunction with an n-gram model. The commercial availability of speech recognition, and the need for web-based language techniques have provided an important incentive for development of real systems. The availability of very large on-line corpora has enabled statistical models of language at every level, from phonetics to discourse. This is the most used method for ASR, but lately, there are also hybrid approaches based on Neural Networks (NN) coupled with statistical ones, such as Hybrid HMM/ANN Systems [Bourlard, 1994]. They approach the acoustic modelling using neural networks but use Markov models for language modeling. Continuous speech recognition also poses some additional difficult issues to be sold, given that the term continuous and speech, bound together, can be understood as spoken language recognition. This gives the user more freedom as he expects to speak freely, without constraints such as intonation pauses or accent correction. Therefore, historically distinct fields (speech recognition, computational linguistics, natural language processing) have begun to merge.

1.3 WHERE SPEECH RECOGNITION IS GOING

This is an exciting time to be working in speech and language processing. Availability of very large on-line corpora have enabled statistical models of language at every level, from phonetics to discourse. This pushed the boundaries of just “plain recognition” and along with the commercial success of speech enabled devices (phones, cars, etc.), just simple recognition is no longer sufficient for a device to be “smart”.

With the ever-increasing amounts of vast digital audio data being created and broadcasted daily from various sources, a pressing need exists for intelligent information extraction and retrieval methods, in the speech community. There are various applications for these methods, from document retrieval containing speech data like broadcast news, telephone conversations and roundtable meetings to audio query searches. In recent years, numerous workshops hosted benchmarking initiatives to evaluate new algorithms for multimedia access and retrieval, such as MediaEval (MediaEval, 2011-2014), or as special sessions at relevant conferences in the field of speech communication (ZeroSpeech Challenge, InterSpeech 2015, OpenKWS). Many of these spoken documents are in different languages, some of those even considered under resourced in the speech community, hence also a growing need for an unsupervised method of information extraction and retrieval. In an ideal information retrieval scenario, the end user should be able to perform open vocabulary search and retrieval in any language, over a large collection of spoken documents, in a front-end application, with results being returned in a matter of seconds. For this reason, most of the systems employ some sort of pre-indexing of the speech corpus, prior to search, without the advanced knowledge of the query terms and make use of unsupervised learning

techniques to adapt to low-resourced language. It is a question of "how much you can learn from a speech signal without knowing the language at hand".

Information retrieval and extraction have direct applications in the field of Natural Language Processing: finding out where needed, textual resources, reside and extracting pertinent facts from those textual resources. Spoken dialogue systems typically use manually predefined semantic elements to parse users' utterances into unified semantic representations. To define the knowledge and the structure, domain experts and professional annotators are often involved, and the cost of development can be expensive. Therefore, current technology usually limits conversational interactions to a few narrow predefined domains/topics. With the increasing conversational interactions, this information retrieval and extraction algorithms come into play and help build Spoken Language Understanding (SLU) component [Chen, 2015]. In order to achieve this goal, two questions need to be addressed: (i) given unlabeled raw audio recordings, how can a system automatically induce and organize the domain-specific concepts? (ii) with the automatically acquired knowledge, how can a system understand individual utterances and user intents? [Chen, 2015] proposes such a SLU system, by focusing on five important stages: ontology induction, structure learning, surface form derivation, semantic decoding, and behavior prediction. To solve the first problem, ontology induction automatically extracts the domain-specific concepts by leveraging available ontologies and distributional semantics. Then an unsupervised machine learning approach is proposed to learn the structure and then infer the meaningful organization for the dialogue system design.

With this information at hand, we can conclude that further development is the addition of a deeper level of understanding, as the aim is to not only to recognize speech, but also to extract the meaning and intent of what has been said, enabling voice driven systems as a whole to react in an intelligent way, appropriate to the user's needs.

1.4 Thesis Objectives and Outline

After a short introduction regarding the field of speech and natural language processing, I will now briefly describe the main objectives of this thesis and summarize its main chapters and applied research.

As resources and methods are no longer a challenging problem for current ASR systems, research is now focused on more advanced problems (noise robustness, emotional classifiers, output intelligibility, low-resourced languages, etc.). Also, with the ever increasing availability of spoken documents in different languages, there is a growing need for unsupervised methods of information extraction, classification and retrieval.

In the above context, the main objectives of this thesis are:

a) Overview over the state of the art in speech recognition and natural language processing;
b) Research the theory and process of automatic speech recognition, build and design of a small vocabulary, automatic speech recognition system;
c) Enhance the capabilities of an automatic speech recognition system by including a post-processing framework to statistically restore capitalization and punctuation of raw text resulted from the output of the system;
d) Identify the most appropriate methods (statistical, neural) in order to obtain a robust phone recognizer toolkit to be used for pattern matching;
e) Research over pattern matching and unsupervised learning techniques for spoken term discovery and detection of spoken audio content;

The thesis is organized around six chapters, as follows:

Chapter 1 started with an introduction in the field of speech recognition and machine learning. Then it summarizes the main tasks needed for obtaining an ASR system and highlights some of the components and methods for post-processing optimizations. In the context of merging historically distinct fields (speech recognition, computational linguistics, natural language processing), this chapter also offers a brief overview of the current issues and directions going forward, such as spoken language recognition and unsupervised learning.

Chapter 2 introduces the basic principles and proposed models for speech and hearing perception. This theoretical chapter describes the proposed literature models for speech and hearing perception, highlights fundamental characteristics of the speech signal along with the main methods for processing and analyzing of the voice signal. As current state-of-the-art ASR techniques do not use directly the time-domain waveform to model the speech signal, special attention is given to current speech features extraction and analysis methods.

Chapter 3 begins by offering an overview, then describes the current state of the art algorithms in automatic speech recognition, looking at both statistical and neural network approaches, for a deeper level of understanding in ASR. The rich mathematical framework of HMMs makes statistical approaches very feasible for ASR, and one of the goals of this chapter is to confirm the validity and reproducibility of this methods. Another objective is the integration of the components and toolkits necessary to build a continuous recognition system, briefly describe the processes involved in speech representation, the mathematics behind it and the analysis and experimental setup for improving and optimizing the primary evaluation metrics. In the second part of the chapter, we also take a look at techniques for automatic phoneme recognition from spoken speech, using Neural Network based approaches (TRAP, STC).

Chapter 4 surveys some of the post-processing means of increasing the output intelligibility of an ASR system. A set of experiments regarding language model generation, training and evaluation in the context of capitalization and punctuation recovery for the Romanian language are presented. To the best of our knowledge this is the first such system developed for the Romanian language and these are the first re-capitalization and punctuation restoration results reported for this language.

Chapter 5 begins with an overview over information processing and retrieval, then proposes several unsupervised spoken term discovery and detection experiments along with evaluation scenarios, using databases and tasks from several relevant workshops in the field (MediaEval, ZeroSpeech). We further investigate whether the use of multi-language resources as input features helps the process of term discovery for under-resourced languages, by a phone recognition approach with multilingual acoustic models from different languages. The novel Power Normalized Cepstral Coefficients (PNCC) features are investigated for improved robustness to noise, along with a three-state posterior representation of the speech signal. Results are compared with current popular baseline MFCC representation.

Chapter 6 summarizes the main conclusions of this thesis and underlines the author's contributions. Along with these, some future work ideas and steps to be taken for our research are provided.

CHAPTER 2

FUNDAMENTAL CHARACTERISTICS OF THE SPEECH SIGNAL

2.1 SPEECH PRODUCTION AND PERCEPTION

To understand the human speech production mechanism, first, we ought to examine the anatomy of the human vocal system (the speech signal production apparatus). Most would agree that one should grasp the process of speech production, before attempting to model a framework that would understand it. Once this mechanism is better understood, we may attempt to create systems and frameworks that recognize its distinguishing characteristics and nuances, thus recognizing the speech, or, even a more complex task, an individual speaker [Beigi, 2011].

The most recent couple of decades have seen enormous advancement in the performance, reliability, and wide-spread use of speech-processing devices. Using mathematical models for human speech production and perception, has been an important factor in the improved performance of these devices. But why is speech so hard to handle? First, listening is much harder than it looks (or sounds): there are all sorts of different problems and sources of variability going on at the same time, thus making this task a very difficult one for machines to handle: background noise, speech accent, homophones (words that sound identical but mean totally different things) all contribute to the complexity of a speech recognition system. Then there are issues like syntax and semantics, and how they help our brain decode the words we hear.

Weighing all these factors up, it`s easy to see that recognizing, moreover, understanding spoken words in real time, requires complex systems and algorithms. In the following chapters

we shall introduce some models and techniques to represents speech in a digital world, to help shed some light on the theory behind ASR.

2.1.1 Models for Speech Production

In speech production, the information to be transmitted is encoded in the form of a continuously varying analog waveform that can be transmitted, recorded, manipulated, and ultimately decoded by a human listener. In the case of speech, the fundamental analog form of the message is an acoustic waveform, the speech signal. Figure 2.1 shows the complete process of producing and perceiving speech from the formulation of a message in the brain of a talker, to the creation of the speech signal, and finally to the understanding of the message by a listener. This process has been referred in literature as the "speech chain" [Denes, 1993].

The speech message could be initially represented as text, which the talker then converts into a phonetic representation that describes the message and the manner in which the sounds are intended to be produced. The International Phonetic Association (IPA) provides a set of rules for phonetic transcription using an equivalent set of specialized symbols. Next step in the speech production process is the neuro-muscular control, in which the human speech articulators (tongue, lips, teeth, etc.) move in a manner consistent with the sounds of the desired spoken message. Finally, the "vocal tract" subsystem physically creates the necessary sound sources to create and acoustic waveform (speech signal) that encodes the information into speech.

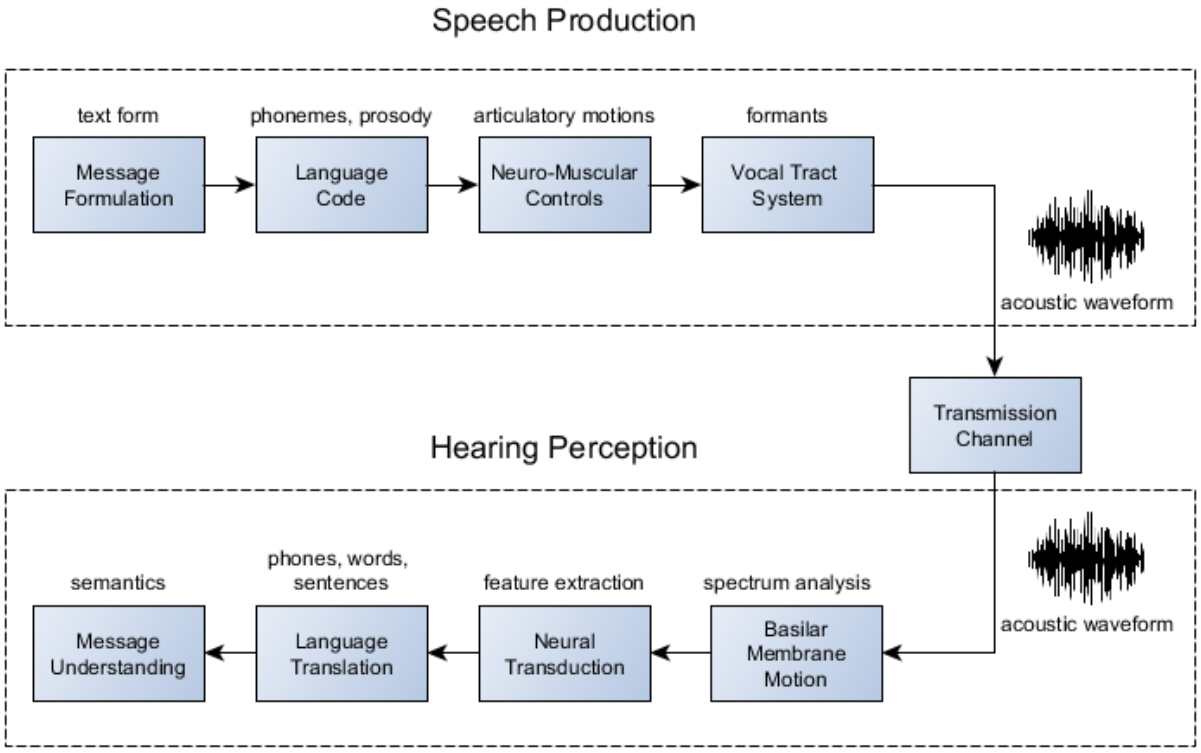

Figure 2.1 Production and speech perception [Rabiner, 2007]

Thus, the "vocal tract system" can be further approximated to a source-filter model (Figure 2.2), where a sound source excites a vocal tract filter [Stuttle, 2003]. This model is at the heart of many speech analysis methods and drives thinking in speech perception research also.

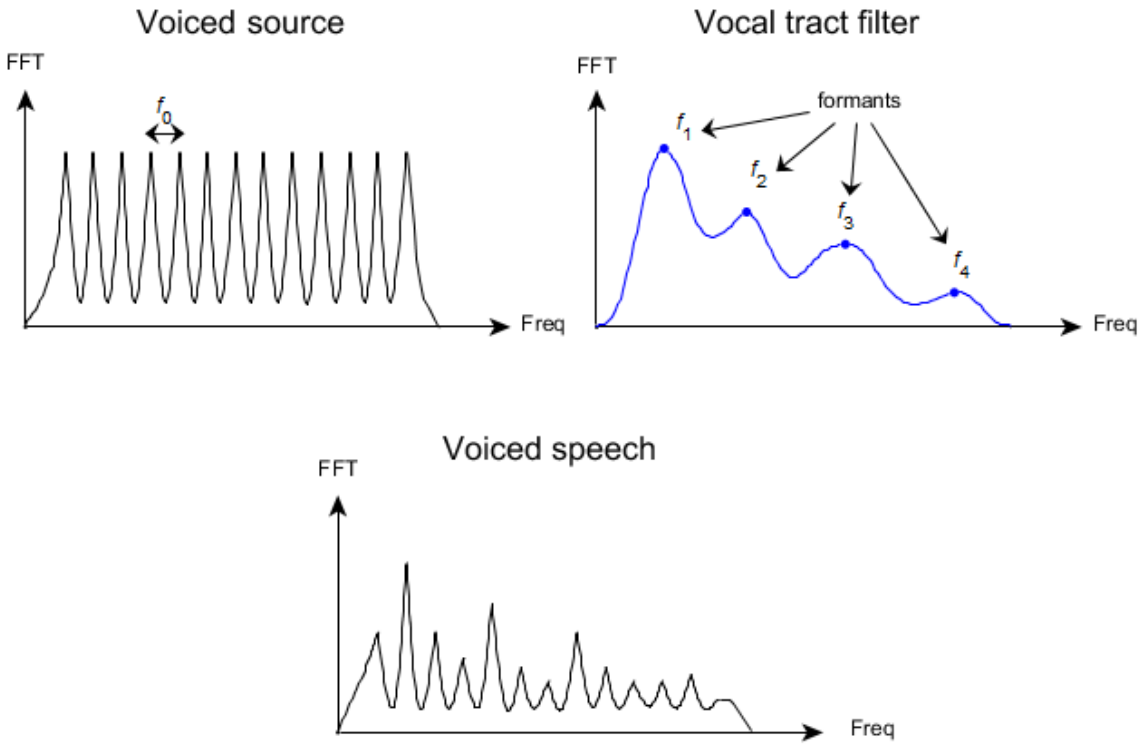

Figure 2.2 The idealized source-filter model for the vocal tract system [Stuttle, 2003]

The source can be split into various broad classes, as it can be periodic, due to the opening and closing of the vocal folds in the larynx. This form of speech is called voiced. In unvoiced speech the sound source is not a regular vibration but rather vibrations are caused by turbulent airflow due to a constriction in the vocal tract. The frequency of vibration of the vocal folds in voiced speech is called the fundamental frequency f_0, and is repeated at regular intervals in the voice signal spectrum. The vocal tract filter response is characterized by a series of formants or resonant frequencies. The attenuation of the source by the vocal tract response is obtained by multiplying the two frequency representations together. Thus, by interpolating the pitch peaks in the resulting speech, it is possible to recover the original vocal tract response or spectral envelope.

As shown in Figure 2.1, this upper part is the "Speech Production" stage, discussed above. The model also contains the "Hearing Perception" stage, as shown progressing to the lower part of the figure, which will be treated in the next sub-section.

2.1.2 Models for Hearing Perception

It has been hypothesized that the human speech production and recognition mechanisms evolved in tandem [O'Shaughnessy, 1987], so it`s important to consider the human auditory mechanism in the recognition process. The human ear focuses acoustic waveforms and converts them to electrical impulses in the cochlea, a liquid-filled concentric spiral tube in the inner ear. Sound waves are then transported by the fluid to the middle ear. Hairs on the organ of Corti will vibrate in response to movements in the fluid to fire all the neurons connected to them. Hairs resonate at different characteristic frequencies. Hence, the neural signals transfer signals proportional to the energy levels in different frequency bands, to the brain. The perception of frequency is uniform within certain frequency bands in the human ear, called critical bands. The resolution is non-linear, with the most sensitive frequency resolution up to about 1 kHz. We will talk more about critical bands in the following chapters, and how are used in the preprocessing

stage of the speech signal. Also, we will introduce a non-linear psychoacoustic frequency scale, the Mel Scale, in the next sub-section.

Returning to the speech chain model, it shows the series of steps from capturing speech at the ear to understanding the message encoded in the speech signal. The first step is to transform the speech signal into a spectral representation. This is done within the inner ear, by the basilar membrane, which acts as a non-uniform spectrum analyzer by spatially separating the spectral components of the incoming speech signal and thereby analyzing them by what amounts to a non-uniform filter bank. The "neural transduction" block translates spectral features into a set of distinctive features that can be decoded and processed by the brain. This sound features are then converted into a set of phones, phones in words, and words in sentences by the human brain. Now the brain processes the semantics behind the sentences to understand the meaning of the message and respond or take appropriate action.

The speech model is rudimentary at best, but it is generally agreed that some physical correlate of each of the steps in the speech perception model, occur within the human brain, and thus the entire model is useful for thinking about the processes that occur [Rabiner, 2007]. Most of the blocks from this model forms the basis of an ASR system, as described in section 2.3 of this thesis. We will also treat the transmission channel, mainly what happens to an ASR system if a channel is noisy, and how we can compensate for channel distortions that make speech and message understanding more difficult in real communication environments.

2.1.3 Critical bands

The perception of frequency is uniform within certain frequency bands in the human ear, called critical bands. In each critical band sound is analyzed independently. Each band corresponds with an equal section of cochlea. The resolution is non-linear, with the most sensitive frequency resolution up to about 1 kHz. Below 500 Hz bandwidths are constant, equal 100 Hz. Over 500 Hz the width of each next critical band is 20% larger than of the band below. It is possible to model the human auditory system as a set of band-pass filters with bandwidth of corresponding critical band. An idealized version of such a filter bank is shown in Figure 2.3.

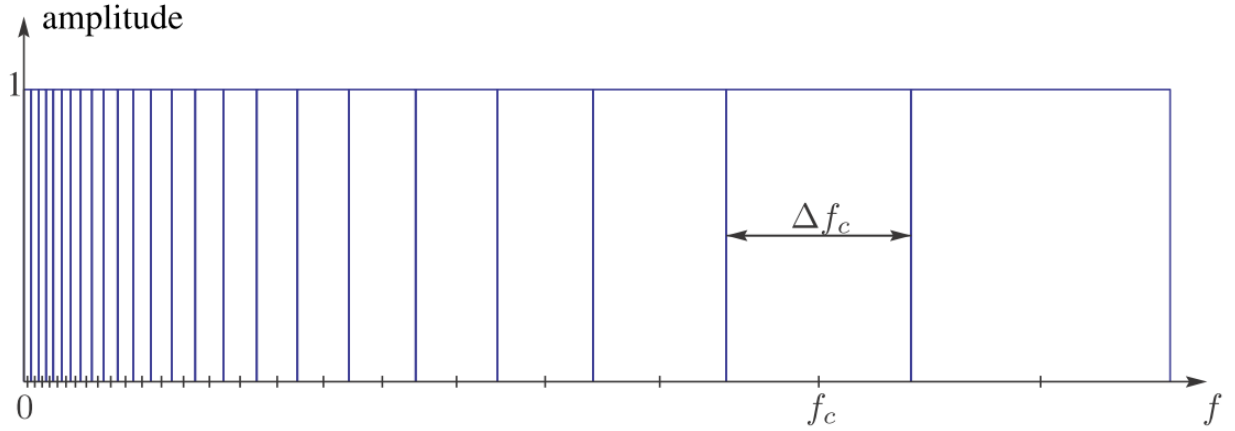

Figure 2.3 The idealized band pass filters, from [Rabiner, 2007]

As stated earlier, there are various scales that can approximate the non-linear frequency scale, such as Bark or Mel scale, treated in the next section.

2.1.4 Mel Scale

As discussed in previous sections, the human ear resolves frequencies non-linearly across the audio spectrum. Empirical evidence suggests that designing an ASR front-end to operate in a

similar non-linear manner improves recognition performance [Young, 2006]. A good approximation of the analysis process behind the human auditory system is represented by Mel scale filtering. “Mel, an abbreviation of the word melody, is a unit of pitch. It is defined to be equal to one thousandth of the pitch of a simple tone with frequency of 1000 Hz with an amplitude of 40 dB above the auditory threshold”. The above definition is based on the experiments done by Stevens, Volkman and Newman in late 1930s. The results were published in 1937 [Stevens, 1937] and 1940 [Stevens, 1940].

It is linear below 1 kHz, and logarithmic above, with equal numbers of samples taken below and above 1 kHz (Figure 2.4 and Figure 2.5). The mel scale is based on experiments with simple tones (sinusoids) in which subjects were required to divide given frequency ranges into four perceptually equal intervals or to adjust the frequency of a stimulus tone to be half as high as that of a comparison tone [Huang, 2001]. One Mel is defined as one thousandth of the pitch of a 1 kHz tone. As with all such attempts, it is hoped that the Mel scale more closely models the sensitivity of the human ear than a purely linear scale and provides for greater discriminatory capability between speech segments. Mel-scale frequency analysis has been widely used in modern speech recognition systems.

A popular formula to convert *f* hertz into *m* Mel is:

$$m=1125\cdot\ln\left(1+\frac{f}{700}\right) \tag{2.1}$$

There is no single mel-scale formula. Equation 2.1 can be expressed with different log bases:

$$m=1125\cdot\ln\left(1+\frac{f}{700}\right)=2595\cdot log_{10}\left(1+\frac{f}{700}\right) \tag{2.2}$$

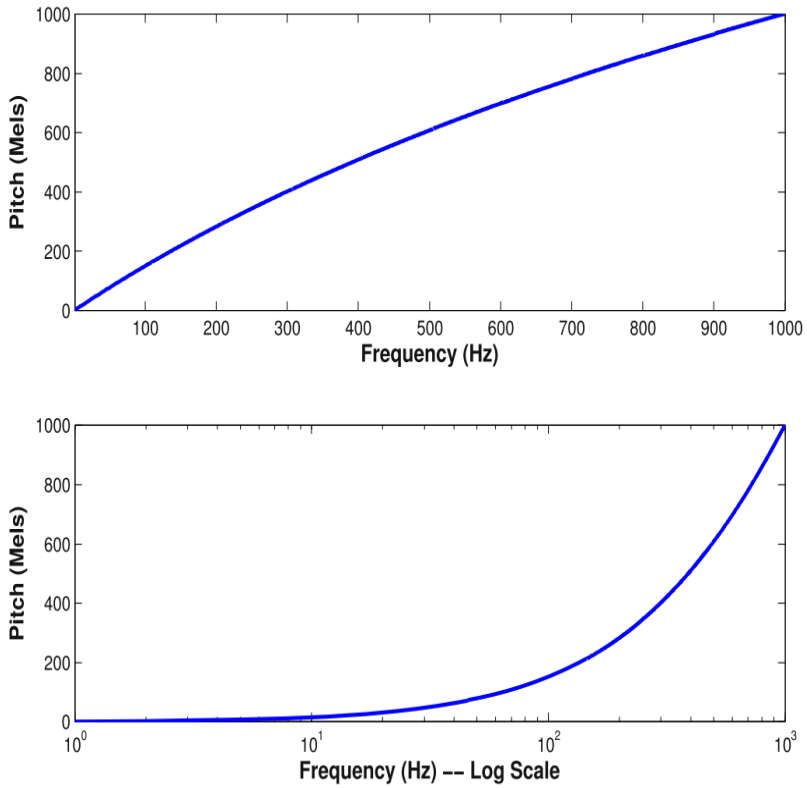

Figure 2.4 Plots of pitch Mel scale versus Frequency for up to 1000 Hz [Beigi, 2011]

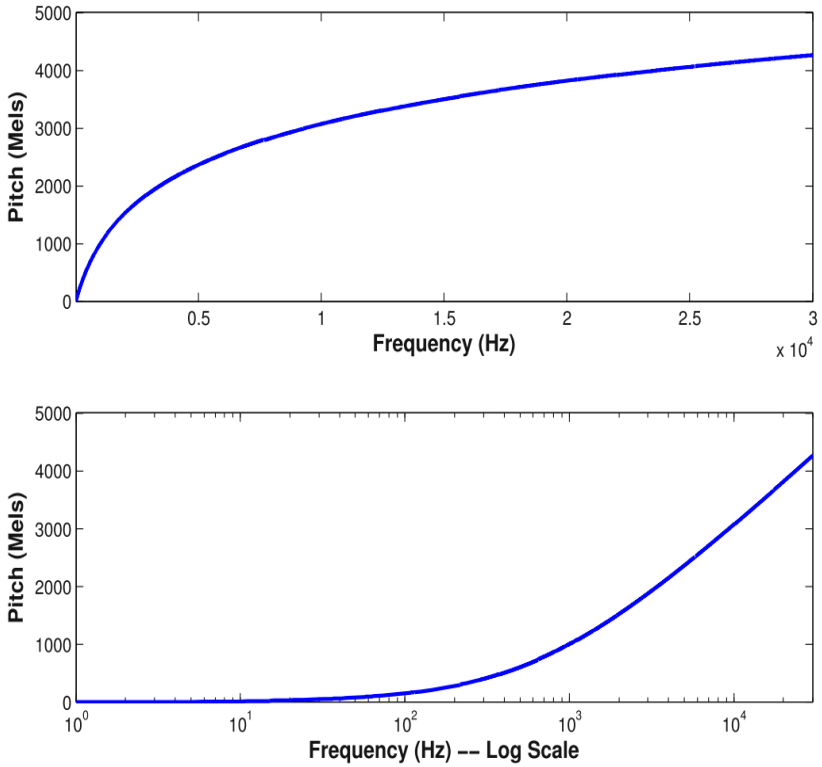

Figure 2.5 Mel scale versus Frequency for the entire audible range [Beigi, 2011]

A number of techniques in the modern spoken language system, such as cepstral analysis, have benefited tremendously from perceptual research as discussed further in this thesis.

2.2 PREPROCESSING OF THE SPEECH SIGNAL

The fundamental process in digital processing of the voice signal is, without doubt, it`s digital representation. The signals digital domain is a simple combination of integral numbers, where the digitization of the analog signal is a transformation of its analog form into a temporal series of integral numbers.

2.2.1 Sampling Process

To simplify the processing of continuous signals, the infinite set of possible values the analog signal may take on in a finite interval [*a*, *b*], may be reduced to a finite set through another mapping process called “sampling”. The speech signal is an observed measurement, done with respect to the passing of time. It may be viewed as the mapping of time into the strength of the speech waves at any given instance of time. This action is called discretization and the newly defined signal, capable of mapping this finite set of points to a higher level measurement, is called a discrete signal.

Since speaker recognition is basically a passive process and only observes the audio signal to make a decision, we are only concerned with a sampling process at the beginning and once the signal is in a sampled state, the algorithms are independent of the analog world. This is not the case for other speech related disciplines, such as Speech Synthesis, that has to deal with the conversion of a sampled signal to an analog one. There we must apply some sort of data reconstruction technique, but it`s not the subject of this thesis.

There are several possible ways to sample a signal, namely, periodic, cyclic rate, multirate, random, and pulse width modulated sampling. In speech processing, most of the time periodic

sampling is used, in which the sampling frequency (rate of sampling) is fixed [Beigi, 2011]. There are some speech related applications with variable sampling, that deal with low-activity signals and that they may use variable sampling. Encoding algorithms are such an example, like MP3, OGG Vorbis, etc. But for the sake of simplicity, speaker recognition systems use periodic sampling, such as Pulse Code Modulation (PCM).

So, for use in speech recognition, the analog speech waveform, *s(t)* has to be converted to a digital representation, *s[n]* which is formed by periodically sampling the analog signal *s(t)* at intervals equally spaced *T* seconds apart, as follows:

$$s[n] = s(n \cdot T) \tag{2.3}$$

where *T* is defined as the sampling period, and its inverse

$$F_s = \frac{1}{T} \tag{2.4}$$

as the sampling frequency. In the speech applications, F_s can range from 8 kHz to 44 kHz for high-fidelity audio applications. Because speech is relatively low bandwidth (mostly between 100 Hz-8 kHz), 8000 samples/sec (8 kHz) is sufficient for most basic ASR. It is important to remember that the analog speech signal *x(t)* can be uniquely recovered given its digital signal *s[n]* if the analog signal s*(t)* has no energy for frequencies above the "Nyquist" frequency of F_s / 2 [Huang, 2001]. For the record, there are two major types of samplers which may be used for sampling time dependent signals such as the speech signal: Pulse Amplitude Modulation (PAM) and Pulse Width Modulation (PWM) samplers.

2.3 SHORT-TIME CHARACTERISTICS

The speech signal is a slowly timed varying signal (it is called quasi-stationary). When examined over a sufficiently short period of time (between 5 to 10 ms), its characteristics are fairly stationary. However, over long periods of time (on the order of 1/5 seconds or more) the signal characteristic change to reflect the different speech sounds being spoken. Therefore, short-time spectral analysis is the most common way to characterize the speech signal, and in the following subchapters we introduce two basic short-time analysis functions useful for speech signals, short-time zero-crossing rate and short-time energy.

2.3.1 Short-time zero-crossing rate

The short-time zero-crossing rate is a simple parameter to calculate and particularly important of the speech signal. Along with the short-time energy, presented in the next section, it is used in Voice Activity Detection (VAD) systems or for speech-silence detection. VAD is a very important practical step in doing speaker or speech recognition. Close to 30% of the audio frames in a normal audio recording are silence frames. This means that through silence removal, the recognition process may become faster by the same rate. In speech recognition, the extraneous silence segments will produce spurious nonsense words by taking leaps through different arcs of Hidden Markov Models. It is much simpler to use some energy threshold to cut out moments of silence (or only consider moments of speech) than to have to model the silence, due to the variability of its noise content. The elimination of silence will not only increase the accuracy, but it will also reduce un-necessary processing energy and some cases bandwidth utilization [Beigi, 2011].

Many different algorithms have been devised to use this information in order to detect the voice activity in a speech signal. Examples are maximum likelihood techniques [Gauci, 2008], neural network techniques and discrete wavelet transform methods [Stadtschnitzer, 2008].

The short-time zero crossing rate is defined as the weighted average of the number of times the speech signal changes sign within the time window, as follows [Huang, 2001]:

$$Z_n = \sum_{n=0}^{N-1} \frac{1 - sgn(x_{n+1}) \cdot sgn(x_n)}{2} \tag{2.5}$$

2.3.2 Short-time energy

Along with ZCR, short-time energy is used in VAD systems, to detect silence and voiced speech. If the energy of the signal is high, then we have a sequence of speech. A lower energy value indicates a non-speech sequence or a silence zone.

The energy, for an analog voice signal *s(t),* is defined as:

$$E = \int_{-\infty}^{\infty} s^2(t)dt \tag{2.6}$$

Because the speech signal is sampled and digitized, thus is discrete, we need to calculate the short-time energy, determined by the analysis window:

$$E_k = \sum_{n=1}^{k} x^2(n) \tag{2.7}$$

where *x(n)* is the discrete sampled signal after windowing:

$$x(n) = s(n) \cdot w(n), 0 \le n \le N-1 \tag{2.8}$$

In the above equation, *s(n)* is the analog voice signal and *w(n)* is the applied window function. When windowing a signal, we multiply the time-domain signal with a window function. The concept here is to minimize the spectral distortion by using the window to taper the signal to zero at the beginning and end of each frame.

Typically the Hamming window is used, which has the form:

$$w(n) = 0.54 - 0.46\cos\left(\frac{2\pi n}{N-1}\right), \quad 0 \le n \le N-1 \tag{2.9}$$

Besides the above example, the energy of the speech signal can also be determined recursively using the following formula:

$$P(n) = -\sum_{i=1}^{N_a} a_{pw}(i) \cdot P(n-i) + \sum_{j=0}^{N_b} b_{pw}(j) \cdot s^2(n-j) \tag{2.10}$$

where a_{pw} and b_{pw} are the coefficients of two low pas filters (usually of first and second order).

Figure 2.6 shows an example of the short-time energy and zero-crossing rate for a segment of speech with a transition from unvoiced to voiced speech. In both cases, the window is a Hamming window of duration 25 ms, equivalent to 401 samples at a 16 kHz sampling rate [Rabiner, 2007].

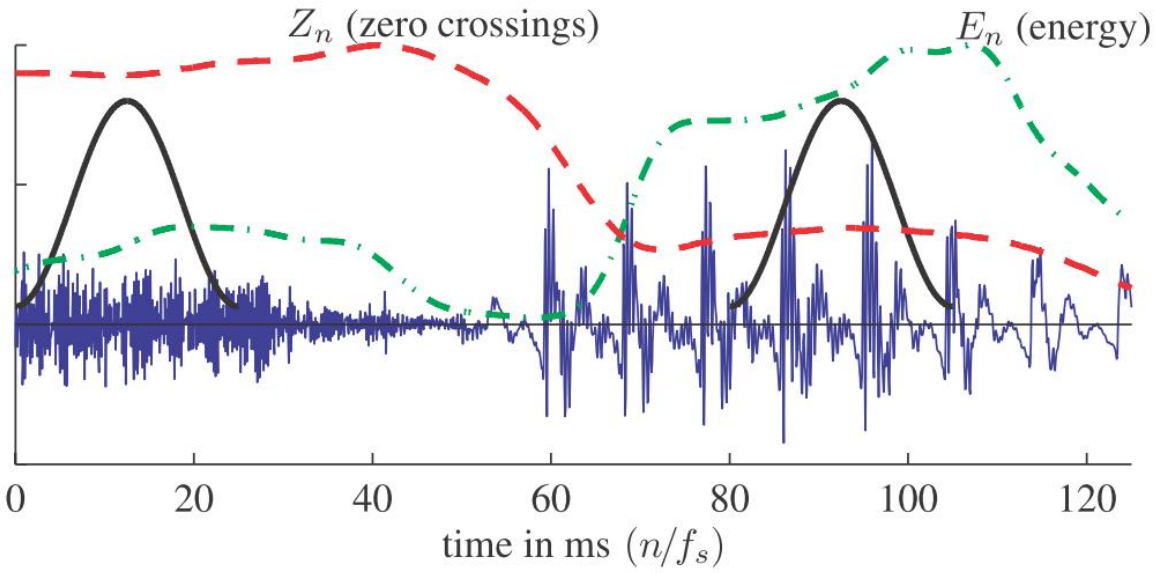

Figure 2.6 Example of zero-crossing rate and short-time energy on a section of speech waveform (unvoiced then voiced) [Rabiner, 2007]

From Figure 2.6 we can see that during the unvoiced interval, the zero-crossing rate is relatively high compared to the zero-crossing rate in the voiced interval. At the same time, the energy is relatively low in the unvoiced region compared to the energy in the voiced region.

2.4 FREQUENCY DOMAIN CHARACTERISTICS

There are a lot of frequency characteristics used in speech processing, which are obtained by converting the time based signal into the frequency domain using the Fourier Transform, like: fundamental frequency, formant frequencies, spectral centroid, spectral roll-off, etc. These features can be used to identify the notes, pitch, rhythm, and melody, but we will cover the first two characteristics, which are the most important for speech analysis.

2.4.1 Fundamental frequency

Continuous speech is a set of complicated audio signals which makes producing them artificially difficult. As previously stated, Speech signals are usually considered as voiced or unvoiced, but in some cases they are something between these two. Voiced sounds consist of fundamental frequency, f_0 and its harmonic components produced by vocal cords (vocal folds). The vocal tract modifies this excitation signal causing formant (pole) and sometimes antiformant (zero) frequencies [Witten, 1982].

Each formant frequency has also an amplitude and bandwidth and it may be sometimes difficult to define some of these parameters correctly. The fundamental frequency and formant frequencies are probably the most important concepts in speech synthesis and also in speech processing in general. With purely unvoiced sounds, there is no fundamental frequency in excitation signal and therefore no harmonic structure either and the excitation can be considered as white noise. The airflow is forced through a vocal tract constriction which can occur in several places between glottis and mouth. Some sounds are produced with complete stoppage of airflow followed by a sudden release, producing an impulsive turbulent excitation often followed by a more protracted turbulent excitation [Kleijn, 1998]. Unvoiced sounds are also usually more silent and less steady than voiced ones. Whispering is the special case of speech. When whispering a voiced sound there is no fundamental frequency in the excitation and the first formant frequencies produced by vocal tract are perceived.

Another commonly used method to describe a speech signal is the spectrogram which is a time-frequency-amplitude presentation of a signal. The spectrogram and the time-domain waveform for the Romanian utterance "*bună ziua*" are presented in Figure 2.7. Higher amplitudes are presented with darker gray-levels so the formant frequencies and trajectories are easy to perceive. Also spectral differences between vowels and consonants are easy to comprehend. Therefore, spectrogram is perhaps the most useful presentation for speech research. From Figure 2.7 it is easy to see that vowels have more energy and it is focused at lower frequencies. Unvoiced consonants have considerably less energy and it is usually focused at higher frequencies. With voiced consonants the situation is something between of these two.

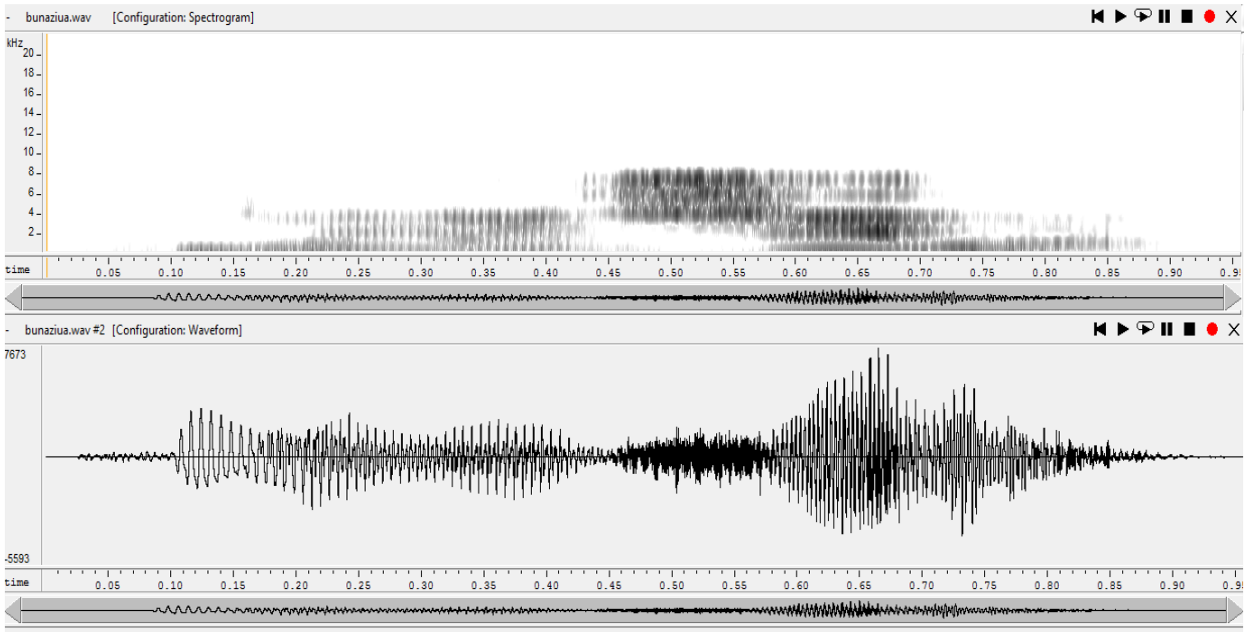

Figure 2.7 Spectrogram and time-domain presentation of the Romanian utterance "bună ziua"

For determining the fundamental frequency or pitch of speech, for example, a method called cepstral analysis may be used [Cawley, 1996; Kleijn, 1998]. Cepstrum is obtained by first windowing and making Discrete Fourier Transform (DFT) for the signal and then logaritmizing power spectrum and finally transforming it back to the time-domain by Inverse Discrete Fourier Transform (IDFT). The procedure is shown in Figure 2.8.

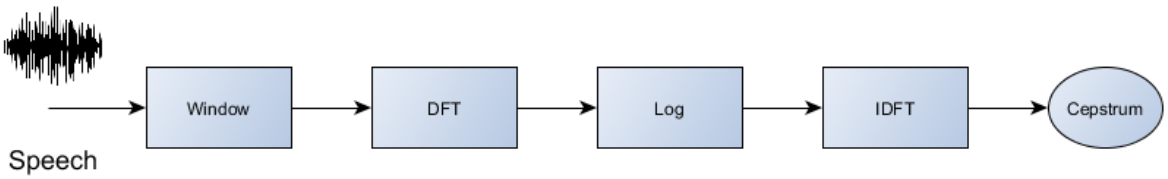

Figure 2.8 Cepstral analysis

Cepstral analysis provides a method for separating the vocal tract information from excitation. Thus the reverse transformation can be carried out to provide smoother power spectrum known as homomorphic filtering. Fundamental frequency or intonation contour over the sentence is important for correct prosody and natural sounding speech. More on cepstral analysis in the detailed subchapter to follow (MFCC features).

2.4.2 Formant frequencies

Formants are distinctive frequency components of the acoustic signal produced by speech or singing. The information that humans require to distinguish between speech sounds can be

represented purely quantitatively by specifying peaks in the amplitude/frequency spectrum. Expert spectrogram readers are able to recognize speech by looking at a spectrogram, particularly at the formants. It has been argued that they are very useful features for speech recognition, but they haven't been widely used because of the difficulty in estimating them [Huang, 2001]. The vocal tract is changing shape so that the resonance is changing. The definition and estimation of formant locations is a difficult task. In general, the vocal tract length is inversely proportional to the height of the format in the frequency range of a speaker. This means that the longer the vocal tract length (for example in adult males), the lower the format. As the vocal tract length is shortened (for example in female speakers and children), the formant locations move up higher in the frequency domain, as illustrated in Figure 2.9.

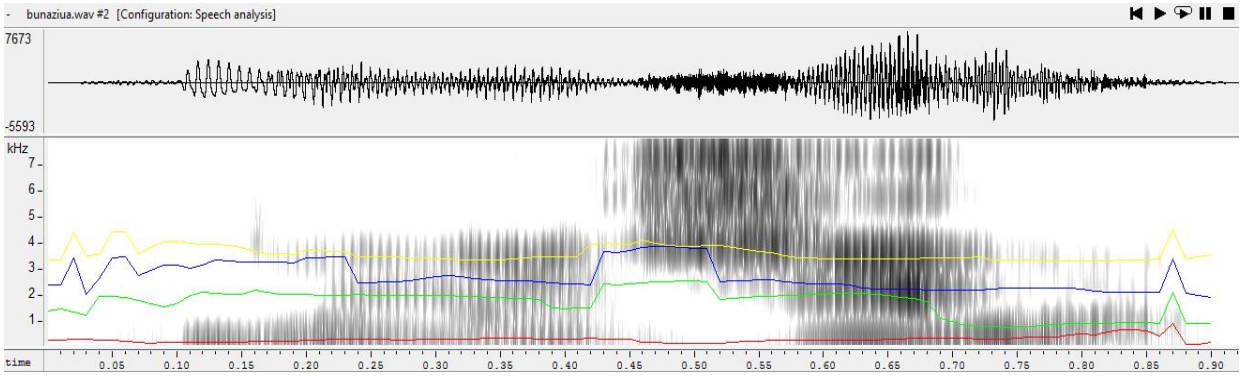

Figure 2.9 Formants shown for the Romanian utterance "bună ziua", plotted in colored lines

The formant with the lowest frequency is called F_1, the second F_2 and the third F_3. Most often the two first formants, F_1 and F_2, are enough to disambiguate the vowel is a spectrogram analysis. In spite of their phonetic significance, nowadays formant frequencies are rarely used as acoustic features for speech recognition, but it is possible to extract formant features through formant analysis for emotion detection, for example [Holmes, 1997].

2.5 Feature Extraction and Spectral Analysis

A speech recognition system depends on two basic stages, the preprocessing stage and the features extraction subsystem. The feature extraction sub-system parameterizes the speech waveform so that the relevant information (in this type of application, the information about the speech units) is enhanced and the non-relevant information is mitigated. The extracted information from speech files is unique and can later be used to compute some feature vectors which will be eventually modelled by the acoustic model. Speech signal is a quasistationary, slowly timed varying signal. Over a sufficiently short period of time (between 5 and 20ms), its characteristics are fairly stationary. However, over long periods of time, the signal characteristic change to reflect the different speech sounds being spoken. There are many techniques used to parametrically represent a voice signal for speech recognition tasks, for digital use.

There are many techniques used to parametrically represent a voice signal for speech recognition tasks, for digital use. These techniques include Linear Prediction Coding (LPC), and the Mel Frequency Cepstrum Coefficients (MFCC). Lately, a new technique, noise-robust PNCC features, that implies usage of power law (instead of log) and posteriorgram representation arose. We will quickly discuss each of this features ant their usage in speech recognition, but before that, we will introduce some mandatory spectral analysis techniques necessary for some of the most popular feature techniques, MFCCs.

Short-time Fourier Transform (STFT) is used in previous chapters to calculate and show spectrograms. However, doing a complete STFT sometimes is not feasible for the process of feature extraction, as discontinuities start to appear in the form of Gibbs phenomenon. So, frame

blocking and windowing is done to isolate a portion of the signal, then a Discrete Fourier Transform (DFT) is performed on that windowed signal much in the same way as in the STFT.

In frame blocking the speech signal is blocked into frames of *N* samples, with adjacent frames being separated by *M (M < N)*. The first frame consists of the first *N* samples. The second frame begins *M* samples after the first frame, and overlaps it by *N* - *M* samples and so on. This process continues until all the speech is accounted for within one or more frames. Typical values for *N* and *M* are *N* = 256 (which is equivalent to ~ 30 ms windowing) and *M* = 100.

To minimize the signal discontinuities at the beginning and end of each frame (Gibbs phenomenon), windowing is applied to each individual frame, to minimize spectral distorsion by tapering the signal to zero at the beginning and end of each frame.

Windowing is defined as:

$$w(n), 0 \le n \le N-1 \tag{2.11}$$

where *N* is the number of samples in each frame. The result of windowing applied to the signal is:

$$y_l(n) = x_l(n)w(n), \quad 0 \le n \le N-1 \tag{2.12}$$

The Hamming window is by far the most popular window used in speech processing, and is defined as follows:

$$\mathrm{w(n)} = \begin{cases} 0{,}54 - 0{,}46cos\left(\frac{2\pi n}{N-1}\right), n = 0, \dots, N-1 \\ 0, rest\ of\ the\ interval \end{cases} \tag{2.13}$$

One reason for the popularity of the Hamming window is the fact that its spectrum falls off rather quickly, so it allows for better isolation. However, its side-lobes (higher harmonics) stay quite flat and it covers most of the spectrum.

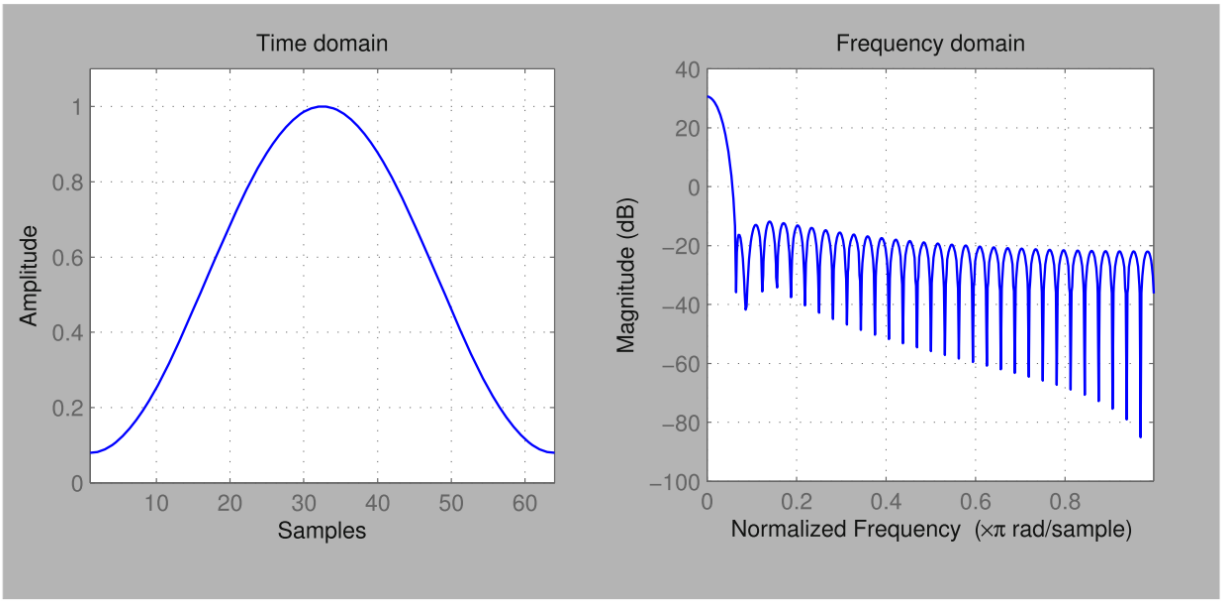

Figure 2.10 Hamming window

Other typical windows used in speech recognition are: Welch window, Triangular window, or Blackman window [Boldea, 2003]. In practice, the usual window used is about 20 ms and no shorter than 8 ms).

The next step in the processing of the speech data to be able to compute its spectral features is to take a Discrete Fourier Transform of the windowed data. This is done using the Fast Fourier Transform algorithm for every window:

$$X_t(e^{j\omega}) = \sum_{n=0}^{N-1} x_t(n) \cdot e^{-j\omega n} \tag{2.14}$$

This is the standard method for spectral analysis. The complexity of the algorithm decreases if we use the Discrete Fourier Transform. If we consider the values ω equally spaced (ex, $\omega = 2\pi k / N$), equation 2.14 becomes:

$$X_t(k) = X_t\left(e^{\frac{j2\pi k}{N}}\right), k = 0, \dots N-1 \tag{2.15}$$

If the number of samples N is chosen as a power of two ($N = 2^p$, where p is an integer), the complexity of the algorithm decreases at the order of $Nlog(N)$ by just using FFT. Now, from the sampled speech waveform, a series of time discrete spectral frames is obtained.

2.5.1 Mel Frequency Cepstral Coefficients (MFCC)

Mel frequency cepstral coefficients (MFCCs) are probably the most commonly used technique to represent the speech spectrum in ASR systems, and can be considered a baseline for performance comparison of feature sets [Stuttle, 2003]. MFCC's (Figure 2.11) are based on the known variation of the human ear's critical bandwidths with frequency. Filters spaced linearly at low frequencies and logarithmically at high frequencies have been used to capture the phonetically important characteristics of speech [Price, 2006]. The MFCCs are generated by first obtaining the speech spectrum as described in the previous section.

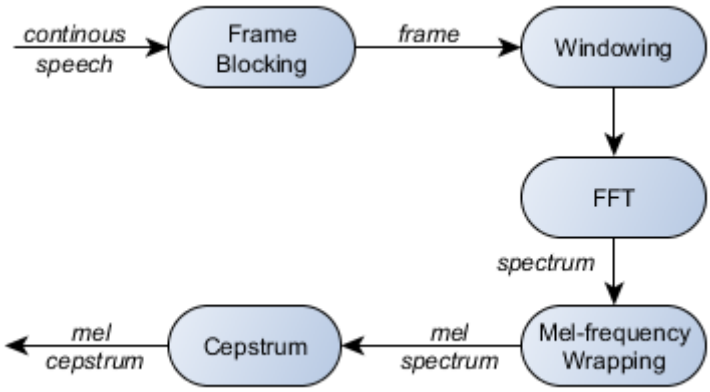

Figure 2.11 Block diagram of the MFCC feature extraction module

We reiterate how previous spectral analysis features help to obtain MFCCs. In the frame blocking section, the speech signal is broken into frames. The windowing block minimizes the discontinuities of the signal by tapering the beginning and end of each frame to zero. The FFT block converts each frame from the time domain to the frequency domain representation. In the Mel-frequency wrapping block, the signal is plotted against the Mel-spectrum. A number (M) of triangular shaped filter bin functions equally spaced on the Mel scale (see Figure 2.12) are taken from the magnitude spectrum:

$$f_{mel} = 1127log[1 + \frac{f_{Hz}}{700}] \tag{2.16}$$

This mimics the human hearing, as studies have shown that human perception of the frequency contents of sounds for speech signals does not follow a linear scale. The mel-frequency scale is a linear frequency spacing below 1000 Hz and a logarithmic spacing above 1000 Hz.

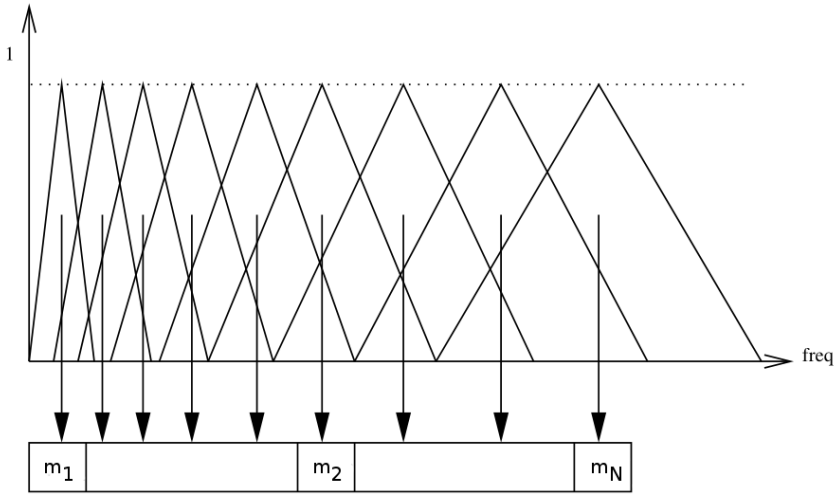

Figure 2.12 An example of Mel-spaced filter bank

Usually around 24 filter-banks are used to represent the spectrum. The log-spectral filter-bank outputs could be used for speech recognition. The problem, however, is that a high energy in a given filter-bank corresponds to a high energy in the surrounding filters, and the features are highly correlated. Finally, the cepstral coefficients, $c_i(t)$, are then calculated by taking the discrete cosine transform (DCT) of the Mel log energies:

$$c_i(t) = \sum_{b=1}^{M} log\big(m_b(t)\big)\, cos\big(\frac{i(b-0{,}5)\pi}{M}\big)] \tag{2.17}$$

Figure 4 shows the impact of the Mel-frequency wrapping on the speech utterance “zero”, pronounced in Romanian. In the first plot, most of the information is contained in the lower frequencies. This information is then amplified in the second plot (formant frequencies) through Mel filter banks.

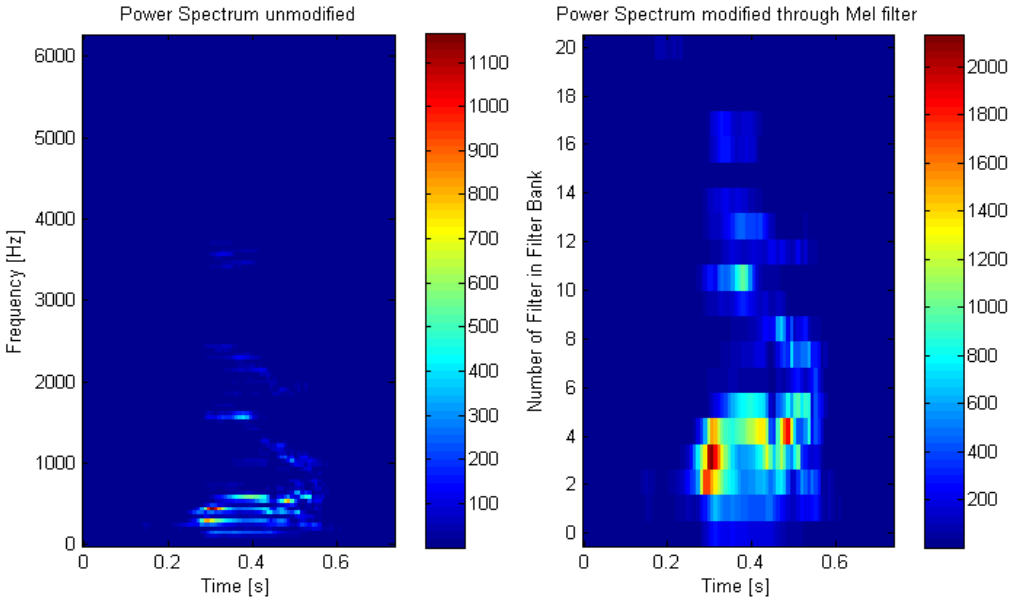

Figure 2.13 Spectrum plot of a speech file, before and after the mel-frequency wrapping block. Note that the spectrum is shown in a linear and not a logarithmic scale

After the above steps, for each speech frame, a set of mel-frequency cepstral coefficients are computed, which are called an "acoustic vector". The MFCCs success arises from the use of perceptually based Mel-spaced filter bank processing of the Fourier Transform and the particular robustness and the flexibility that can be achieved using the general cepstral analysis.

2.5.2 Perceptual Linear Predictive (PLP) Analysis

Perceptual linear prediction (PLP), introduced by [Hermansky, 1990], includes an auditory-inspired cube-root compression and uses an all-pole model to smooth the spectrum before the cepstral coefficients are computed [Cucu, 2011a]. The motivation of PLP is to closely model the psychoacoustics of hearing.

Three properties of the human auditory system are implemented in PLP:

- the nonlinear frequency response of the human ear;
- the critical bands in the cochlea;
- the non-linear amplitude response.

The PLP analysis is an extension of the Linear Prediction Coding (LPC) technique, but it is more effective because it takes advantage of some characteristics derived from the psycho-acoustic properties of the human ear [Stuttle, 2003]. The preprocessing that leads to the LP stage is very similar to the preprocessing which was discussed in the process leading to the MFCCs. This steps can be seen in Figure 2.14, where the block diagram of the PLP method is shown.

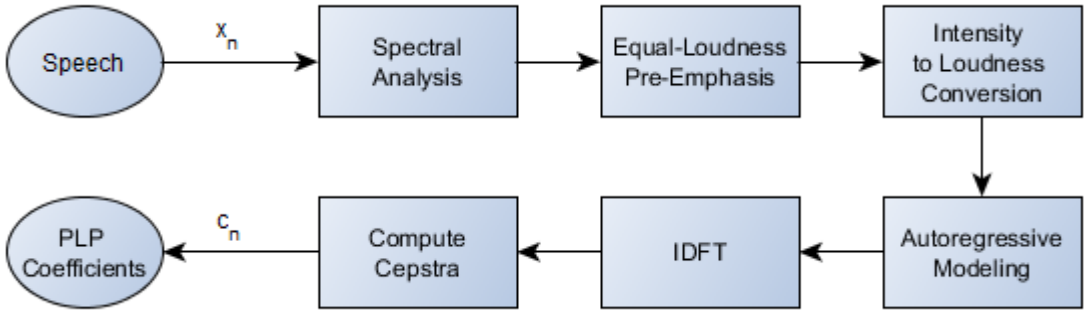

Figure 2.14 Block diagram for PLP feature vectors analysis

The first step in the PLP is identical to the spectral analysis which was described in the computation of MFCC features, with one observation: similar to the the Mel Frequency Warping that was done in previous section, [Hermansky, 1990] chooses to use the Bark scale, a related to, but somewhat less popular than Mel scale. So, the nonlinear frequency response of the human ear is approximated by warping the spectrum to the Bark frequency scale f_{bark}, as follows:

$$f_{bark} = log\left\{\frac{f_{Hz}}{600} + \left[\left(\frac{f_{Hz}}{600}\right)^2 + 1\right]^{\frac{1}{2}}\right\} \tag{2.18}$$

Using Mel-scaled triangular bins to model the critical bands has been equally successful in other implementations of PLP features [Woodland, 1998]. To model the variations in perceived loudness in the human auditory response an equal loudness function $E(\omega)$ is applied to the critical band filter-bank values, as follows in the next equation:

$$E(\omega) = \frac{(\omega^2 + 56{,}8 \cdot 10^6)\omega^4}{(\omega^2 + 6{,}3 \cdot 10^6)^2(\omega^2 + 0{,}38 \cdot 10^9)(\omega^6 + 9{,}58 \cdot 10^{26})} \tag{2.19}$$

Afterwards, a cube-root compression of the equal loudness bins is taken. Once the spectrum is obtained it is then converted back into the time domain and an auto correlative all pole LP

analysis is performed to obtain the PLP coefficients. The auto correlative function can be obtained from the inverse Fourier transform of the power spectrum and the perceptual linear prediction coefficients $c_N = [c_1, \ldots, c_N]^T$ are calculated from the prediction filter coefficients $[a_1, \ldots, a_n]$ from a prediction filter:

$$c_n = -a_n + \frac{1}{n}\sum_{i=1}^{T-1}(T-i)a_i p_{n-i} \tag{2.20}$$

where c_n is the n^{th} PLP coefficient. In some scenarios for speech recognition, these features slightly outperformed MFCCs (more noise robustness [Stuttle, 2003]).

2.5.3 Noise robust Power Normalized Cepstral Coefficient features (PNCC)

One of the most challenging contemporary problems is that recognition accuracy degrades significantly if the test environment is different from the training environment or if the acoustical environment includes disturbances such as additive noise, channel distortion, speaker differences, reverberation, and so on. In recent decades following the introduction of HMMs and statistical language models, the performance of speech recognition systems in noisy environments has dramatically improved. Nevertheless, most ASR systems still remain sensitive to the nature of acoustical environments, and their performance deteriorates rapidly in the presence of sources of degradation [Kim, 2010].

Many compensations algorithms have been introduced over the years to compensate for noisy channels. Many provided substantial improvement in accuracy for speech recognition in the presence of quasistationary noise, like: [Acero, 1990; Moreno, 1996; Sigh, 2002; Pujol, 2006]. But this approaches do not provide significant improvements in more difficult environments with transitory disturbances such as a single interfering speaker or background music [Kim, 2010].

Lately, a new technique, noise-robust PNCC features, that implies usage of power law (instead of log) and gamma tone filters (instead of triangular) arose. In theory, it should provide superior recognition accuracy over a broad range of conditions of noise and reverberation with a computational complexity that is comparable to that of traditional MFCC and PLP features. Development was motivated by a desire to obtain a set of practical features for speech recognition that are more robust with respect to acoustical variability in their native form, without loss of performance when the speech signal is undistorted.

PNCC features can be seen as a variant on MFCC feature extraction, but with different stages of the conventional algorithm replaced with auditory motivated elements. Firstly the triangular filter bank used by MFCC is replaced with a gamma tone filter bank. The novel aspects of the algorithm are the use of a Power Function Nonlinearity (replacing MFCC's log nonlinearity) and the use of Medium-Duration Power Bias Subtraction to suppress the effects of background excitation. The block diagram of the PNCC feature extraction can be seen in Figure 2.15, below.

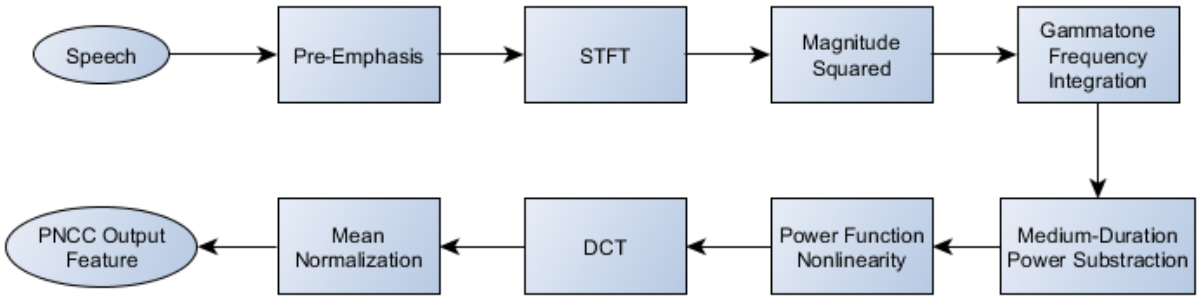

Figure 2.15 Block diagram for PNCC feature extraction

The nonlinearity of the human auditory system was discussed in earlier chapters and the use of a nonlinear function in feature extraction methods is common. MFCCs pass the filter outputs through a log nonlinearity. PNCC adopts a power function which aims to better model peripheral nonlinearities than a log function.

In [Kelly, 2010], quickly describes the main components of the PNCC extraction scheme: the graph related to auditory nerve firing rate is S-shaped [Zhang, 2001], by looking at the auditory models in that paper. He observed that for decibels below a certain threshold, the firing rate is almost constant. Above this, the increase in decibels with firing rate is almost linear, until it reaches a saturation point. If a log nonlinearity is adopted then there is no lower threshold. Thus small changes at a low power can result in large changes at the output of the log function.

With the help of the power function, when the input level is close to zero, so is the output level, as it is observed in the human auditory system. The power nonlinearity is described as follows:

$$y = x^{a_0} \tag{2.21}$$

where the best value of the exponent was calculated by [Kim, 2009] at $a_0 = 0{,}1$.

In the Medium-Duration Power Subtraction module, the algorithm subtracts a 'bias' from the speech segment that is assumed to represent an unknown level of background excitation. The newly adjusted power *P'(m, n)* of the *m*-th channel and *n*-th frame is given by equation:

$$P'(m,n) = \left(\frac{1}{2m_r + 1} \sum_{m'=\max(m-m_r,1)}^{\min(m+m_r,M)} w(m',n) \right) P(m,n) \tag{2.22}$$

where *P(m,n)* is the original power of the frame, and *w(m',n)* is the power normalization gain, given by the ratio of the normalized power to the average power of a frame. *M* is the total number of gammatone channels. In theory, PNCC features should provide better recognition accuracy than MFCC and PLP features, and next chapters will make a comparison in Spoken Term Detection tasks with all proposed features.

2.5.4 Posteriorgram Representation

Acoustic pattern-matching techniques have recently become prominent for automatically processing speech utterances, where no prior knowledge of the spoken language at hand is required. Conventional ASR systems use the phonetic transcription to build the acoustic models of the system vocabulary. The phonetic transcription of each word is used to concatenate hidden Markov models (HMMs) representing phoneme-level units (typically context-dependent phonemes). In some applications, a proper phonetic transcription cannot be easily obtained. Applications of such technology include, but are not limited to, query by-example search, spoken term detection, automatic word discovery or database retrieval applications. Obtaining content-aware acoustic features as independent as possible from speaker and acoustic environment variations is a key step in these scenarios [Anguera, 2012].

A phone posteriorgram is defined by a probability vector representing the posterior probabilities of a set of pre-defined phonetic classes for a speech frame, with entries summing up to one. By using a phonetic recognizer, each input speech frame is then converted to its corresponding posteriorgram representation.

There are multiple ways to obtain posterior features, depending on the system requirements:

- Gaussian posteriorgram: each class represents a component of a Gaussian mixture model (GMM) trained in an unsupervised way from a training data set or from the test data itself.

- Hidden Markov model (HMM) state posteriorgram: each class represents the state of a HMM modeling a language-specific phone.
- Multi-layer perceptron (MLP) estimated posteriors: the speech variability present in the features is reduced by applying the speech knowledge captured by the MLP on a training database.

Gaussian posterior features seem to be very a very popular way to obtain these features, as the GMM/HMM framework is a popular approach to speech recognition. The core idea is to train a Gaussian mixture model (GMM) without using any supervised annotation (transcriptions), and represent each speech frame by calculating a posterior distribution over all computed Gaussian components. Then a modified DTW matching algorithm can be used to evaluate the similarity between two speech segments represented by Gaussian posteriorgrams in terms of an inner-product distance. The entire process is completely unsupervised and does not depend on speakers [Zhang, 2013]. This has proven to be a success in spoken term discovery tasks [Zhang, 2010], and chapter 5 will present a series of experiments and evaluations on this task, done by the thesis author.

Other approaches use neural classifiers, such as Multilayer perceptrons to estimate the posterior probability of phonemes given the acoustic evidence. Neural nets employ huge parallel networks of many densely interconnected computational elements called neurons. Multi-layer neural networks consist of a large number of neurons. A MLP consists of multiple layers of nodes in a directed graph, with each layer fully connected to the next one. Except for the input nodes, each node is a neuron (or processing element) with a nonlinear activation function. For posteriors, each output unit of the MLP is associated with a particular HMM state, to allow these probabilities to be used as emission probabilities of a HMM system. Viterbi algorithm is then applied on the hybrid system to decode phoneme sequences and each time frame in the acoustic signal is associated with a phoneme in the decoded output. Then posterior probabilities of phonetic sound classes are estimated using a hierarchical configuration of MLPs [Thomas, 2011].

Recent developments in acoustic pattern-matching allow the processing of audio data without the need for large transcribed datasets, standard in most HMM-based speech processing algorithms, and posteriorgrams allow for obtaining features derived from the audio data that are content-aware and independent of speaker variability and changes in the acoustic conditions. More on how we used posteriors combined with BUT Neural TRAP`s system to output posteriorgrams in chapter 5, with a direct application in spoken term detection and discovery.

2.6 PHONETIC REPRESENTATION OF SPEECH

In most languages, the written text does not always correspond to its oral pronunciation, so that in order to describe it acoustically some kind of symbolic presentation is needed. Phonetics study speech sounds and their production, classification, and transcription. It treats the sounds as independent units, as if they would not contain any linguistic information [Huang, 2001]. Phonology is the study of the distribution and patterning of speech sounds in a language and of the rules governing pronunciation. As opposed to phonetics, phonology treats sounds as intertwined units, as such being an interface between phonetics and the superior linguistic units [Burileanu, 1999]. Every language has a different phonetic alphabet and a different set of possible phonemes and their combinations. A set of phonemes can be defined as the minimum number of symbols needed to describe every possible word in a language, and it varies with every language, due to complexity and different kind of definitions. The number of phonetic symbols is between 20 and 60 in each language [O'Saughnessy, 1987], but there are languages where it cannot be defined exactly [Lemmetty, 1999].

Sometimes phones are considered in context. Phonemes are abstract units and their pronunciation depends on contextual effects, speaker's characteristics and emotions. During continuous speech, the articulatory movements depend on the preceding and the following phonemes. The articulators are in different position depending on the preceding one and they are preparing to the following phoneme in advance. This causes some variations on how the individual phoneme is pronounced. Thus, a phoneme is strongly affected by its immediately neighboring phonemes, making this units context dependent. These variations are called allophones which are the subset of phonemes and the effect is known as coarticulation [Lemmetty, 1999]. Recognition accuracy can be significantly improved if there is enough training data to estimate these context-dependent parameters.

As we will see in chapter 3.3, statistical language models need to estimate the acoustic data for a given word sequence, $p(X|W)$, by generative approaches and sub-word units. And in this case, the speech unit used for modeling the acoustic data are phones described above, which are most commonly linked in models during the decoding process, to form words models and eventually word sequence models (which are finally used to estimate $p(X|W)$). This generative approach has been proven to work well with the Hidden Markov Model (HMM) mathematical apparatus [Baker, 1975; Rabiner, 1989; Jelinek, 1998]. More on statistical modeling in chapter 3.

Phones present the following characteristics, which gives them preferential status for a proper sub-word unit, to be used in ASR:

- Phones are accurate, as it represents the acoustic realization that appears in different contexts.
- Phones are trainable, so there is enough data to estimate the parameters of the unit, because sufficient occurrences for all phones can be found in just a couple thousand phrases.
- Phones are also generalizable, so that any new word can be derived from a predefined unit inventory for task-independent speech recognition. Moreover, they are also vocabulary independent by nature and can be trained on one task and tested on another.

However, in some cases the phonetic model is inadequate for ASR if it assumes that a phoneme is identical in any context. As stated at the beginning of this section, although we may try to say each word as a concatenated sequence of independent phonemes, these phonemes are not produced independently, because our articulators cannot move instantaneously from one position to another. Thus, the realization of a phoneme is strongly affected by its immediately neighboring phonemes. While word models are not generalizable, phonetic models over-generalize and, thus, it might lead to less accurate models if we do not make units context dependent. Context-dependent phonemes have been widely used for large-vocabulary speech recognition, thanks to its significantly improved accuracy and trainability. A context usually refers to the immediately left and/or right neighboring phones [Cucu, 2011a].

Regarding the classification of Romanian phonemes, there is still disagreement about the exact classification of phonemes, as it is not standardized yet [Pașca, 2012]. Our research group uses the proposed classification in [Cucu, 2011a], as shown in Table 2.1, where we employ our in-house notations for ease of use, but the correspondence to the IPA symbols is one-to-one.

The phonetic alphabet is usually divided in two main categories, vowels and consonants. Vowels are always voiced sounds and they are produced with the vocal cords in vibration, while consonants may be either voiced or unvoiced. Vowels have considerably higher amplitude than consonants and they are also more stable and easier to analyze and describe acoustically. Because consonants involve very rapid changes they are more difficult to synthesize properly.

The most basic classification of Romanian phonemes is as follows:

- Vowels: a, a1, e, i, i1, i2, o, u;
- Semivowels: e1, i3, o1, w;
- Consonants: k2, b, p, k, k1, g, g1, g2, d, t, f, v, h, j, s1, l, m, n, s, z, r, t1.

Table 2.1 Romanian 34-Phoneme Set [Pașca, 2012]

Phoneme (IPA symbol)	**Phoneme (system symbol)**	**Word Example (and translation to English)**
a	a	mare (sea/large)
ə	a1	pară (pear)
b	b	bicicletă (bicycle)
d	d	dinte (tooth)
e	e	vedere (sight)
e	e1	deal (hill)
f	f	fericit (happy)
g	g	gol (empty)
ʤ	g1	girafă (giraffe)
ɟ	g2	unghi (angle)
h	h	harnic (hard-working)
i	i	ține (to hold)
j	i1	tari (strong)
ɨ	i2	între (between)
j	i3	fiară (wild animal)
ʒ	j	jenă (embarrassment)
k	k	acord (agreement)
ʧ	k1	ceva (something)
c	k2	chiar (even)
l	l	lună (month)
m	m	mic (small)
n	n	neutru (neutral)
o	o	volei (voleyball)
o	o1	oase (bones)
p	p	papagal (parrot)
r	r	rinichi (kidney)
s	s	surpriză (surprise)
ʃ	s1	uşor (easy)
t	t	atent (careful)
ʦ	t1	aţă (thread)
u	u	unic (unique)
v	v	viteză (speed)
w	w	sau (or)
z	z	varză (cabbage)

2.7 CHAPTER CONCLUSIONS

This theoretical chapter started by offering and overview then describing the basic principles and proposed models for speech and hearing perception.

Also highlighted in this chapter were the fundamental characteristics of the speech signal, along with the main methods for processing and analyzing of the voice waveform. This was necessary in order to understand how speech is digitized, then numerically processed in order to extract speech features used in this thesis (MFCC, PNCC and Posteriorgram representation). These speech features are used in all proposed ASR techniques, starting from chapter 3, as HMMs or Neural Net (NN) approaches do not use directly the time-domain waveform to model the speech signal.

One more thing to add about some of the described features sets (MFCC and PLP) is that even though each set is computed on a short time frame of speech signal, it is well known that information embedded in the temporal dynamics of the features is also useful for recognition.

Typically two kinds of dynamics have been found useful for speech recognition:

- Velocity of the features (known as delta features), which is determined by its average first-order temporal derivative
- Acceleration of the features (also known as delta-delta features), which is determined by its average second-order temporal derivative.

Moreover, the total log energy of the feature and its derivatives have been proven to be useful for speech recognition. Consequently, speech recognition accuracy is substantially improved if the feature vectors are augmented with the first and second temporal derivatives of the acoustic features, thus adding some information about the local temporal dynamics of the speech signal to the feature representation [Furui, 1986].

To sum up, most commonly, ASR systems use a 39-dimensional feature vector, corresponding to twelve MFCCs plus energy, along with their first and second temporal derivatives [Cucu, 2011a].

CHAPTER 3

STATE OF THE ART IN SPEECH RECOGNITION

3.1 OVERVIEW OF ASR

Research in the field of automatic speech and speaker recognition has now spanned more than five decades [Furui, 2009]. After all these years of research and development, the problem of automatic speech recognition is still an open issue. To design a machine that mimics human behavior, particularly the capability of speaking naturally and responding properly to spoken language, in the context of a high degree of correlation in the spoken content, has intrigued engineers and scientists. The end goal of a perfect translation into a word sequence, very accurate and efficient, unaffected by speaker particularities, noisy environment or transmission channel, is very difficult to achieve, and many challenges are to be faced. Figure 3.1 shows a timeline of progress in speech recognition and multimodal understanding technology, over the past decades.

The earliest attempts to build ASR systems were made in the 1950s and 1960s. Various researchers tried to exploit fundamental ideas of acoustic phonetics (Bell Laboratories, NEC Laboratories), by using the formant frequencies measured / estimated from the speech signal, to successfully recognize isolated digits. Throughout the decades, digital signal processing advances coupled with increases in computational power, led to the introduction in the 1960's and 1970's of the advanced speech representations, based on LPC analysis and cepstral analysis methods. In 1980's, through the introduction of rigorous statistical methods based on Hidden Markov Models, a shift in methodology took place, from the more intuitive template-based approach (a

straightforward pattern recognition paradigm), towards a more rigorous statistical modelling framework. This was possible with significant research contributions from academia, private industry and the governments.

Nowadays, most practical speech recognition systems, commercial or for academic research, are based on the statistical framework developed in the 1980s, by Baker (1975), a team at IBM [Jelinek 1976; Bahl, 1983], and a team at AT&T [Levinson, 1983; Rabiner 1989]. HMMs are used because speech can be viewed as a stationary signal or a short-time stationary signal, under certain conditions. For short time-periods (up to 20ms), speech can be approximated as a stationary process, and analyzed. This way, speech can be thought of as a Markov model for many stochastic purposes. Significant additional improvements were made during the 90s, in the field of pattern recognition, with focus on the optimization problem, involving minimization of the empirical recognition error.

The dominance of GMM-HMM in acoustic modelling, led to an ecosystem of speaker adaptation and front-end processing techniques, tailored to maximize the performance under this model. This was hard to challenge over time, until very recently, with a new competing acoustic modelling approach: Deep neuronal network acoustic model for large vocabulary continuous speech recognition systems. [Dahl, 2012] reported a 33% relative improvement in WER over a discriminatively trained GMM-HMM on a 300 hour English conversational telephone transcription task. “Deep” comes from using more than one hidden layer, typically three to five, to model context-dependent output distributions directly. In contrast to HMMs, Neural Networks make no assumptions about feature statistical properties. Neural Networks allow discriminative training, in a natural and efficient manner, when used to estimate the probabilities of a speech feature segment. Neural networks have been used in many aspects of speech recognition, such as phoneme classification [Waibel, 1989] recognition of isolated words [Wu, 1993] and speaker adaptation.

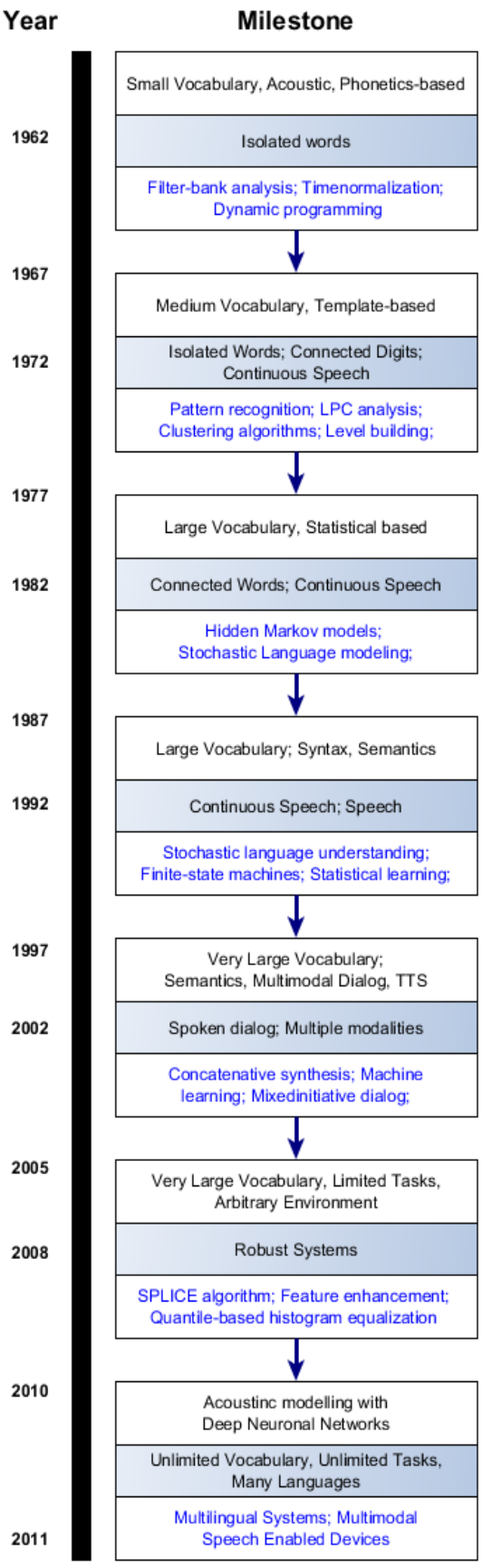

Figure 3.1 Milestones in Speech Recognition, adapted [Juang, 2005]

In Romania, the interest for automatic speech recognition and processing manifested since three decades ago, but studies became systematic after 1980. Research teams were organized in major academic centers, such as Bucharest (prof. Corneliu Burileanu, SpeeD group), Cluj (prof. Gavril Toderean, prof. Mircea Giurgiu), Iași (prof. Horia-Nicolai Teodorescu) and Timișoara (prof. Marian Boldea). Areas of interest include automatic speech recognition, speaker recognition and identification, voice synthesis, speech coding, natural language processing (Burileanu et al., 2004), spoken term detection and lately document indexing/retrieval. Besides HMM, other known strategies used by Romanian authors for speech recognition are the neural network connectionist ones, with fuzzy sets [Dumitru, 2008]. Lastly, there are the hybrid methods. An example is the Fuzzy-HMM approach, based on fuzzy integrals. Fuzzy measures have an essential property: monotonicity with respect to set inclusion, is far weaker than the usual additive property for probability measures [Militaru, 2014].

This section offered a brief overview in the field of speech recognition. One mention should be that, like we stated in the introductory chapter, in recent years, the scientific community also focused on increasing the intelligibility of the ASR output, through methods like punctuation and capitalization restoration [Gravano, 2009], robust diacritics restoration, in the context of high recognition accuracy and performance.

3.2 ARCHITECTURE OF A SPEECH RECOGNITION SYSTEM

As stated in the overview section of this chapter, the most common approach to the problem of classifying speech signals in ASR is statistical, by the use of Hidden Markov Models. One advantage of this statistical method of dealing with audio for pattern recognition, in HMM`s, is that it allows a number of techniques for adapting and extending the models.

If a statistical model is to be used, the goal is to find the most likely word sequence W^* given the recorded acoustics X:

$$W^* = arg \max_W p(W|X) \tag{3.1}$$

$$W^* = arg \max_W \frac{p(X|W)P(W)}{p(X)} \tag{3.2}$$

$$W^* = arg \max_W logp(X|W) + \log(P(W) \tag{3.3}$$

Equation (3.1) specifies the most probable word sequence as the one with the highest posterior probability given the acoustics and the model. Equation (3.2) is a result of equation (3.1), through the application of Bayes's theorem, to compute this posterior probability. Since $p(X)$, the probability of the speech utterance, is independent of the word sequence, it can be ignored. Consequently, equation (3.2) becomes equation (3.3) after applying logarithm. P denotes a probability and p denotes a probability density function.

From equation (3.3), the problem of searching for the most likely sequence of words in an utterance may be split into two separate components: *language modelling*, which is concerned with estimating the prior probability of a word sequence $P(W)$, and *acoustic modelling*, in which the likelihood of the acoustic data given the words, $p(X|W)$, is estimated. The parameters of both of these models are learned from large data corpuses. Obtaining the optimal word sequence W^* is the search / decoding problem.

The two models can be constructed independently as shown in Figure 3.2, but will be used together to decode a speech utterance as specified in Equation 3.3. Figure 3.2 presents the architecture of an ASR system and also shows the methods and type of data required in the training phase. Usually, all the processes involved in a typical ASR system are organized in blocks for future extensibility, as Figure 3.2 shows, but also for ease of development and portability.

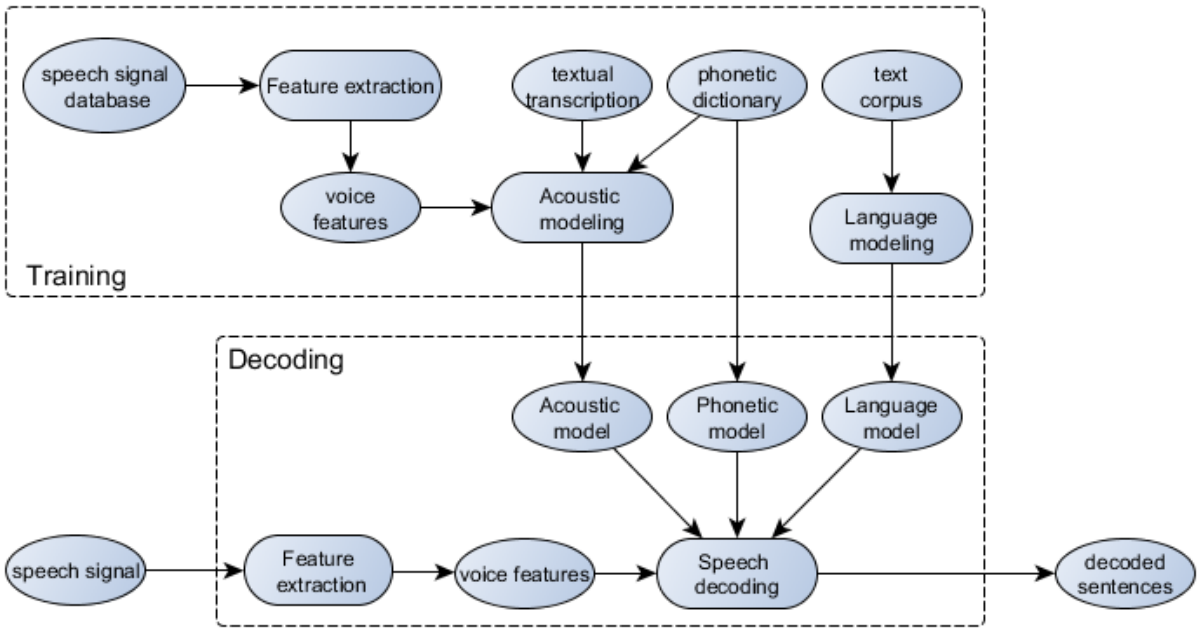

Figure 3.2 General architecture of a speech recognition system

The language model *P(W)*, models a word sequence by providing a predictive probability distribution for the next word based on a history of previously observed words. Since this probability distribution does not depend on the acoustics, language models may be estimated from large textual corpora. The text corpus also plays an important role in ASR, especially in the training / decoding part of the architecture. A speech corpus (or spoken corpus) is a database of speech audio files and text transcriptions. In Speech technology, speech corpora are used, among other things, to create acoustic models (which can then be used with a speech recognition engine).

We will also introduce acoustic modeling, and the development of systems for the recognition of conversational speech. In particular, the trainable hidden Markov/Gaussian mixture model (HMM/GMM), for acoustic modeling. To achieve high recognition accuracy and performance, an ASR system should use all the available acoustical information derived from the training phase, but also information about the language at hand. Besides the acoustic model and the language model which have been mentioned in the above paragraphs, the general ASR architecture also includes a phonetic model. This is due to the fact that, for large vocabulary systems, the acoustic model does not model all the words in the vocabulary, but sub-words units such as phonemes, introduces in Chapter 2. The phonetic model is most of the times a pronunciation dictionary which maps the words in the vocabulary to their phonetic representation.

Moving forward, this chapter will continue with an in-depth analysis of several blocks in Figure 3.2 and describe various language and acoustic modeling issues.

3.3 LANGUAGE MODELLING

The language model *P(W)* models a word sequence by providing a predictive probability distribution for the next word based on a history of previously observed words. The sequence of words is selected by the recognizer so that it maximizes the product between the probabilities of observing the acoustic evidence *X* when the speaker utters *W*, *P(X|W)*, and the sequence of words *W* that will be utter, *P(W)* in a given task. The first probability is estimated by the acoustic models and the second one is estimated by the language model. The goal of the language model is to model the sequence of words in the context of the task being performed by the ASR system.

In continuous speech recognition, the incorporation of a language model is crucial to reduce the search space of sequence of words [Adami, 2010].

Since this probability distribution $P(X|W)$ does not depend on the acoustics, language models may be estimated from large textual corpora. In general, the purpose of any type of language model is to estimate how likely is a sequence of words $W = w_1, w_2, \ldots, w_n$, going to be a sentence in the source language, to help the acoustic decoding in the decision process.

The language model $P(W)$ can be decomposed as:

$$P(W) = P(w_1, w_2, \ldots, w_n) \tag{3.4}$$

$$P(W) = P(w_1)P(w_2|w_1)P(w_3|w_1, w_2) \ldots P(w_n|w_1, w_2, \ldots, w_{n-1}) \tag{3.5}$$

$$P(W) = \prod_{i=1}^{n} P(w_i|w_1, w_2, \ldots, w_{i-1}) \tag{3.6}$$

where $P(w_i|w_1, w_2, \ldots, w_{i-1})$ is the conditional probability that will occur given the previous word sequence $w_1, w_2, \ldots, w_{i-1}$.

This means that the task of estimating the probability of the word sequence W is split into several tasks of estimating the probability of one word given a history of preceding words. Due to computational reasons, the history of preceding words cannot extend to include an indefinite number of words and has to be limited to an integer m (3 to 5) words. Only a limited number of previous words affect the probability of the next word. The conventional *n-gram* language model, which approximates the history as the immediately preceding $n - 1$ words, has represented the state of the art for large-vocabulary speech recognition for about 25 years [Renalds, 2010]. Most commonly, trigram language models are used. They consider a two-word history to predict the third word. This requires the collection of statistics over sequences of three words, so-called 3-grams (trigrams).

But for simpler tasks, like a limited vocabulary command and control system, Finite State Grammar model might be a better match, as we will show in this chapter experiments.

3.3.1 Finite State Grammar language modelling

Building a language dictionary is a difficult task, because a potential speaker can theoretically utter any succession of words, in any order that has meaning in that particular language. Therefore, the language model block should contain virtually all words that can be spoken by anyone in that language, and the model should be able to correctly estimate the probabilities of occurrence for each possible sequence of words. Accordingly, this makes statistical *n-gram* language models very suitable for general ASR systems. If the recognition scenario puts restrictions on the vocabulary that can be used, or the successions of words for that special task, then the *n-gram* model is not so suitable anymore, because the effort to train the model is not justified.

In this case, a succession of valid words and their probabilities of occurrence can be specified directly by using a formal language model, like finite state grammar (FSG). In formal language theory, a language is a set of strings. A string is just a sequence of symbols chosen from an agreed-upon set of symbols, called the vocabulary or lexicon. The idea of generative grammar is to use grammars to define a set that closely resembles a natural language. For instance, all and only the acceptable English sentences. However, not all sets are definable by all types of grammars. A finite state grammar is a graph model in which the nodes represent the language words, and transitions between words are the arcs of the graph. This type of language model explicitly specifies all sequences of words allowed by the recognition task. Moreover, each arc may be assigned a cost specifying the probability that a word is preceded by another (in other

words, the probability of the two sequences of words). To define ("generate") strings over {*a,b,c,d*} in alphabetical order, let *S* be the start symbol of the grammar in 3.7. The general form for a finite state grammar is that any grammar having only rules of the form $A \rightarrow bC$ where *A*, *B* are nonterminals and *b* is a terminal has a corresponding finite state machine. Given a string, if a path can be found through the machine, the string is generated by the grammar and vice versa [MSU, 2003].

$$\begin{aligned} &S \rightarrow a\ S1 \\ &S \rightarrow b\ S2 \\ &S \rightarrow c\ S3 \\ &S \rightarrow d \\ &S1 \rightarrow b\ S2 \\ &S1 \rightarrow c\ S3 \\ &S1 \rightarrow d \\ &S2 \rightarrow c\ S3 \\ &S2 \rightarrow d \\ &S3 \rightarrow d \end{aligned} \tag{3.7}$$

Grammar in 3.7 has a corresponding finite state machine that recognizes all and only the sentences it generates, as shown in Figure 3.3.

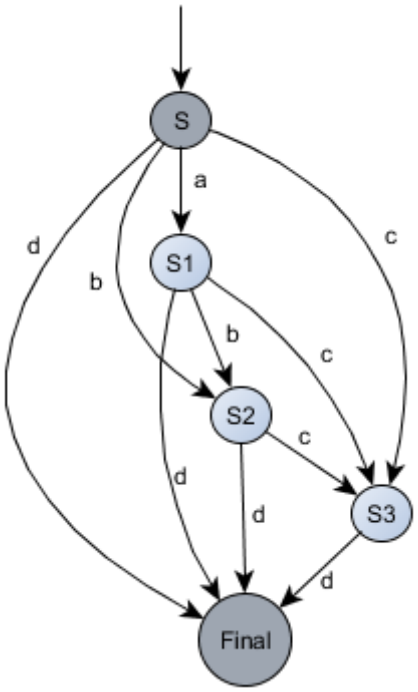

Figure 3.3 Corresponding Finite State Grammar for 3.7 grammar

Figure 3.4 shows the finite state grammar for a simple connected digits recognition task. 14 nodes make up the model, with only 10 nodes representing the digits. The other four are used for "entering" and "leaving" the graph, respectively for a back trace transition. The transitions show the way to trace this grammar and the word sequences allowed: in every audio clip one or more digits can be spoken.

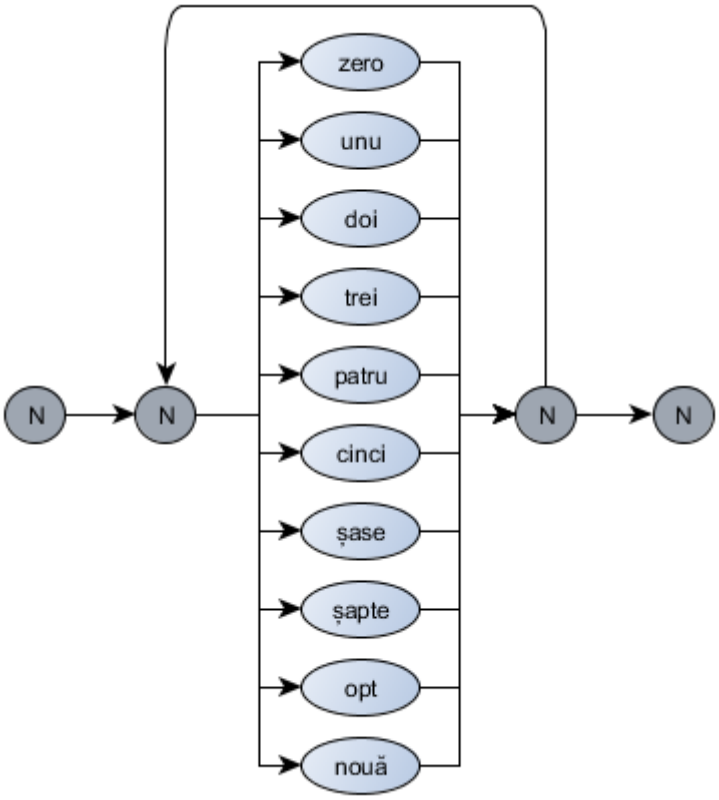

Figure 3.4 Finite State Grammar example for a digit recognition task

As shown above, for a simple task, a Finite State Grammar model might be a better match, and computationally friendly.

3.3.2 N-gram language modelling

An *n-gram* language model is constructed by estimating the probabilities discussed in section 3.3, using a large enough text corpus. For example, in the case of a bigram language model, the probabilities $p(w_j|w_i)$ for every pair of words (w_i, w_j) have to be estimated. In order to compute this probability, we use the maximum likelihood (ML) principle and count how often w_i is followed by w_j as opposed to other words:

$$p(w_j|w_i) = \frac{count(w_i, w_j)}{\sum_w count(w_i, w)} \tag{3.8}$$

From 3.8, the probability for the sequence "*W=pisicile dorm foarte mult*" can be estimated as follows, for a bigram model:

$$P(W)=P(pisicile|<s>)P(dorm|pisicile)P(foarte|dorm)P(mult|foarte)P(</s>|mult) \tag{3.9}$$

For a trigram language model one needs to estimate all the probabilities $P(w_k|w_i,w_j)$:

$$p(w_k|w_i, w_i) = \frac{count(w_i, w_j, w_k)}{\sum_w count(w_i, w_j, w)} \tag{3.10}$$

A large amount of training data (typically hundreds of millions or even billions of words) is needed to accurately estimate these probabilities. Also, higher order n-gram language models require larger amounts of training data, and *data sparseness problem*, which is a typical problem for any statistical system, has to be taken into account. No matter how large the training corpus is, there will be n-grams which will not be seen within it, because equation 3.9 and 3.10 will assign the maximum likelihood probability to 0, but may appear in the evaluation or test corpus.

Moreover, there are n-grams which occur only a few times in the training corpus, and this issue will become more severe with higher order n-grams [Cucu, 2011a].

In these scenarios, the probabilities which were estimated based on the empirical counts that are observed in the training corpus, are very rough estimates and need to be adjusted, by means of *smoothing n-grams* or *back-off methods* [Jelinek, 1998].

Smoothing methods subtract probability mass from seen n-grams and redistribute it to unseen n-grams. As we will see below, there are several smoothing methods which tend to particularize the redistribution of probability mass given some specific reasons.

A basic smoothing method, *add smoothing*, simply adds a fixed number to every n-gram count. This means that even n-grams which do not appear in the training corpus, but are made up of words in the vocabulary, will be assigned non-null probabilities. This gives undue credence to n-grams that do not appear in the training corpus [Koehn, 2010], and one quick simple fix would be to add a smaller number α, empirically estimated on a held-out corpus. Equation 3.11 presents a technique to interpolate trigram, bigram and unigram relative frequencies using this smoothing method [Adami, 2010]. Considering a trigram model ($n=3$), the interpolation is defined as:

$$P(w_i)|w_{i-2}, w_{i-1}) = \alpha_3 \frac{F(w_{i-2}, w_{i-1}, w_i)}{F(w_{i-2}, w_{i-1})} + \alpha_2 \frac{F(w_{i-1}, w_i)}{F(w_{i-1})} + \alpha_1 \frac{F(w_i)}{\sum F(w_i)} \tag{3.11}$$

where the non-negative weights satisfy $\alpha_1 + \alpha_2 + \alpha_3 = 1$. Applying the cross-validation principle, the α weights are obtained. An issue with this approach is that it uses information from lower-order distributions even when the estimate of the probability of an n-gram is reliable.

Deleted interpolation method splits the training corpus into two parts. It uses one part to estimate n-gram counts (c) and the second part to see how often we expect to see (c) in a real application. Secondly, by switching the roles of the two parts and interpolating the results, this method comes up with better expected counts than add smoothing method above.

Good Turing is another smoothing method that uses actual counts (c) and count-of-counts statistics (N_c is the number of n-grams which occur c times in the training corpus) to adjust the counts (c^*) for all seen and unseen n-grams. This method is not very reliable for large c, for which N_c is typically 0. This drawback can be solved by simply not adjusting the counts for frequent n-grams. The adjusted counts (c^*) can be calculated as follows:

$$c^* = (c+1)\frac{N_{c+1}}{N_c} \tag{3.12}$$

Another approach to solve data sparseness is to use several language models, through *backoff smoothing* methods. This has several particular advantages, and one would be to create an interpolated language model that may benefit from all its constitutive parts. For example, higher order n-grams may provide valuable additional context, but lower order n-grams are more robust. If several orders (1, 2 and 3) n-gram language models p_n have been already built, an interpolated language model p_I can be constructed by linearly combining all the probabilities. Through this interpolation, we provide a better smoothing technique, and one very famous method is the Katz smoothing (or Katz backoff) [Chen, 1999]. This method reduces (using a discounting factor) the unreliable probability estimates given by the observed frequencies and redistributes the discounted probability mass among the n-grams that never occurred in the training data. For a bigram model, Katz smoothing is defined by the following equation:

$$P_{Katz}(w_i|w_{i-1}) = \begin{cases} \dfrac{F(w_{i-1},w_i)}{F(w_{i-1})} \; if \; r > k \\ d_r \dfrac{F(w_{i-1},w_i)}{F(w_{i-1})} \; if \; 0 < r \leq k \\ \alpha(w_{i-1})P(w_i) \; if \; r = 0 \end{cases} \tag{3.13}$$

where r is the count for an n-gram w_{i-1}, w_i, k is a count threshold (5-8), d_r is a discount coefficient, and α is a normalization coefficient defined by:

$$\alpha(w_{i-1}) = \frac{1 - \sum_{w_i:r>0} P_{Katz}(w_i|w_{i-1})}{1 - \sum_{w_i:r>0} P(w_i)} \tag{3.14}$$

We use Maximum Likelihood (ML) estimate when n-gram count exceeds *k* threshold. When the count is below the threshold and above zero, the same ML count is used but weighted by a discount factor. The discounted probability mass is then distributed among the zero-count bigrams according to the next lower-order distribution, e.g., unigram model. The discount factor is based on the Good-Turing estimate, an estimate that adjusts the count of an n-gram by the n-grams that have the same count.

Another backoff algorithm is the Kneser-Ney method [Chen, 1999], that uses a modified backoff distribution based on the number of contexts where each word occurs in, rather than the number of occurrences of the word. For a bigram model, the Kneser-Ney smoothing is defined as:

$$P_{KN}(w_i|w_{i-1}) = \begin{cases} \dfrac{\max\{F(w_{i-1},w_i) - D, 0\}}{F(w_{i-1})} & if \; F(w_{i-1},w_i) > 0 \\ \alpha(w_{i-1})P_{KN}(w_i) & otherwise \end{cases} \tag{3.15}$$

where $P_{KN}(w_i)$ is the number of unique words preceding w_i. The normalization coefficient α is defined by:

$$\alpha(w_{i-1}) = \frac{1 - \sum_{w_i:F(w_{i-1},w_i)>0} \frac{\max\{F(w_{i-1},w_i)-D,0\}}{F(w_{i-1})}}{1 - \sum_{w_i:F(w_{i-1},w_i)>0} P_{KN}(w_i)} \tag{3.16}$$

An extension to this method is the modified Kneser-Ney smoothing [Chen, 1999], which uses a method called absolute discounting to reduce the probability mass for seen events. More information on smoothing techniques can be found on [Chen, 2000; Goodman, 2001].

3.4 ACOUSTIC MODELLING WITH STATISTICAL HMM/GMM FRAMEWORK

For large vocabulary systems, the acoustic model does not model all the words in the vocabulary. Sub-words unit such as phonemes described in Chapter 2 are used in this case, to map the words in the vocabulary to their phonetic representation (phonetic model). Sometimes phones are considered in context. "Thus, a phoneme is strongly affected by its immediately neighboring phonemes, making this units context dependent. Recognition accuracy can be significantly improved if there is enough training data to estimate these context-dependent parameters. Both phonetic and sub-phonetic units have the same benefits, as they share parameters at unit level. Parameter sharing is extended to subphonetic models, to treat the state in phonetic hidden Markov models as the basic subphonetic unit, a senone [Huang, 1993]. Huang and Hwang [Huang, 1993] further generalized clustering: senones are constructed by clustering the state dependent output distributions across different phonetic models. Each cluster thus represents a set of similar Markov states and is called a senone. A sub-word model is thus composed of a sequence of senones after the clustering is finished. A dictionary of pronunciations is used to build word models from subword models, and models of word sequences are constructed by concatenating word models, thus enabling information to be shared across this models: the number of distinct HMM states in

a system is determined by the size of the set of subword units. This approach is illustrated in Figure 3.5 "[Clark, 2010].

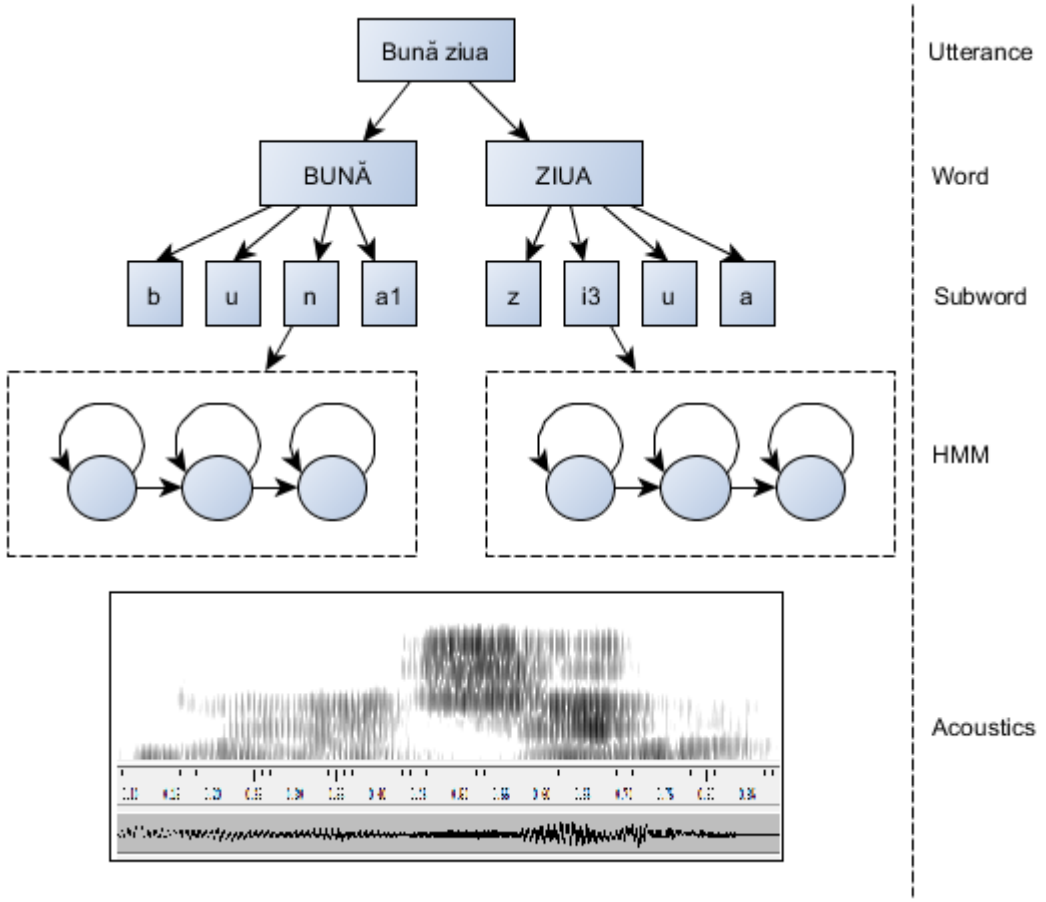

Figure 3.5 Representation of a HMM-based hierarchical modeling of speech, adapted [Clark, 2010]

Therefore, a senones dependence on context could be more complex than just left and right context, it can rather be defined by a complex function with a decision tree. Phones vary enormously, they are influenced by phones on either side, because of the articulators (tongue, lips) movements during speech production process. Therefore, an articulator may start moving during one phone to get into place in time for the next phone, and so on. Context dependent phones capture an important source of variation, and are a key part of modern ASR systems. But context-dependency also introduces the same problem found in language modelling: training data sparsity. The more complex the model to be trained, the less likely to have seen enough observations of each phone type to train on.

Consequently, the acoustic model consists of a set of phones models which are linked, during the decoding process, to form words models and eventually word sequences models, which are finally used to estimate $p(X|W)$. This generative approach has been proven to be very well served by the Hidden Markov Model (HMM) mathematical apparatus [Baker, 1975; Poritz, 1988; Rabiner, 1989].

3.4.1 Hidden Markov Modeling (HMM)

HMMs do not use directly the time-domain waveform to model the speech signal. We reiterate some of the preprocessing steps discussed in Chapter 2.3, related to preparing the analog signal for use with the HMM framework. As Figure 3.2 has shown, a feature extraction block is employed that extracts unique information from speech files, which can later be used to compute some feature vectors. These vectors will be eventually modeled by the acoustic model. Because

speech signal is a quasistationary, slowly timed varying signal, over a sufficiently short period of time (between 5 and 20ms), its characteristics are fairly stationary. However, over long periods of time, the signal characteristic change to reflect the different speech sounds being spoken.

There are many techniques used to parametrically represent a voice signal for speech recognition tasks, for digital use. Several types of speech features and techniques, which can be extracted out of these frames with the purpose to model speech, have been studied in Chapter 2.6. These techniques include Linear Prediction Coding (LPC), and the Mel Frequency Cepstrum Coefficients (MFCC). Lately, new techniques, like Posteriorgrams used in audio template-matching algorithms or noise-robust PNCC features that implies usage of power law (instead of log) and gammatone filters (instead of triangular), arose.

Previous paragraphs reminded us the features types that are extracted "out of the speech signal, for further modelling (training phase) or for speech recognition (decoding phase)." The approach for modelling basic speech units (phones) makes use of the HMM/GMM framework, introduced in this section. A HMM is a probabilistic finite state automaton, consisting of a set of states connected by transitions, in which the state sequence is hidden. Instead of observing the state sequence, a sequence of acoustic feature vectors is observed, generated from a Probability Density Function (PDF) attached to each state. This is why the Markov process is considered to be "hidden" – the state sequence is not directly available to the observer. This probability density functions are usually a Gaussian mixture models (GMM) density distributions that characterizes the statistical behavior of the feature vectors within the states of the model [Rabiner, 2007].A more detailed representation of an HMM is presented in Figure 3.6. As the figure shows, an HMM is characterized by these parameters:

- States: a set of states $Q = q_1 q_2 \ldots q_N$;
- Transition probabilities: a set of probabilities $A = a_{11} a_{12} \ldots a_{NN}$. Each $a_{ij} = p(q_j|q_i)$ represents the probability of transitioning from state i to state j;
- Observation likelihoods: a set of observation likelihoods $B = b_i(x_t) = p(x_t|q_i)$, each expressing the probability of an observation x_t being generated from the state i.

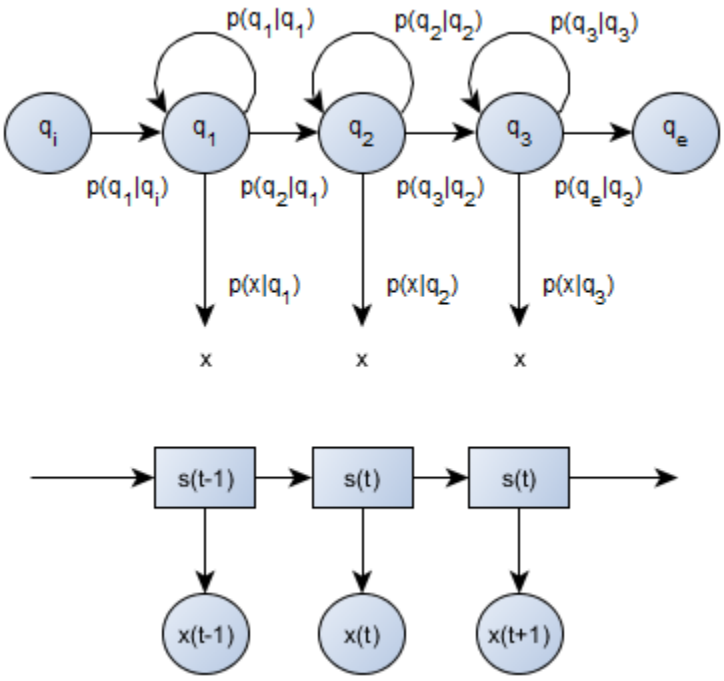

Figure 3.6 Representation of a HMM as a parameterized stochastic finite state automaton and in terms of probabilistic dependences between variables

In speech recognition, the models are created to disallow arbitrary transitions, just as Figure 3.6 shows, to model the sequential nature of speech, placing strong constrains on transitions backward or skipping transitions. The use of self-loops allows a sub-phonetic unit to repeat so as to cover a variable amount of the acoustic input.

The observation likelihoods are probability density functions, for a state q_i is a d-dimensional Gaussian, parameterized by a mean vector μ_i and a covariance matrix Σ_i as follows:

$$b_i = \frac{1}{(2\pi)^{d/2} |\sum_i|^{1/2}} \exp(-\frac{1}{2}(x-\mu_i)^T \Sigma_i^{-1}(x-\mu_i)) \quad (3.17)$$

where d equals 39 for a typical acoustic vector comprising 12th-order MFCCs plus energy, with first and second temporal derivatives, as discussed in chapter 2.

Modeling speech using hidden Markov models makes two main assumptions [Clark, 2010]:

- Markov process: the state sequence in an HMM is assumed to be a first-order Markov process, in which the probability of the next state transition depends only on the current state: a history of previous states is not necessary.
- Observation independence: all the information about the previously observed acoustic feature vectors is captured in the current state: the likelihood of generating an acoustic vector is conditionally independent of previous acoustic vectors given the current state.

These two assumptions may lead to an unrealistic model of speech, but they are needed due to the mathematically and computationally simplifications they bring. The estimation and decoding problems cannot be addressed, or can be addressed in a very complicated way without these assumptions. Nevertheless, the last two decades of HMMs success in speech signal modeling prove that these "limitations" are not so important. The HMM decoding issue, like finding the most likely sequence of states that have generated a sequence of observations, is solved by a variant of the Viterbi algorithm, and the various parameters of a HMM/GMM system are estimated using Forward-Backward algorithm (Baum-Welch). More on the search problem in the sections to come.

All of the above aspects of the HMM paradigm play a crucial role in acoustic modelling for all general statistical based ASR systems.

3.4.2 Model Design and states

In this section we go a little more in depth with aspects of model design and states in Hidden Markov Modelling. We start from the definition of HMMs that are basically first-order discrete time series with some hidden information. "Namely, the states of the time series are not the observed information, but they are related through an abstraction to the observation. The existence of this indirect abstraction is what gives HMM the hidden qualifier. Take away the hidden aspect and we are left with basic Markov chains. Although the process is a discrete time series, the observations, which are the interesting part of the information, may be in the form of discrete as well as continuous random variables. In other words, an HMM is a Markov chain with the capacity to contain extra information, either associated with its states or its transitions. [Beigi, 2011]".

Mathematically, we consider Markov models as randomly determined state machines with a finite set of N states. Given a pointer to the active state at time t the selection of the next state has a constant probability distribution. Thus the sequence of states is a stationary stochastic (randomly determined) process. An n^{th} order Markov assumption is that the likelihood of entering a given state depends on the occupancy in the previous n states. In speech recognition a 1st order

Markov assumption is usually used. The probability of the state sequence $q_T=(q_1, \dots q_T)$ is given by:

$$P(q_T) = P(q_1)\prod_{t=2}^{T} P(q_t|q_1, \dots, q_{t-1}) \tag{3.18}$$

and using the first order Markov assumption we can approximate this:

$$P(q_T) \cong P(q_1)\prod_{t=2}^{T} P(q_t|q_{t-1}) \tag{3.19}$$

The observation sequence is given as a series of points in vector space $X_T = \{x(1), \dots, x(T)\}$ or alternatively as a series of discrete symbols. Markov processes are generative models and each state has associated with it a probability distribution function (pdf) for the points in the observation space. The extension to "hidden" Markov models is that the state sequence is hidden, and becomes an underlying unobservable stochastic process. The state sequence can only be observed through the stochastic processes of the vectors emitted by the state output probability distributions. Thus the probability of an observation sequence can be described by:

$$P(X_T) \cong \sum_{Q_T} p(X|q_T)P(q_T) \tag{3.20}$$

where the sum $\sum_{QT}$ is over all possible state sequences q_T through the model and the probability of a set of observed vectors, $p(X_T|q)$, can be defined by:

$$p(X_T|q_T) = \prod_{t=1}^{T} p(x(t)|q_t) \tag{3.21}$$

Using a HMM to model a signal makes several assumptions about the nature of the signal. One is that the likelihood of an observed symbol is independent of preceding symbols (the independence assumption) and depends only on the current state q_t.

Figure 3.6, in the upper part, shows the topology of a typical HMM used in speech recognition. Transitions may only be made to the current state or the next state, in a left-to right fashion. In common with the standard HMM toolkit (HTK) software terminology conventions, the topology includes non-emitting states for the first and last states. These non-emitting states are used to make the concatenation of basic units simpler. HTK toolkit will be briefly described in section 3.8. We reiterate the form in which HMMs can be described by these set of parameters (previous section briefly mentioned these parameters):

- States: HMMs consist of N states in a model; the pointer ($q_t=i$) indicates being in state i at time t.
- Transitions: A transition matrix A is defined, that gives the probabilities of traversing from one state to another over a time state:

  $$a_{ij} = P(q_{t+1} = j|q_t = i) \tag{3.22}$$

  The form of the matrix can be constrained such that certain state transitions are not permissible, as shown in Figure 3.6. Additionally, the transition matrix has the following constraint:

  $$\sum_{j=1}^{N} a_{ij} = 1, \qquad a_{ij} \geq 0 \tag{3.23}$$

- State emissions: each emitting state has associated with it a probability density function $b_j(x(t))$, where the probability of emitting a given feature vector if in state *j* time *t* is:

$$b_j\big(x(t)\big) = p(x(t)|q_t = j))) \tag{3.24}$$

An initial state distribution is also required. In common with the standard HTK conventions, the state sequence is constrained to begin and end in the first and last states, with the models begin concatenated together by the non-emitting states [Stuttle, 2003].

In the end, three problems must be solved before HMMs can be applied to real-words applications [Rabiner, 1993; Huang, 2001]:

- Evaluation: given an observation sequence $X = \{x_1, x_2, \dots, x_T\}$ and a model λ, how the probability of the observation sequence given the model, $P(O|\lambda)$, is efficiently computed?
- Decoding: given an observation sequence $X = \{x_1, x_2, \dots, x_T\}$ and a model λ, how to choose the corresponding state sequence $S = \{s_1, s_2, \dots, s_T\}$ that is optimal in some sense?
- Learning: given a model λ, how to estimate the model parameters to maximize $P(O|\lambda)$?

In the section to follow we will shortly cover solutions to each of the above problem.

3.4.3 Evaluation

Regarding the evaluation problem, the simplest way to compute the probability $P(O|\lambda)$ of the observation sequence, $X = \{x_1, x_2, \dots, x_T\}$, given the λ model, is through summing the probabilities of all possible sequences *S* to *T*, as follows:

$$P(X|\lambda) = \sum_S P(X, S|\lambda) = \sum_S P(X|S, \lambda)P(S|\lambda) \tag{3.25}$$

where $P(X|S, \lambda)$ is the probability of observing the sequence *X* given a particular state sequence *S* and $P(S|\lambda)$ is the probability of occurring such a state sequence *S*. Given the output independence assumption and applying the 1st order Markov, $P(S|\lambda)$ can be rewritten as follows:

$$P(X|\lambda) = \sum_S \pi_{S_1} b_{S_1}(x_1) a_{S_1 S_2} b_{S_2}(x_2) \dots a_{S_{T-1} S_T} b_{S_T}(x_T) \tag{3.26}$$

As equation 3.26 is computationally infeasible, because it requires $(2T-1)N^T$ multiplications and N^T-1 additions, more efficient methods exist, such as "Forward algorithm" or "Viterbi" search, to compute $P(X|S, \lambda)$.

3.4.4 Decoding

The decoding problem for HMMs involves finding the state sequence that is most likely to have generated an observation sequence. This may be solved using a dynamic programming algorithm, often referred to as Viterbi decoding referenced above, which has a very similar structure to the Forward algorithm, with the exception that the summation at each time step is replaced by a max operation, since just the most probable state sequence is required. The decoding problem is, in fact, the speech recognition problem. The Viterbi algorithm is used to find the most likely sequence of words and estimate the probability that this sequence has generated the acoustic observations. These algorithms are described in the sections to follow.

3.4.4.1 Forward–backward algorithm

The forward algorithm is a type of dynamic programming algorithm that stores intermediate values as it builds up the probability of the observation sequence. The algorithm evaluates state by state the probability of being at that state given the partial observation sequence, that is:

$$\alpha_t(i) = P(x_1, x_2, \dots, x_t, S_t = i|\lambda \tag{3.27}$$

where $\alpha_t(i)$ is the probability of the partial observation sequence in state *i* at time *t*, given the model λ.

The variable $\alpha_t(i)$ can be solved inductively, starting from the initialization step:

$$\alpha_1(i) = \pi_i b_i(x_1) \;\; 1 \le i \le N \tag{3.28}$$

Induction phase:

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i)\, x_{ij}\right], 1 \le t \le T-1, 1 \le j \le N \tag{3.29}$$

Solution:

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i) \tag{3.30}$$

This algorithm reduces complexity to $X(N^2T)$, much better than the exponential complexity, according to [Adami, 2010]. The temporal constraints assumed in HMMs for speech terminates the search of the forward backward algorithm when $P(X|\lambda) = \alpha_T(S_F)$, and this further limits the number of solutions and simplifies the search.

3.4.4.2 Viterbi Search

The Viterbi algorithm is used to find the most likely sequence of words and estimate the probability that this sequence has generated the acoustic observations. It estimates the probability that the HMM is in state j after seeing the first t observations, like in the forward algorithm, but only over the most likely state sequence $s_1, s_2, \dots, s_{t-1}$, given the model λ, that is [Adami, 2010]:

$$\delta_1(i) = \max_{s_1, s_2 \dots, s_{t-1}} P(s_1, s_2, \dots, s_{t-1}, s_t = i, x_1, x_2, \dots, x_t|\lambda) \tag{3.31}$$

where $\delta_1(i)$ is the probability of the most likely state sequence in state *i* at time *t* after seeing *t* observations. An array $\psi_t(t)$ is used to keep track of the previous state with highest probability so the state sequence can be retrieved at the end of the algorithm. The Viterbi algorithm can also be solved iteratively, starting from the initialization step:

$$\delta_1(i) = \pi_i b_i(x_1);\; \psi_t(t) = 0 \quad 1 \le i \le N. \tag{3.32}$$

Recursion phase:

$$\delta_t(j) = \max_{1 \le i \le N} [\, \delta_{t-1}(i)\, a_{ij}] b_j(x_t) \tag{3.33}$$

$$\psi_t(t) = \operatorname*{argmax}_{1 \le i \le N} [\, \delta_{t-1}(i) a_{ij}], \;\; 2 \le t \le T, 1 \le j \le N \tag{3.34}$$

Termination phase:

$$p* = \max_{1 \le i \le N} [\, \delta_t(i)] \tag{3.35}$$

$$s_t* = \operatorname*{argmax}_{1 \le i \le N} [\, \delta_t(i)] \tag{3.36}$$

Path backtracking:

$$s_{t1} *= \psi_t(s_{t+1} *), \quad t = T-1, T-2, \ldots, 1. \tag{3.37}$$

3.4.5 Learning

The estimation of the model parameters λ is the most difficult of the three problems, because there is no known analytical method to maximize the probability of the observation sequence in a closed form. However, the parameters can be estimated by maximizing $P(X|\lambda)$ locally using another iterative algorithm, like the Baum-Welch algorithm (also known as the forward-backward algorithm). The forward-backward algorithm is essentially the maximum likelihood estimation algorithm for a hidden Markov model, maximizing the likelihood of observing the training data through a given HMM structure.

In the first part of the algorithm, a probabilistic state-time alignment is computed, assigning a state occupation probability to each state at each time, given the observed data. Then parameters are estimated by an average weighted by the state occupation probabilities. The maximization algorithm has been shown to converge in a local maximum of the likelihood function. The state occupation probabilities can be computed recursively, as follows:

$$\upsilon_t(q_j) = \frac{1}{\alpha_T(q_E)} \alpha_t(q_j)\beta_t(q_j) \tag{3.38}$$

where $\alpha_t(q_j)$ is the forward probability for state q_j at time t, $\beta_t(q_j) = p(x_{t+1}, x_{t+2}, x_T|q_t = q_j$ is called the backward probability and $\alpha_T(q_E)$ is a normalization factor, defined as the forward probability for the end state q_E at the end of the observation sequence, time T. The backward probabilities are called so because they may be computed by a recursion that goes backwards in time. But the output PDFs are the most important part of this model, and restricting them to single Gaussians results in a significant limitation on modeling capability. In practice, Gaussian mixture model (GMMs) are used as output pdfs. A GMM is a weighted sum of Gaussians, defined as:

$$b_i(x) = p(x|q_i) = \sum_{k=1}^{k} c_{ik} N(x; \mu_{ik}, \sum_{ik}) \tag{3.39}$$

where we have a mixture of K Gaussian components, with mixture weights c_{ik}, for every HMM state. Training a GMM is analogous to HMM training: for HMMs the state is a hidden variable, for GMMs the mixture component is a hidden variable. Again the EM algorithm may be employed, with the E-step estimating the component occupation probabilities, and the M-step updating the means and covariance's using a weighted average. Next section will go a little more in detail regarding the Gaussian Mixture Model representations.

3.4.6 Gaussian Mixture Models (GMM)

A Gaussian Mixture Model (GMM) is a parametric probability density function represented as a weighted sum of Gaussian component densities [Reynolds, 2009]. As shown above, they are commonly used as a parametric model of the probability distribution of continuous measurements or features seen in the training phase in a speaker recognition system, such as spectral features. GMM parameters are estimated from training data using the iterative Expectation-Maximization (EM) algorithm. Figure 3.7, adapted from [Stuttle, 2003], offers a quick overview of the extraction of GMM parameters from the speech signal.

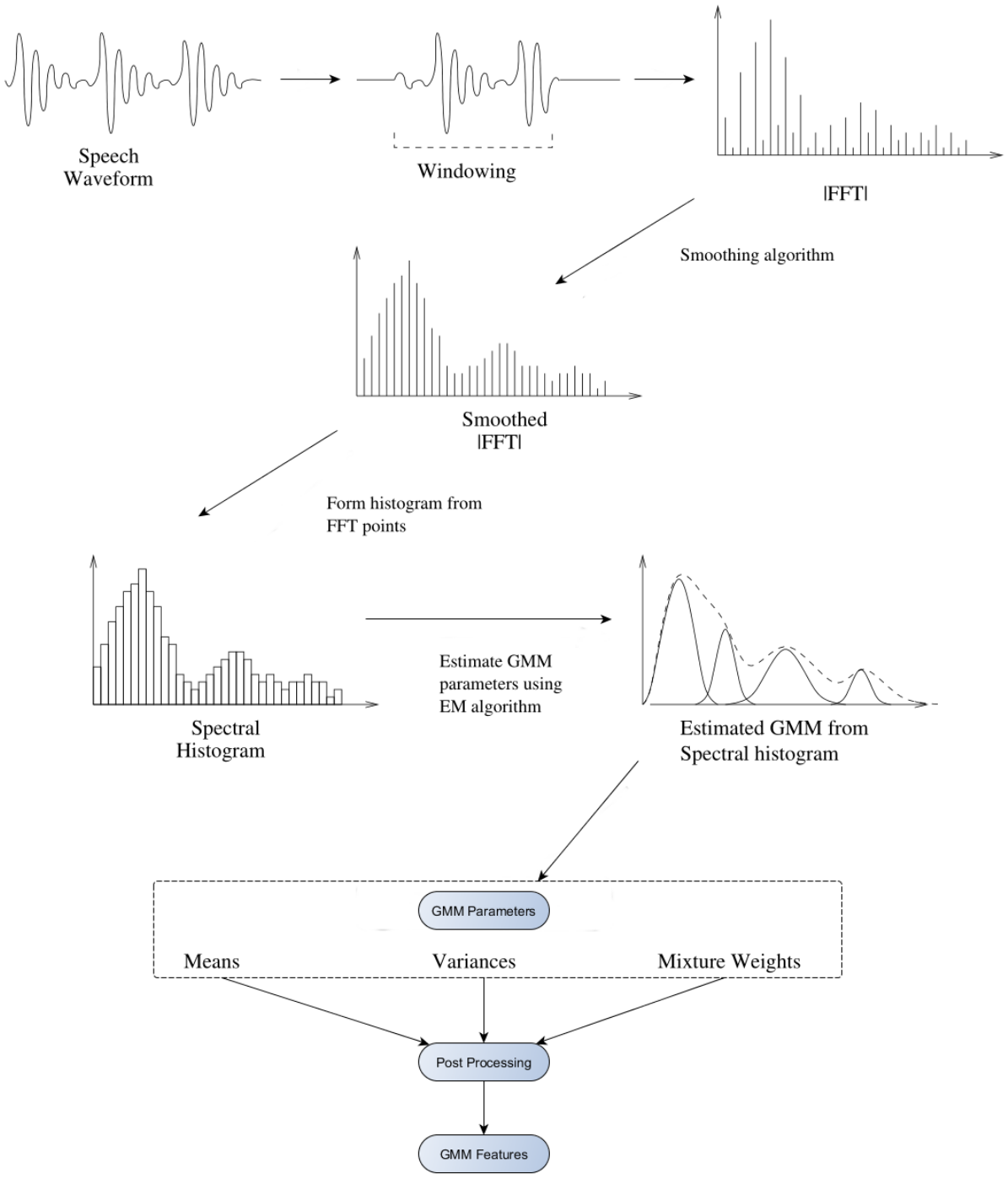

Figure 3.7 Overview of the extraction of GMM parameters from the speech signal, as shown in [Stuttle, 2003]

A Gaussian mixture model is a weighted sum of *M* component Gaussian densities as given by the equation:

$$p(x|\lambda) = \sum_{i=1}^{M} w_i g(x|\mu_i, \sum_i) \tag{3.40}$$

where *x* is a *D*-dimensional continuous-valued data vector, like measurement or features, w_i, $i = 1, ..., M$, are the mixture weights, and $g(x|\mu_i, \sum_i)$, $i = 1, ..., M$ are the component Gaussian densities. Each component density is a *D*-variate Gaussian function of the form:

$$g(x|\mu_i, \sum_i) = \frac{1}{(2\pi)^{D/2}|\sum|^{1/2}} e^{-1/2(x-\mu_i)'\sum_i^{-1}(x-\mu_i)} \quad (3.41)$$

where μ_i is the mean vector and $\sum_i$ is the covariance matrix, with the mixture weights satisfying the following constraint: $\sum_{i=1}^{M-1} w_i = 1$.

In a compact formulation, the complete Gaussian mixture model is parameterized by the mean vectors, covariance matrices and mixture weights from all component densities as follows [Reynolds, 2009]:

$$\lambda = \{w_i, \mu_i, \sum_i\}, i = 1, \dots, M \quad (3.42)$$

The choice of model configuration, like number of components, full or diagonal covariance matrices, and parameter tying, is often determined by the amount of data available for estimating the GMM parameters and how the GMM is used in a particular ASR application. In [Reynolds, 2009] the author mathematically describes the steps necessary to estimate the parameters of the GMM, λ, given training vectors and a GMM configuration, using EM algorithm.

At the end of this chapter, a set of experiments and evaluations are presented for building a limited vocabulary ASR system, where the choice of the number of GMMs is presented. GMMs are used in speaker recognition systems due to their capability of representing a large class of sample distributions and one of the powerful attributes of the GMM is its ability to form smooth approximations to arbitrarily shaped densities.

3.4.7 Noise robustness

In Chapter 1, section 1.2, various sources of speech variability were introduced, that make the general task of ASR a very challenging one. Background noise present in the recognition environment plays a key role in distorting the speech signal, and several noise compensation algorithms are available to "clean" the audio signal, especially if we work with under-resourced languages, were the number of training resources are limited.

Noise compensation algorithms usually start from the premise that a clean, time domain voice signal *x(t)*, is altered by an additive noise *n(t)* and a convoluted finite impulse response *h(t)*, from the recognition channel, resulting thus *y(t)*, the noisy signal:

$$y_t = x_t \otimes h_t + n_t \quad (3.43)$$

In practice, an ASR uses extracted features form the voice signal, as detailed in previous chapter. If MFCC parameters are used, the above equation for *y(t)* becomes:

$$y_t^S = x_t^S + h + Clog\left(1 + e^{C^{-1}(n_t^S - x_t^S - h)}\right) = x_t^S + f(x_t^S, n_t^S, h) \quad (3.44)$$

where *C* is the DCT matrix. For a given additive noise, the observation vector y_t^S is a nonlinear function of the clean signal, x_t^S. In other words, with the decrease of Signal-to-noise-ratio (SNR), the second parameter in the above equation becomes dominant, and the vectors means tend to noises means, with lower variances. In a very low SNR conditions, noise dominates and there is very limited information in the signal, and ASR recognition rate decreases [Buzo, 2011].

Another approach is use speech features which are inherently noise robust. For instance, cepstral mean normalization will remove some of the effects of convolutional channel noise. Convolutional noise can also be removed by the JRASTA and RASTA-PLP approaches [Hermansky, 1994], or PNCC features, presented in Chapter 2 of this thesis. Inherently noise robust approaches are desirable as they do not need to be adapted to a particular type or source of noise. However, most noise robust features can be further improved by other noise robustness techniques, such as "speech compensation/enhancement" [Stuttle, 2003].

Models used for acoustic recognition can be also used for compensating the "unclean" speech, by adapting the clean model set or algorithm to the corrupted speech. Techniques using this approach include linear regression adaptation approaches [Woodland, 1999], speech and noise decomposition [Varga, 1990] and parallel model combination [Gales, 1995]. Parallel model combination (PMC) attempts to combine the "clean" speech HMM models with a model of the noise distribution [Gales, 1993]. There are no closed-form solutions for the problem of combining the models and noise suppression, so this is still an open issue in the speech community.

3.4.8 ASR evaluation metrics

If the speech recognition problem is posed as the transformation of an acoustic signal to a single stream of words, then there is widespread agreement on word error rate (WER) as the appropriate evaluation measure. The sequence of words output by the speech recognizer is aligned to the reference transcription using dynamic programming. The accuracy of the speech recognizer may then be estimated as the string edit distance between the output and reference strings. If there are *N* words in the reference transcript, and alignment with the speech recognition output results in *S - substitutions*, *D - deletions*, and *I - insertions*, the word error rate is defined as:

$$WER[\%] = \frac{Insertions + Substitions + Deletions}{Words\ in\ reference\ transcriptiox} \times 100 \tag{3.45}$$

Sometimes the word error rate can be greater than 100% because the above equation also includes the number of insertions. In some applications, a second evaluation metric, the sentence error rate (SER), might also be important. The sentence error rate is based on the word error rate and can be computed as follows:

$$SER[\%] = \frac{Sentences\ with\ at\ least\ one\ word\ error}{Sentences\ in\ the\ reference\ transcription} \times 100 \tag{3.46}$$

3.4.9 Limitations and practical issues

Hidden Markov Models (HMMs) provide a simple and effective framework for modelling time-varying spectral vector sequences, as those present in speech recognition tasks. This is one of the reasons why HMM methods have become so popular nowadays in ASR systems, along with the following advantages:

- The ease and availability of training algorithms, for estimating the parameters of the models from finite training sets of speech data.
- The availability or large written text corpora available on the internet, along with the increasing computer power, allow for better language modelling techniques.
- The flexibility of the resulting recognition system in which one can easily change the size, type, or architecture of the models to suit particular words, sounds or recognition task.
- Ease of implementation of the overall recognition system.

As a consequence, almost all present day large vocabulary continuous speech recognition (LVCSR) systems are based on HMMs. The use of HMM models for speech recognition has become predominant in the last several years, as evidenced by the number of published papers and talks at major speech conferences.

The first and one of the foremost problems with training a statistical system, regardless of whether it is an HMM or a GMM, was the shortage of training data. Seen as a limitation in the past, with the advent of the internet, this remains an issue mostly for under-resourced languages, where more acoustical data is needed. There is also the issue of "sufficient statistic", as to when more data decreases WER and increases accuracy, as it is quite impossible to have accounted for

all the different possible observations which may be encountered in a language. Also, HMM approach can easily accommodate different levels of constraints, like phonological or syntactical. However, algorithms must explicitly or implicitly make numerous assumptions about speech, although some of them are obviously unrealistic. For example, it is often necessary to assume that the features extracted within a phonetic segment are uncorrelated with another. Also, HMM training algorithms presented in precious sections are based on likelihood maximization, which assumes correctness of the models, and it cannot always be true [Morgan, 1995].

However, spoken language recognition is still not a solved problem in a fundamental sense. While existing technology may be sufficient for continuous large vocabulary speech recognition, human beings expect speech systems to behave much as people would, to "understand" the uttered content (unfamiliar accents, background noise and reflective room acoustic, improper grammar, unfamiliar words, etc.). In any of the above mentioned context, we can generally understand what is being said and human recognition performance under many realistic conditions, is still much better than that of any machine.

Research in this area continues, as there is much room for improvement in speech relates domains, such as spoken language understanding, audio database retrieval and indexing, etc. For this reason, in this thesis we investigate the use of alternate approaches, such as neural-network based methods for acoustic modelling, and use them to learn as much as we can from fewer data, to be used in unsupervised discovery techniques later in this thesis.

3.5 A NEURAL NETWORK APPROACH TO ACOUSTIC MODELING

One of the most popular alternative approaches to acoustic modelling used in ASR is the combination of an Artificial Neural Net (ANN) with a HMM to form a hybrid HMM-ANN system, which in recent years has become a very powerful tool in the field of pattern recognition. Pattern recognition refers to means of analyzing data originating from analog sources (such as speech signal) and classifying them into categories, for further processing. Besides fingerprint identification, pattern matching can be successfully applied in other applications, such as handwritten recognition, identification of DNA sequences to automatic speech recognition [Domokos, 2009].

Artificial neural networks cannot be independently used to achieve continuous speech recognition, but together with hidden Markov models or other architectures, they constitute systems that can offer considerable results in probability estimation and feature induction, and have been extended for modeling both sequential and structured data [Deller, 2000]. The modular approach of an ASR system enables the integration of different technologies, such as hybrid systems, TRAP systems, STC Nets to estimate, for example, phoneme probabilities, probabilities that can be used in HMM models, for example.

Their most successful applications in spoken language recognition have been language modeling and parsing. Neural nets have also been inspirational for much current research in machine learning methods, and can be reinterpreted in terms of approximations to latent variable models. ANNs have the advantages of being robust in training and testing, of being fast in testing, and of requiring little prior knowledge of the domain. ANNs are also interesting because they discover compact feature-based representations specific to the task they are trained on [Clark, 2010].

We offer a quick overview of the current proposed neural approaches studied by the thesis author, and in Chapter 5 some experiments with unsupervised pattern-recognition approaches are proposed, in the field of spoken language indexing, namely spoken term detection and discovery.

3.5.1 Artificial Neural Networks (ANN)

The term artificial neural network (ANN), or often just a neural network (NN), refers to a variety of computational models which share certain properties inspired by the networks of neurons found in the brain. They consist of a distributed network of simple processing units, and usually they are designed to be trained from data. Most research on ANNs has lost any pretense of being neurologically motivated, and today is mostly of engineering interest. It is mostly its usefulness for engineering solutions which has interested research in spoken language recognition, hence in ASR systems [Henderson, 2010].

Another property typically associated with ANNs is the unsupervised induction of representations during learning. Some of the processing units in the ANN have no predefined meaning, they acquire their meaning during training. In some cases, these units are the output of the ANN, as for example for the unsupervised clustering of self-organizing maps [Kohonen, 1984]. In other cases, these units form an intermediate representation in between the input and the output of the ANN. Such units are called "hidden units". By far the most popular form of ANN has been the multilayered perceptron (MLP), and its recurrent variants discussed in the section to follow. MLPs are used for function approximation, categorization, and sequence modeling [Henderson, 2010].

We start first by defining the artificial neuron, or a perceptron, that forms the basis of every neuron modelling technique. Figure 3.8 highlights the model of an artificial neuron, where the underlying elements of such a model can be observed: model inputs $x_1, x_2, \dots, x_N$, the output of the model *y*, the activation or threshold function *f*, and weights $w_1, w_2, \dots, w_N$.

Mathematically, the output y of a perceptron can be defined as:

$$y_k = f\left(\sum_{j=0}^{m} w_{kj}x_j - B\right) \tag{3.47}$$

Where *B* is the threshold bias. Usually each input (signal) is weighted (multiplied by an adjustable weight value w_i), and their sum is passed through a non-linear function known as an activation function, or transfer function. The transfer functions usually have a sigmoid shape, but they may also take the form of other non-linear functions, piecewise linear functions or step functions. They are also often monotonically increasing, continuous, differentiable and bounded.

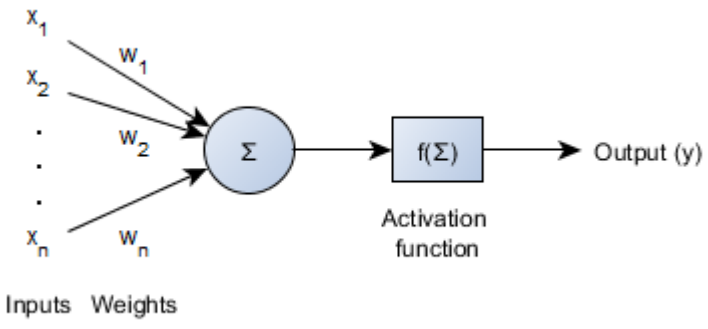

Figure 3.8 Artificial neuron model (perceptron) processing unit

The most popular neural non-linear transfer functions are:

- Linear transfer function
- Sigmoid transfer function
- Hyperbolic tangens transfer function

- Gaussian radial transfer function

Perceptron learning cannot adapt to the data that are not linearly separable. In order to separate more complex data there are used non-linear activation functions, more layers of an artificial neural network (ANN) and most often the popular back-propagation algorithm to train such kinds of networks, hence more complex interconnected networks are used, like Multi-Layer Perceptrons introduced in the following section.

3.5.2 Multi-Layer Perceptrons

A Multi-Layer Perceptron is illustrated in Figure 3.9. The nodes of the graph are the processing units, and the edges are weighted links. The units are organized into input units, output units, and hidden units. Given a vector of input values *x* placed on the input units, the MLP will compute a vector of output values *y* on its output units. In the process it will iteratively compute values for each layer of hidden units. MLPs are "feed-forward" networks, which means that there can be no loops in the directed graph of links, so this iterative computation can be done in a single pass from the inputs to the outputs.

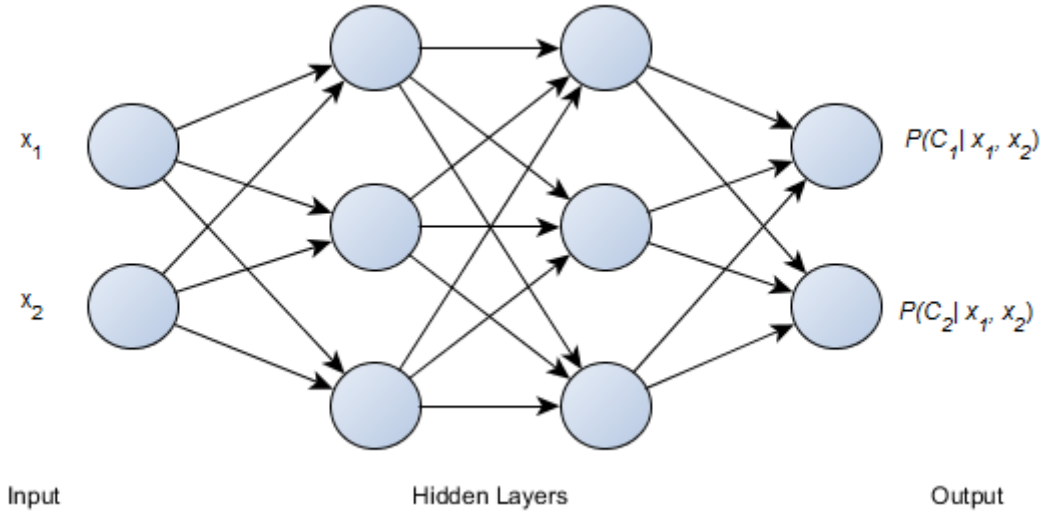

Figure 3.9 A multi-layered perceptron

A unit *j* computes its output value, called its activation, as a function of the weights w_{ji} on links from units *i* to unit *j* and the activations z_i of these units *i*. For the hidden units, the output of each unit z_j is often a normalized log-linear function of its weighted inputs, called a sigmoid function:

$$z_j = \frac{1}{1 + \exp(-(w_{j0} + \sum_i z_i w_{ji})} \tag{3.48}$$

For the output units y_j, we can approximate a continuous function by simply using the weighted sum $w_{j0} + \sum_i z_i w_{ji}$. Multi-class classification can be done using one output unit per class, and choosing the maximum weighted sum. But most often in NLP we are interested in a probability distribution over classes. There are several ways of interpreting hidden layers, but the most intuitive for our purposes is interpreting them as computing a new set of continuous-valued features from their input features. These vectors of features are often called distributed representations [Henderson, 2010].

Statistical approaches to NLP often require models which produce proper probability estimates. MLPs can be trained to produce probability estimates by choosing appropriate functions for the output and the error. If we use a normalized exponential output function and cross-entropy

error function, then after training, the MLP will output an estimate of the probability distribution over output categories [Henderson, 2010].

Training tries to find the weights which minimize the cross-entropy error function given the normalized exponential output function. Given enough data, the global minimum will be at weights which give us the true probability distribution $P(y_k \mid x_k)$ over output categories given the input x^k, back-propagation algorithm being one popular approach to train such kind of networks.

3.5.3 Learning in NN-MLP

Back-propagation starts with a random set of weights, and at each step changes the weights a little bit in the direction which will maximally reduce the error, discussed below. Calculating the direction for this update step requires computing the first derivative of the error with respect to every weight for a given data point, and then summing over data points in the training set. Computing this derivative can be easily done in MLPs by iteratively computing the derivative of the error for each layer, starting from the outputs and proceeding backward towards the inputs. As with computing outputs, computing these derivatives can be done in a single pass through the network. This process of propagating the error derivatives back through the network gives rise to the name backpropagation.

The backpropagation learning algorithm is divided into two phases [Horzyk, 2013]:

- Forward propagation of pattern's inputs through all neurons and all layers of the neural network till the output values of the output neurons of the network are computed.
- Update of the weights during back propagation of the computed errors on neurons outputs.

The weights are updated using the calculated output delta δ (error) and input activation. These two values are multiplied in order to compute the gradient for the weight correction. Next, substract a learning ratio of the gradient form the weight. The greater the learning ratio is, the faster the weights adapt, but the lower learning ratio is more accurate and limits fluctuation close to an error function minimum. The learning ratio can also change during learning process. It is usually downgraded during a training process.

The following paragraphs will illustrate the principles of training a multi-layer neural network using backpropagation, as explained graphically in [Bernacki, 2005], starting from a three layer neural network with two inputs and one output. Each neuron is composed of two units. First unit adds products of weights coefficients and input signals. The second unit uses a nonlinear function, called neuron activation function. Signal *e* is the summed output signal, and *y = f(e)* is output signal of the nonlinear element and also the output signal of neuron unit. This process was described in the previous section, and Figure 3.10 illustrates the neural network and its unit used in this example.

To teach the neural network we need a training data set. The training data set consists of input signals (x_1 and x_2) assigned with corresponding target (desired output) *z*. The network training is an iterative process. In each iteration weights coefficients of nodes are modified using new data from training data set.

Modification are calculated using the algorithm described below:

- Each teaching step starts with forcing both input signals from training set. After this stage we can determine output signals values for each neuron in each network layer.
- Figure 3.11 illustrates how signal is propagating through the network, symbols $w_{(xm)n}$ represent weights of connections between network input x_m and neuron *n* in input layer. Symbols y_n represents output signal of neuron *n*.

- Signals are propagated through the hidden layer, then through the output layer. Symbols w_{mn} represent weights of connections between output of neuron m and input of neuron n in the next layer.

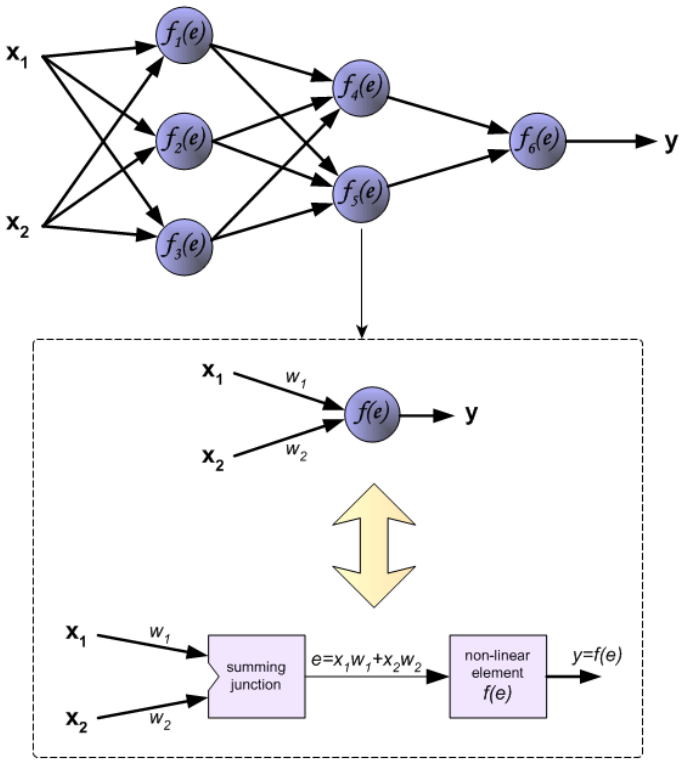

Figure 3.10 Three layer MLP and its neural unit, adapted [Bernacki, 2005]

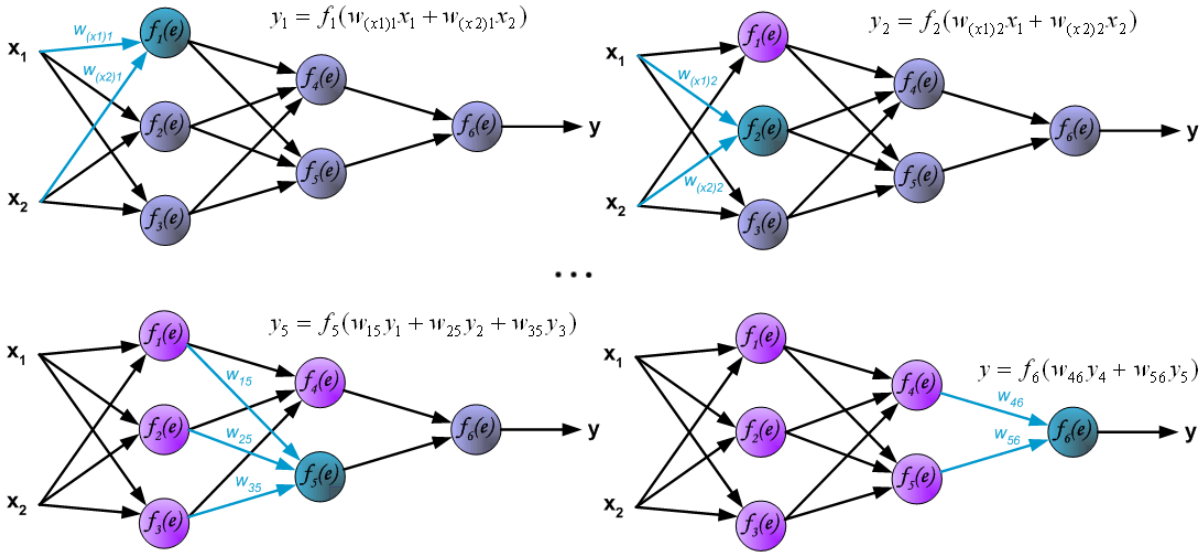

Figure 3.11 Iterative network training, adapted [Bernacki, 2005]

Next step consists in calculating signal error $\delta = z - y$ of the output layer neuron, by comparing y, the output signal of the net with the desired output target, z, found in the training data set. It is impossible to compute error signal for internal neurons directly, because output

values of these neurons are unknown, so the idea is to propagate the error signal δ, computed in each single teaching step, back to all neurons (backtracking), which output signals were input for current neuron (Figure 3.12). The weights coefficients w_{mn} used to propagate the errors back are equal to the ones used during the computation of the output value. Only the direction of data flow is changed now.

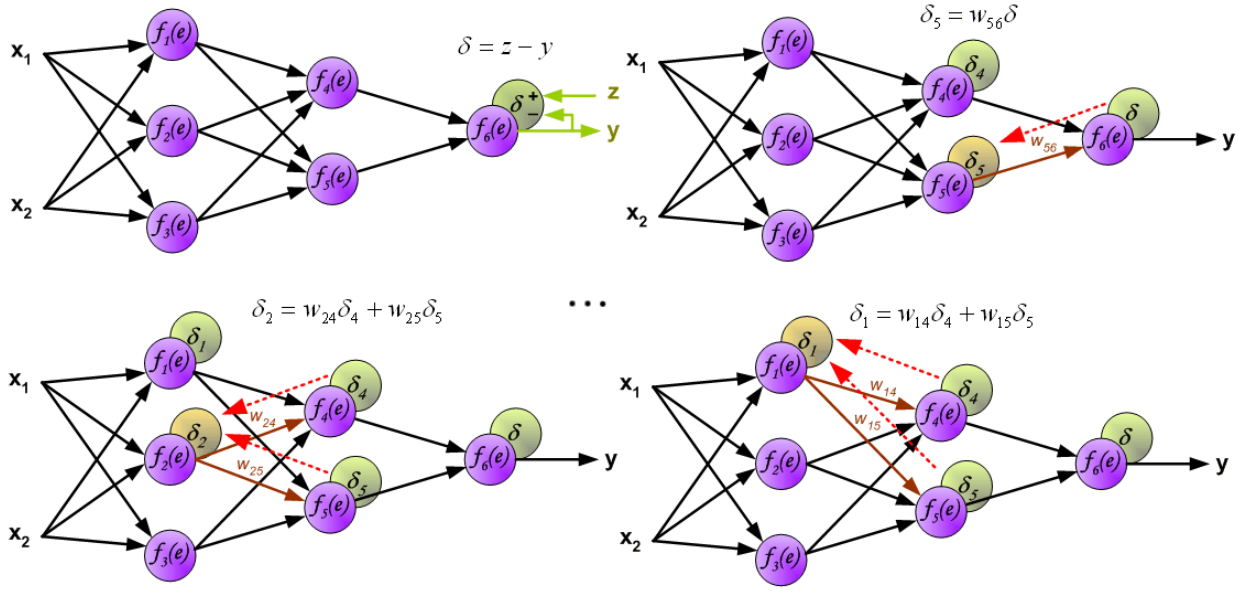

Figure 3.12 Backtracking used to calculated the error signal δ

When the δ error is computed for each neuron, the weights coefficients for each neuron input node may be modified. In Figure 3.13, $\frac{df(e)}{de}$ represents the derivative of neuron activation function (which weights are modified), where η is a coefficient that affects the network teaching speed.

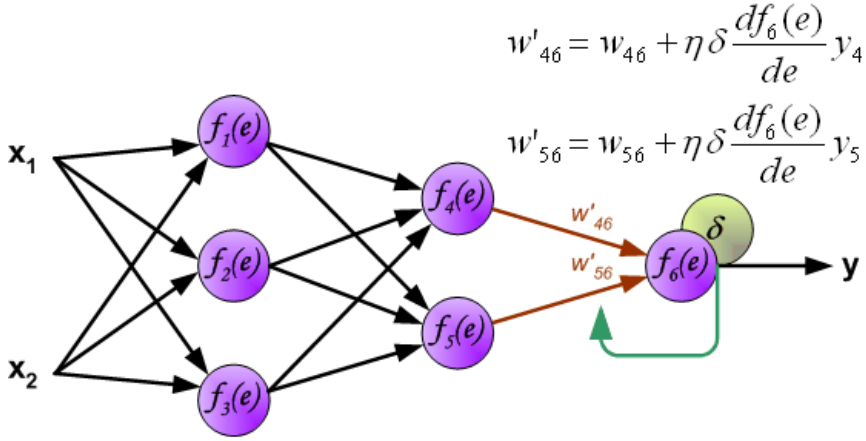

Figure 3.13 Final weights calculation

We stop training when performance on this development set goes down, and we use the best performing set of weights as our final model. By stopping training before performance on the training set reaches its maximum, we prevent weights from growing too large.

Multi-layered perceptron's presented in this section compute a function from a fixed-length vector of input values to a fixed-length vector of output values. This is often insufficient for NLP tasks, because inputs and outputs are sequences which can be arbitrarily long, such as the words in a sentence. For such problems we can use recurrent MLPs, because their graph of links includes links which loop back towards the input of the node.

3.5.4 TRAPs systems

The multilayer perceptron is one possibility for acoustic matching. But [Hermansky, 1999] and [Chen, 2005] found that more complicated neural network structures can be beneficial for speech recognition. Such an approach are the Temporal Patterns (TRAPs) systems, benchmarked in [Hermansky, 1999] and revisited and improved in [Schwarz, 2009], which is in fact a HMM/NN hybrid.

A TRAPs system uses a separate neural network for each critical band. These "front-end nets" are trained to classify input patterns to phoneme posteriors. Another neural network (a "merger" or "back-end net") is trained to merge the posteriors from all bands. The outputs are again phoneme posteriors. Separate input patterns for all the frontend nets are simpler than the whole input pattern: they are more easily learned by networks and the input patterns can be longer than if the whole pattern was processed by one net [Schwarz, 2006]. Such a system is shown in Figure 3.14.

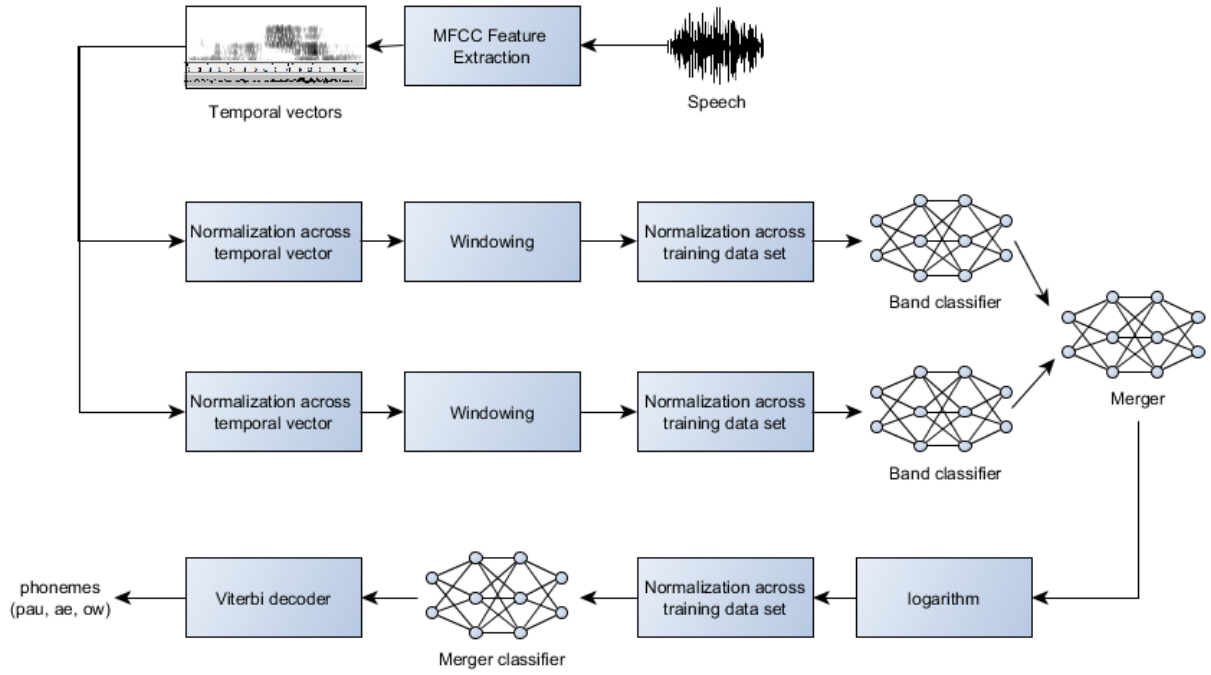

Figure 3.14 Trap system architecture, adapted [Schwarz, 2006]

[Schwarz, 2006] further details the TRAP Architecture: "Speech is segmented into frames 25 ms long and for each frame, Mel-bank energies are calculated. Temporal evolution of energy for each band is taken (101 values = 1 second), normalized to zero mean and unit variance across the temporal vector, windowed by Hamming window and then normalized to zero means and unit variances across all training vectors. This is beneficial for the ANN as it is ensured that all inputs have the same dynamics. For testing, the later normalization coefficients are not calculated but taken from the training set. Such prepared temporal vectors are presented to band neural networks. These neural networks are trained to map temporal vectors to phonemes. A vector of phoneme posterior probabilities is obtained at the output of each band neural network. The posterior probabilities from all bands are concatenated together, the logarithm is taken and this vector is presented to another neural network (merger). The merger is trained to map the vectors to phonemes again. The output is a vector of phoneme posterior probabilities. Such vectors are then sent to the Viterbi decoder to generate phoneme strings". Phoneme strings can then be further processed in a HMM chain, typical for an ASR system.

3.5.5 Systems with Split Temporal Context (STC LC-RC system)

The disadvantage of the system described above is its quite huge complexity and high computational demand, which could be a problem in real-time applications. Therefore, [Schwarz, 2006] introduced a simplified version of the phoneme recognition system, at least to reduce the computational cost, called a "Split temporal context system" (LC-RC system) [Schwarz, 2004].

The split temporal context system approach is based on the theoretical study that significant information about phoneme is spread over few hundreds milliseconds and that an STC system can process two parts of the phoneme independently. The trajectory representing a phoneme feature can then be decorrelated by splitting them into two parts, to limit the size of the model, in particular the number of weights in the neural-net (NN). The system uses two blocks of features, for left and right contexts (the blocks have one frame overlap). Before splitting, the speech signal is filtered by applying the Hamming window on the whole block, so that the original central frame is emphasized. Dimensions of vectors are then reduced by DCT and results are sent to two neural networks. The posteriors from both contexts are, in the final stage, merged, after the front-end neural networks are able to generate a three-state per phoneme posterior model. The whole process is detailed in Figure 3.15.

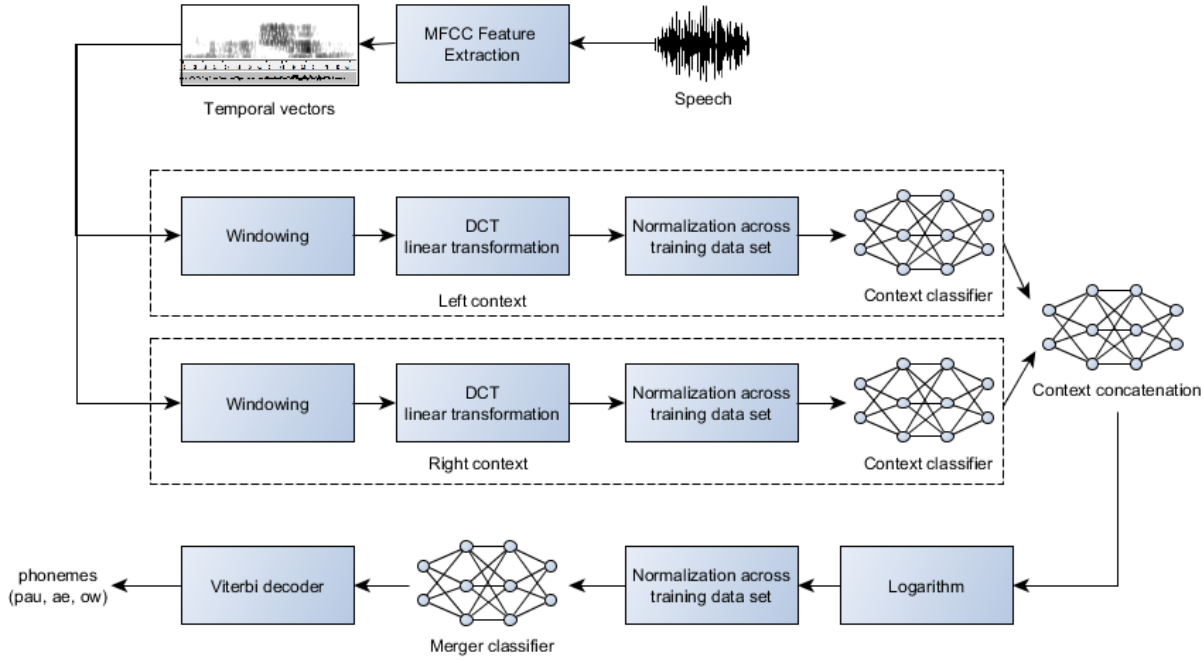

Figure 3.15 Split Temporal Context system proposed by [Schwarz, 2006]

The above technique is inspired by the function of band neural networks in the TRAP system described in previous section, and is able to classify long trajectories in the feature space by splitting the trajectories into more parts. "The Mel-bank energies were extracted and the 310 ms long temporal vectors (31 values) of evolution of critical bank energies were taken. Each temporal vector was split into two parts – left part (values 0 - 16) and right part (values 16 - 31). Both parts were windowed by corresponding half of Hamming window and projected to the DCT bases. 11 DCT coefficients were kept for each part. Such preprocessed vectors were concatenated together for each part of context separately and sent to two neural networks – these are trained to produce phoneme posteriors, similarly as in the TRAP system. Output posterior vectors are concatenated, transformed by logarithm and sent to another (merging) neural network trained again to deliver phoneme posteriors. All neural networks were trained using classical back-propagation algorithm with cross-entropy error function. Finally, the phoneme posteriors are decoded by a Viterbi decoder and strings of phonemes are produced" [Schwarz, 2006], that can be further processed.

3.6 SOFTWARE TOOLKITS

This section describes some of the tools used by the thesis author to build an ASR system, perform features or phoneme extraction, and then further build a spoken term detection / discovery system.

The CMU Sphinx Toolkit [Lamere, 2003] is used to implement the ASR architecture described at the end of this Chapter, in the section to follow. CMU Sphinx, also called Sphinx in short, is the general term to describe a group of speech recognition systems developed at Carnegie Mellon University. These include a series of speech recognizers (Sphinx 2 - 4) and an acoustic model trainer (SphinxTrain). The code is available open source for download and use [Sphinx, 2015].

Another popular speech development toolkit is Hidden Markov Model Toolkit (HTK), also open source. Some studies compare the "speech recognition performance of the two toolkits [Kačur, 2006; Ma, 2009]. They generally conclude that similar systems developed with the two toolkits have a similar performance, but the acoustic modelling performed by Sphinx" is slightly better. HTK Toolkit is also used for MFCC feature extraction, used in the unsupervised motif discovery experiments done by the thesis author, and is available at [HTK, 2015].

Phoneme recognizers used in Chapter 5, for unsupervised spoken term discovery and detection are based on [BUT, 2015]. This phoneme recognizer was developed at Brno University of Technology, Faculty of Information Technology and was successfully applied to tasks including language identification, indexing and search of audio records, and keyword spotting. Outputs from this phoneme recognizer can be used as a baseline for subsequent processing, as for example input to our DTW block used for string search in Chapter 5.

For Unsupervised Spoken Term Discovery experiments we used [Modis, 2015], a free speech and audio motif discovery software developed at IRISA Rennes. Motif discovery is the task of discovering and collecting occurrences of repeating spoken patterns in the absence of prior acoustic and linguistic knowledge, or training material.

Language models, text corpuses and miscellaneous toolkits for text processing were provided and used from SpeeD Research Lab, Romania [SpeeD, 2015].

3.7 DEVELOPING A SPEAKER DEPENDENT / SPEAKER INDEPENDENT CONNECTED DIGITS RECOGNITION SYSTEM

A connected-digits speech recognition system is a limited-vocabulary recognizer. This means the system will only recognize and transcribe the decimal system, in Romanian: *zero, unu, doi, …, nouă*. Speaker characteristics were taken into account in this paper. Theoretically, a speaker-dependent system should be better at decoding speech uttered by the specific user for which it was trained. However, this demands a new system to be constructed and trained individually for each speaker, a time consuming and non-scalable task. To address this issue, a second speaker-independent system was trained with multiple audio files from SpeeD (Speech & Dialogue Research Laboratory) "roDigits" speech database. Results were compared in terms of word-error-rate (WER), sentence-error-rate (SER) and are presented in Section 3.7.3. The effects of increasing/decreasing the number of Gaussians per senone and the number of tied-states (senones) for each system were also shown in Section 3.7.3.

3.7.1 Methodology

The CMU Sphinx Toolkit described in the Software toolkits section is used to implement the ASR architecture described in Figure 3.2.

3.7.1.1 Speech Recording

A speech database comprising recorded audio files is required to build an acoustic model for the speech recognition system. An online speech recorder application, developed by SpeeD research group, is used to record the audio files. Several speakers recorded predetermined audio messages, containing multiple groups of random digits. For the initial training of the speaker-depended system, 100 audio clips were recorded. Each clip contains 12 uttered digits. Integrated laptop microphones were avoided and recording volume was set to high. An initial recording calibration was required (to detect background noise). Figure 3.16 presents the GUI for the recording application.

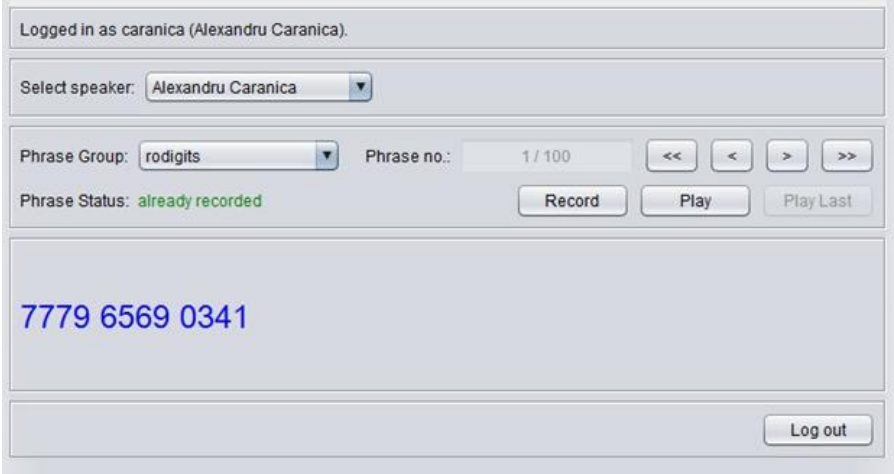

Figure 3.16 Speech recording application

3.7.1.2 Phonetic dictionary

A phonetic dictionary is a linguistic tool that specifies how to pronounce words in a language. In other words, a phonetic dictionary makes the correspondence between writing and phonetic form of words in a language. In a continuous speech recognition system, a phonetic dictionary is intended to link the acoustic model (which models how to produce language-specific sounds) and language model (which models the succession of words in a language). As a result, the phonetic dictionary should contain all possible words for the given recognition task and, of course, a phonetic transcriptions of these words.

For the current task (digits recognition from recorded audio waves), the phonetic dictionary must contain transcriptions for only the ten digits of the decimal system: *zero, unu, doi, trei, patru, cinci, şase, şapte, opt and nouă* (including Romanian diacritics).

Below there is a sample of this phonetic dictionary file, already formatted to work with the proposed toolkit (phonemes used for Romanian language are described in Table 2.1:

```
zero z zero e zero r zero o_zero
unu u_unu1 n_unu u_unu2
doi d_doi o_doi i3_doi
trei t_trei r_trei e_trei i3_trei
patru p patru a patru t patru r patru u patru
cinci k1_cinci1 i_cinci n_cinci k1_cinci2 i1_cinci
şase s1_şase a_şase s_şase e_şase
şapte s1_şapte a_şapte p_şapte t_şapte e_şapte
opt o opt p opt t opt
nouă n_nouă o1_nouă w_nouă a1_nouă
```

3.7.1.3 Acoustic model training

Training the acoustic model requires the following resources:

- Audio waves containing speech (previously recorded using the speech GUI on the SpeeD server);
- Corresponding textual transcription of the words spoken in the audio waves;
- A phonetic dictionary containing all the words (the dictionary mentioned in the previous paragraph);
- A dictionary with acoustic elements that are not phonemes usually called *fillers* (silence, cough, laugh, music, etc.).

From all the 100 audio waves recorded per speaker, the first 50 and the last 30 file are used for training. The rest will be used for the evaluation of the system.

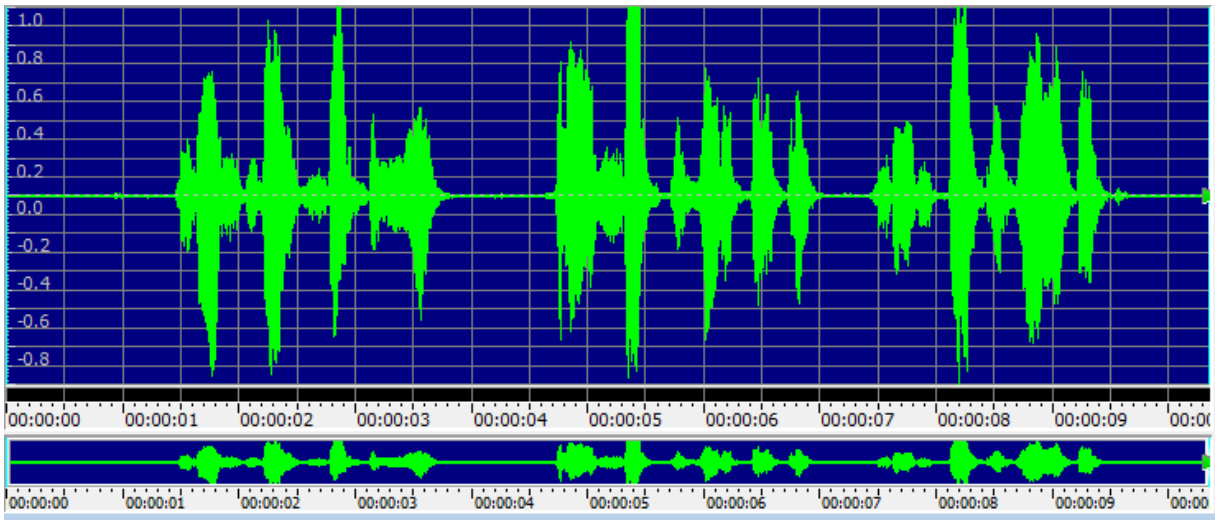

Figure 3.17 Example waveform for the "5261 3704 5408" conversational audio clip, uttered in Romanian

3.7.1.4 Creating the language model

Current systems, trained with a large vocabulary for speech recognition, use an n-gram statistical language model. These language models are built based on large text corpora, specific for the recognition task, estimating the probability of occurrence for words and sequences of words for that task. The n-gram language models are then used in the process of decoding (speech recognition) to select the most likely sequence of words proposed by the acoustic model. The mathematical apparatus behind n-gram language models was described in Chapter 2.

The task of recognizing audio sequences containing digits is a limited vocabulary recognition scenario, which is not suitable for a statistical language model. Furthermore, digits and the succession of digits in the recorded audio clips appear approximately with equal probability (one cannot say that a digit is used systematically more often than the other).

In these circumstances, a finite state grammar (FSG) model is more suitable. A finite state grammar is a graph model in which the nodes represent the language words, and transitions between words are the arcs of the graph. This type of language model explicitly specifies all sequences of words allowed by the recognition task. Moreover, each arc may be assigned a cost specifying the probability that a word is preceded by another (in other words, the probability of the two sequences of words). Figure 3.18 shows the finite state grammar of our recognition task. 14 nodes make up the model, with only 10 nodes representing the digits. The other four are used for "entering" and "leaving" the graph, respectively for a back trace transition. The transitions

show the way to trace this grammar and the word sequences allowed: in every audio clip one or more digits can be spoken.

This type of FSG grammar can be easily implemented using the Java Speech Grammar (JSGF) format. JSGF stands for Java Speech Grammar Format or the JSpeech Grammar Format (in a W3C Note). Developed by Sun Microsystems, it is a textual representation of grammars for use in speech recognition for technologies like XHTML+Voice. JSGF adopts the style and conventions of the Java programming language in addition to use of traditional grammar notations. The Speech Recognition Grammar Specification was derived from this specification.

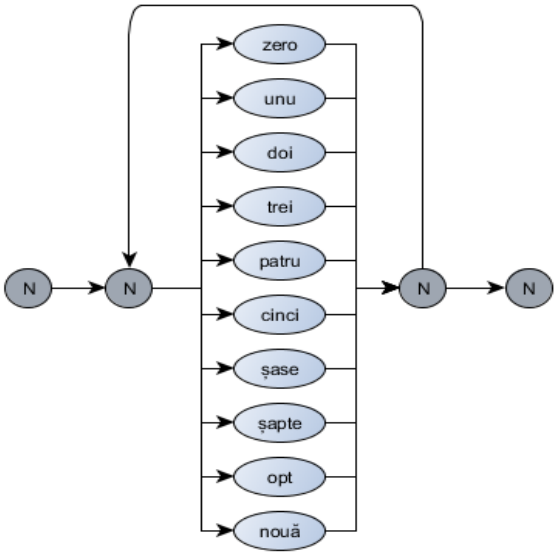

Figure 3.18 Finite state grammar for digits task

3.7.1.5 Decoding

The three basic components of a speech recognition system (acoustic model, language model and phonetic model), mentioned in previous subsections, are available for use in the decoding process. As a result, the system can now decode using the evaluation data, and then compare the textual transcription of the decoding process with the reference transcription. A corresponding report file is generated, with statistics and alignment details, in Table 3.1.

Table 3.1 Alignment details and recognition report

SYSTEM SUMMARY PERCENTAGES by SPEAKER

/home/biosinf01/projects/rodigits/result.cd_cont_100_4/rodigits.match2117								
SPKR	# Snt	# Wrd	Corr	Sub	Del	Ins	Err	S.Err
354	20	240	99.6	0.0	0.4	0.0	0.4	5.0
Sum/Avg	20	240	99.6	0.0	0.4	0.0	0.4	5.0
Mean	20.0	240.0	99.6	0.0	0.4	0.0	0.4	5.0
S.D.	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Median	20.0	240.0	99.6	0.0	0.4	0.0	0.4	5.0

```
id: (354-354_10_0056)
Scores: (#C #S #D #I) 12 0 0 0
REF:  unu nouă șase șase patru trei trei doi patru doi unu șase
HYP:  unu nouă șase șase patru trei trei doi patru doi unu șase
Eval:

id: (354-354_10_0057)
```

```
Scores: (#C #S #D #I) 12 0 0 0
REF:  şapte trei unu doi trei opt cinci trei nouă trei unu şapte
HYP:  şapte trei unu doi trei opt cinci trei nouă trei unu şapte
Eval:
```

3.7.2 Evaluation setup

To build up the connected-digits recognition system, 90 speakers were used to build "roDigits" database (Table 3.2), comprising of 100 audio files per speaker, with a total of 20 hours of recorded speech. The phrases contain 12 spoken digits, in arbitrary order. From the audio files, 80% were used for training, and the rest for evaluation.

Table 3.2 Speaker database summary

Database name:	roDigits
Hours of speech:	20
Number of speakers:	90
Speaker ID:	1 - 90
Audio files per speaker:	100
Training files:	80%
Evaluation files:	20%

Phonemes were modelled in a context-dependent manner. To study the effects of increasing/decreasing the number of senones and Gaussian mixtures per senone, they were varied, according to Table 3.3.

Table 3.3 Number of senones and GMMs summary

Test	No of senones	GMMs
Speaker dependent	100	1/2/4/8/16/32/64/128/256
	200	1/2/4/8/16/32/64/128/256
Speaker independent	100	1/2/4/8/16/32/64/128/256/512

For all ASR experiments presented in this work, we proposed in Table 3 the evaluation setups and their corresponding ids, along with the training and evaluation files used for each setup. These setups where chosen to highlight the importance of training a speaker independent system to avoid mismatch (speakers that are not in the training database), and to select the optimum number of senone states and Gaussian densities, for each task.

The following tests were conducted, for both speaker dependent / independent acoustic models:

Speaker dependent:

- *EvalDepSame1, EvalDepSame2* and *EvalDepSame3* setups were especially created to evaluate the performance of training and decoding with the same speaker (speaker dependent), and choose the optimum number of senones for the next experiments (100 / 200). The tests are identical for all 3 randomly chosen speakers, to validate the results.

- *EvalDepRest1*, *EvalDepRest2* and *EvalDepRest3* use the previously trained models to decode speech from the rest of the roDigits database, to emphasize the mismatch between this model and unknown speech.

Speaker Independent:

- *EvalIndepSame* proposes a model trained with 60 speakers from the roDigits database, to compensate for the high WER with the previous speaker dependent trained models.
- *EvalIndepRest* evaluates the speaker independent model, previously obtained, with the remaining 30 speakers from the roDigits database.

Table 3.4 roDigits evaluation setup

Evaluation setup	Training files	Evaluation files	Setup id
Speaker Dependent	Speaker ID 1	Speaker ID 1	EvalDepSame1
	Speaker ID 59	Speaker ID 59	EvalDepSame2
	Speaker ID 82	Speaker ID 82	EvalDepSame3
	Speaker ID 1	rest of roDigits	EvalDepRest1
	Speaker ID 59	rest of roDigits	EvalDepRest2
	Speaker ID 82	rest of roDigits	EvalDepRest3
Speaker Independent	Speaker ID 1 - 60	Speaker ID 1 - 60	EvalIndepSame
	Speaker ID 1 - 60	Speaker ID 61 - 90	EvalIndepRest

If the speech recognition problem is posed as the transformation of an acoustic signal to a single stream of words, then there is widespread agreement on word error rate (WER) as the appropriate evaluation measure. The sequence of words output by the speech recognizer is aligned to the reference transcription using dynamic programming. The industry standard SCLITE (NIST, 2014) application is used for scoring and evaluating the output of the system. SCLITE is part of the NIST SCTK Scoring Toolkit. The program compares the hypothesized text (HYP) output by the speech recognizer to the correct, or reference (REF) text. After aligning REF to HYP, statistics are gathered during the scoring to output a performance report.

An example report is further detailed in Table 3.5.

Table 3.5 Alignment report using SCLITE

```
SENTENCE RECOGNITION PERFORMANCE

 sentences                    20
 with errors                 5.0%   (1)
   with substitions          0.0%   (0)
   with deletions            5.0%   (1)
   with insertions           0.0%   (0)

WORD RECOGNITION PERFORMANCE

Percent Total Error       =    0.4%   (1)
Percent Correct           =   99.6%   (239)
Percent Substitution      =    0.0%   (0)
Percent Deletions         =    0.4%   (1)
Percent Insertions        =    0.0%   (0)
Percent Word Accuracy     =   99.6%

DUMP OF SYSTEM ALIGNMENT STRUCTURE
.........
id: (354-354_10_0056)
```

```
Scores: (#C #S #D #I) 12 0 0 0
REF:  unu nouă şase şase patru trei trei doi patru doi unu şase
HYP:  unu nouă şase şase patru trei trei doi patru doi unu şase
Eval:
```

Word error rate (WER), sentence error rate (SER) can be consulted in the report, along with a comparison between the reference transcription and the hypothetical transcription of the decoded speech. Number of substitutions, deletions and insertions are also shown, along with detailed information about the substituted words, deleted words, etc. The accuracy of the speech recognizer may then be estimated as the string edit distance between the output and reference strings. If there are *N* words in the reference transcript, and alignment with the speech recognition output results in *S* substitutions, *D* deletions, and *I* insertions, the word error rate is defined as:

$$WER[\%] = \frac{I + S + D}{N} \cdot 100 \tag{3.49}$$

Sometimes the word error rate can be greater than 100% because the above equation also includes the number of insertions. In some applications, a second evaluation metric, the sentence error rate (SER), might also be important depending on the application. The sentence error rate is based on the word error rate and can be computed as follows:

$$SER[\%] = \frac{Sentences\ with\ at\ least\ one\ error}{Sentences\ in\ the\ reference\ transcription} \cdot 100 \tag{3.50}$$

3.7.3 Evaluation results and discussion

Speaker dependent models

a) The following set of results were obtained using a single speaker trained model, with *EvalDepSame1*, *EvalDepSame2* and *EvalDepSame3* setups.

Table 3.6 and Table 3.7 presents the results for the first trained speaker dependent model, EvalDepSame1. Figure 3.19 and Figure 3.20 were plotted, to visually represent the results.

Table 3.6 WER for *EvalDepSame1*

WER [%]		**# GMMs**								
		1	**2**	**4**	**8**	**16**	**32**	**64**	**128**	**256**
No. senone	**100**	2.9	0.4	0.0	0.4	0.8	0.4	2.1	22.5	80.4
	200	9.6	3.3	7.5	12.9	37.9	52.5	77.5	85.8	85.4

Table 3.7 SER for *EvalDepSame1*

SER [%]		**# GMMs**								
		1	**2**	**4**	**8**	**16**	**32**	**64**	**128**	**256**
No. senone	**100**	25.0	5.0	0.0	5.0	10	5.0	15.0	85.0	100.0
	200	40.0	25.0	35.0	50.0	80.0	95.0	100.0	100.0	100.0

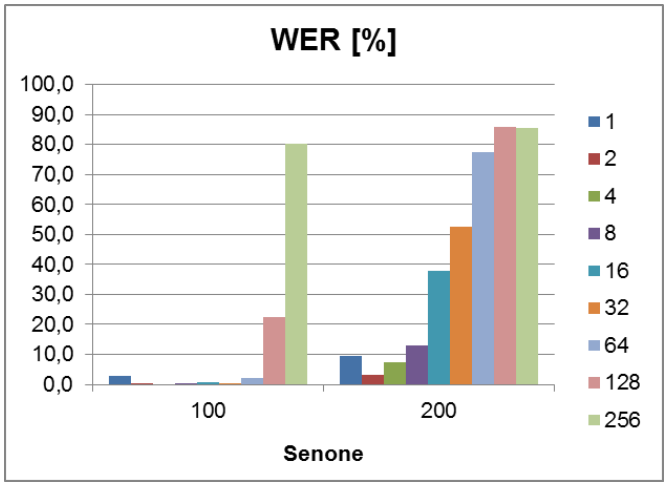

Figure 3.19 Comparison of WER depending on the number of senones and GMMs, for *EvalDepSame1*

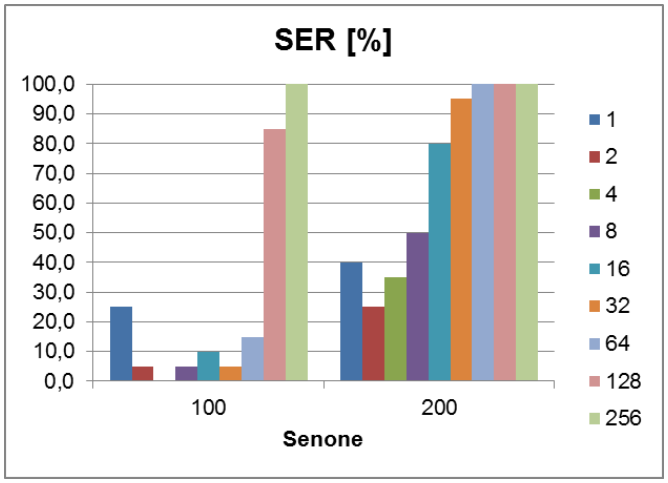

Figure 3.20 Comparison of SER depending on the number of senones and GMMs, for *EvalDepSame1*

Results are consistent with the rest of the randomly selected speakers, as shown in Table 3.8, Table 3.9, Table 3.10 and Table 3.11, for *EvalDepSame2* and *EvalDepSame3* setups.

Table 3.8 WER for *EvalDepSame2*

WER [%]		**# GMMs**								
		1	**2**	**4**	**8**	**16**	**32**	**64**	**128**	**256**
No. senone	**100**	0.4	0.4	0.4	0.4	0.0	0.4	2.1	24.6	89.6
	200	2.9	8.3	9.2	23.8	29.2	40.8	60.0	75.0	94.6

Table 3.9 SER for *EvalDepSame2*

SER [%]		**# GMMs**								
		1	**2**	**4**	**8**	**16**	**32**	**64**	**128**	**256**
No. senone	**100**	5.0	5.0	5.0	5.0	0	5.0	20.0	95.0	100.0
	200	20.0	40.0	40.0	85.0	95.0	95.0	100.0	100.0	100.0

Table 3.10 WER for *EvalDepSame3*

WER [%]		# GMMs								
		1	**2**	**4**	**8**	**16**	**32**	**64**	**128**	**256**
No. senone	**100**	2.1	1.3	0.8	0.8	0.8	1.7	9.6	40.0	79.6
	200	3.8	3.8	5.8	7.5	14.6	41.3	70.0	80.0	87.9

Table 3.11 SER for *EvalDepSame3*

SER [%]		# GMMs								
		1	**2**	**4**	**8**	**16**	**32**	**64**	**128**	**256**
No. senone	**100**	20.0	10.0	5.0	5.0	5.0	15.0	65.0	100.0	100.0
	200	30.0	30.0	35.0	40.0	85.0	100.0	100.0	100.0	100.0

For the task of single speaker identification, 100 senones seems to be the best option, obtaining the best results between 4 and 16 GMMs, depending on the speaker. The more senones a model has, the better it discriminates sounds, and if a high number of senones are set (more than necessary), the model might not be universal enough to recognize unseen speech. WER will be higher on new data, so it is important not over-train the models. The test was run for 3 randomly selected speakers from roDigits, to validate the results.

It is interesting to see how this model, trained with only one speaker (speaker dependent), performs for new speakers, in terms of WER and SER. Next, the same model is used to decode the audio files in a larger data set, from multiple speakers, to evaluate its performance. Senone are set to 100, for the next experiments.

b) The following results were obtained using the previously single speaker trained model, but the decoding was done with the rest of roDigits database, not part of the training process (*EvalDepRest1*, *EvalDepRest2* and *EvalDepRest3*). Tables in this section present the results. Figure 3.21 and Figure 3.22 are plotted for comparison with results from subsection a).

Table 3.12 WER for *EvalDepRest1*

WER [%]	# GMMs								
	1	**2**	**4**	**8**	**16**	**32**	**64**	**128**	**256**
	68.2	74.0	69.4	72.6	73.6	73.9	86.7	94.4	98.8

Table 3.13 SER for *EvalDepRest2*

SER [%]	# GMMs								
	1	**2**	**4**	**8**	**16**	**32**	**64**	**128**	**256**
	97.0	97.0	98.1	98.6	98.7	98.7	98.9	99.8	100.0

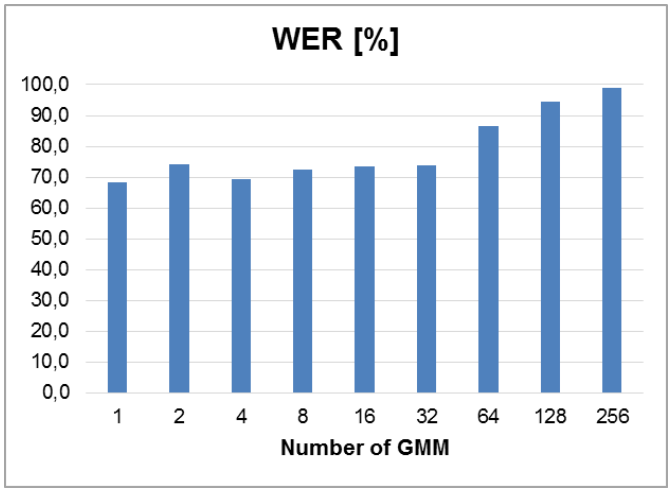

Figure 3.21 Comparison of WER depending on the number of GMMs, for *EvalDepRest1*

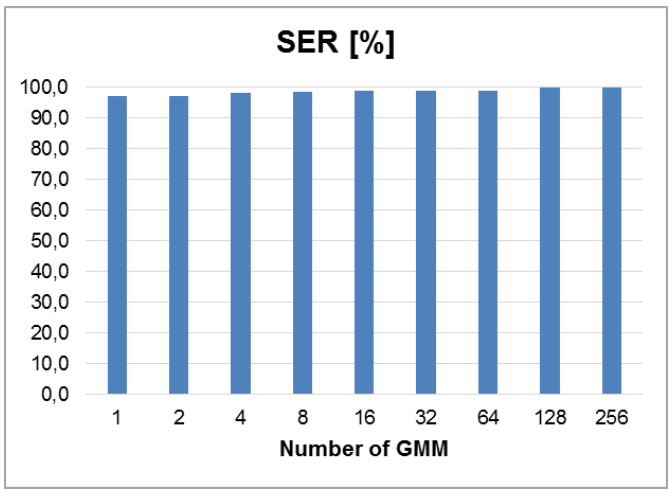

Figure 3.22 Comparison of SER depending on the number of GMMs, for *EvalDepRest1*

Table 3.14 WER for *EvalDepRest2*

WER [%]	# GMMs								
	1	2	4	8	16	32	64	128	256
	57.8	59.9	54.9	60.4	62.7	66.8	79.7	89.4	98.0

Table 3.15 SER for *EvalDepRest2*

SER [%]	# GMMs								
	1	2	4	8	16	32	64	128	256
	87.0	85.6	86.7	89.7	92.0	95.8	99.6	100.0	100.0

Table 3.16 WER for *EvalDepRest3*

WER [%]	# GMMs								
	1	2	4	8	16	32	64	128	256
	65.5	58.4	59.0	65.5	66.0	77.2	81.7	89.5	99.0

Table 3.17 SER for *EvalDepRest3*

SER [%]	# GMMs								
	1	**2**	**4**	**8**	**16**	**32**	**64**	**128**	**256**
	92.0	95.8	97.6	98.0	99.6	99.8	100.0	100.0	100.0

As expected, results show a big difference in error rate, when using multiple speakers for decoding. Results are consistent for all the 3 models trained with data from one speaker, and show a weaker recognition rate in identifying utterances from multiple speakers. A model trained with only one speaker (speaker dependent) cannot be successful in decoding utterances from multiple speakers. A different model must be constructed, with a larger training dataset.

Speaker independent model

A speaker independent ASR system requires a bigger database for the training process. For this purpose, 60 speakers, from roDigits, are used for the training process. Multiple tests were conducted: after training, a different set of audio waves are used for decoding. The results are compared afterwards with a different batch of 30 speakers, which were not used in the training process, to evaluate the performance of decoding unseen speakers.

a) Results from *EvalIndepSame* evaluation setup, trained with multiple speakers, to compensate for the high error rate the previous models offered. The evaluation is done with the same speakers, used for training.

Table 3.18 WER for *EvalIndepSame*

WER [%]	# GMMs									
	1	**2**	**4**	**8**	**16**	**32**	**64**	**128**	**256**	**512**
	10.0	4.9	3.3	2.2	1.8	1.4	1.1	0.9	0.6	0.5

Table 3.19 SER for *EvalIndepSame*

SER [%]	# GMMs									
	1	**2**	**4**	**8**	**16**	**32**	**64**	**128**	**256**	**512**
	53.1	36.5	26.8	18.7	16.1	12.2	10.4	9.3	7.9	5.6

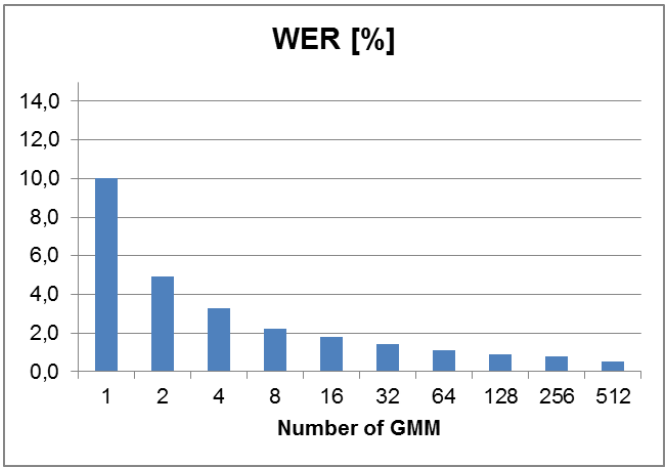

Figure 3.23 Comparison of WER depending on the number of GMMs, for *EvalIndepSame*

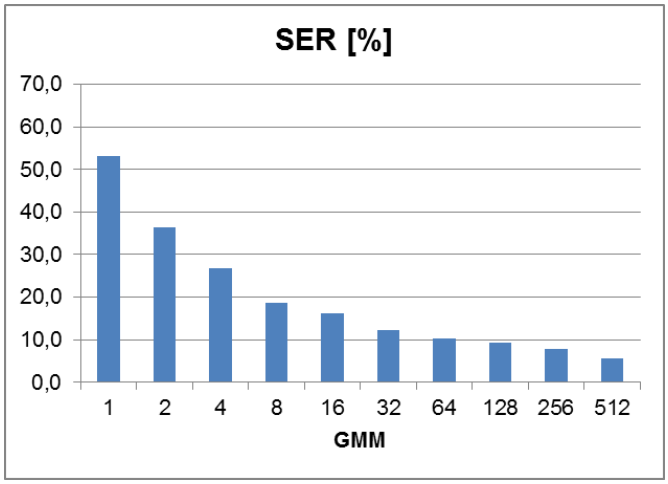

Figure 3.24 Comparison of WER depending on the number of GMMs, for *EvalIndepSame*

The speaker-independent ASR system obtains better results (a lower WER) than the speaker-dependent ASR system. This means that speakers that are contained in the training database are better recognized. In general, the little the mismatch (be it speaker, environment, encoding, etc.) between the training and the evaluation data, the better the results. The best WER is obtained for around 256-512 GMM densities, and there is not much incentive to go further, as the results are in the error interval and the training and decoding time does not justify the gained improvements. These speech recognition results were for "known speakers" (speakers which were also part of the training process), but they might not be as good for "unknown speakers" (speakers to which the system was not exposed during training).

Consequently, the next experiment aimed to evaluate the speaker-independent ASR system on speech uttered by 30 other speakers, which were not part of the training batch.

b) Using the previously speaker independent training data, the decoding is done with the rest of the 30 roDigits speakers to evaluate the model performance. The setup is *EvalIndepRest*, described in Table 3.4.

Table 3.20 WER for *EvalIndepRest*

WER [%]	**# GMMs**									
	1	**2**	**4**	**8**	**16**	**32**	**64**	**128**	**256**	**512**
	20.1	13.8	11.3	10.4	9.4	9.0	8.0	7.6	7.2	7.3

Table 3.21 SER for *EvalIndepRest*

SER [%]	**# GMMs**									
	1	**2**	**4**	**8**	**16**	**32**	**64**	**128**	**256**	**512**
	65.3	48.7	40.0	33.3	24.8	25.2	24.2	24.2	24.0	26.5

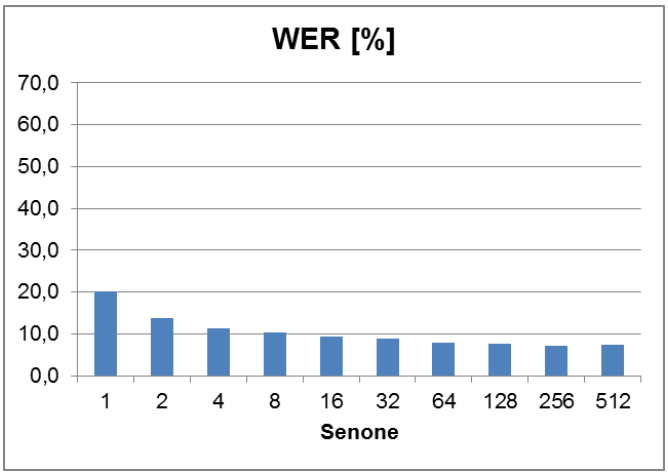

Figure 3.25 Comparison of WER depending on the number of GMMs, for *EvalIndepRest*

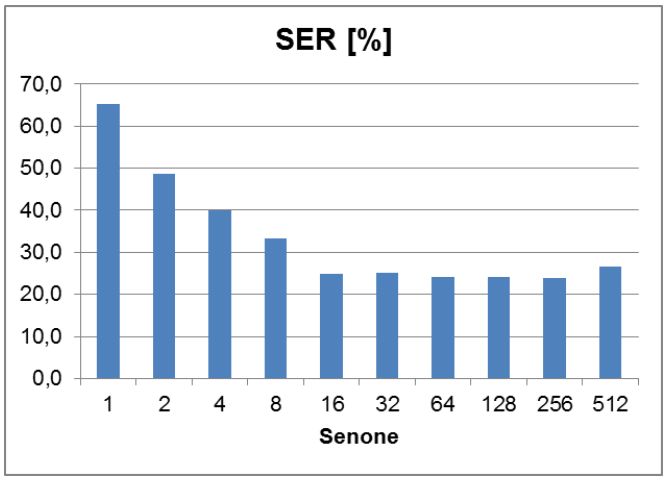

Figure 3.26 Comparison of SER depending on the number of GMMs, for EvalIndepRest

As the above results showed, to obtain a connected-digits recognition system in Romanian language, to successfully recognize unseen speakers, the acoustic model needs to be trained against a larger set of speakers. The number of senones and the number of Gaussian mixtures per senone are variables of the system, to be optimized based on each specific database. For this last experiment, 256 GMMs offers the best results, in terms of error rate, after which more densities cannot successfully model the output distributions, requiring a more detailed model. Depending on the desired error rate, smaller GMMs can be used, which offer faster decoding speed.

Other interesting observations can be taken by looking at word confusion pairs, for this last experiment, in Table 3.22.

Table 3.22 Word confusion pair example

No. of confusions	Confusion pair
5	șapte ==> șase
3	șase ==> șapte
2	nouă ==> unu

The 7 digit (șapte) is often confused with 6 (șase), as only two phones are different in the phonetic transcription. This mistake is usually made by human speakers also, and the language

model can be improved by including, then training the acoustic model, to contain the alternate „șepte" spelling. This alternate spelling is used in numerous telephone conversations, to avoid confusion between these two digits.

3.7.4 Conclusions

This section presented the processes involved in building a fairly representative speech recognition system for decoding connected digits, using speech recorded from multiple speakers to train and evaluate the system. It offered a quick overview of the processes involved in ASR, and in particular, the trainable hidden Markov/Gaussian mixture model (HMM/GMM), for acoustic modelling. Information for improving the models and the training set, along with decreasing word error rate (the primary evaluation metric) are provided. The proposed ASR system can be used in commercial applications, to recognize connected-digits from multiple speakers. An example would be automatic recognition of National Identification Number (CNP – "Cod Numeric Personal"), for certain applications. The commercial success of these speech recognition systems in general, is an impressive testimony to how far research in ASR has come.

3.8 CHAPTER CONCLUSIONS

This chapter offered an overview of the current state of the art algorithms in automatic speech recognition, looking at both statistical and neural network approaches, for a deeper level of understanding in ASR. The application of neural networks is not only limited to acoustic modelling, as it was also successful at improving accuracy over n-gram models by exploiting similarities between words, and thereby estimating reliable statistics even for large n-grams [Bengio, 2003].

But the rich mathematical framework of HMMs makes statistical approaches very relatively easy to implement, thus extremely feasible for ASR, and one of the goals of this chapter was to confirm the validity and reproducibility of this methods. Hence another objective was the integration of the components and toolkits necessary to build a continuous recognition system. I briefly described the processes involved in speech representation, the mathematics behind it and the analysis and experimental setup for improving and optimizing the primary evaluation metrics.

In the second part of the chapter, we also take a look at techniques for automatic phoneme recognition from spoken speech, using NN based approaches (TRAP, STC). The goal is to extract as much information about phoneme from as long temporal context as possible, to be used in pattern recognition applications, evaluated in Chapter 5.

CHAPTER 4

TRANSCRIPTION POST-PROCESSING OF AN ASR SYSTEM

4.1 OVERVIEW OF CURRENT POST-PROCESSING ISSUES

Nowadays, enormous quantities of digital audio and video data are daily produced by TV stations, radio, online video streaming sites and other media. ASR systems can now be applied to such sources of data in order to enrich them with additional information for applications, such as: indexing, cataloging, subtitling, translation, multimedia content production and even on-screen reading by a person.

But the output of an Automatic Speech Recognition (ASR) system consists of raw text, often in lowercase format and without any punctuation information. The transcript is intended to be as close as possible to the speech content of the audio file [Buzo, 2014]. This may be useful for a wide range of applications, such as database indexing and classification (as presented in Chapter 6), where a machine uses this information in search related algorithms. For other tasks, where humans need to easily read and understand the text (e.g. subtitling, dictation and broadcast news transcription), capitalization, punctuation and diacritics restoration greatly improves the readability of automatic speech transcripts. Apart from the insertion of punctuation marks, diacritics and capitalization, enriching speech recognition covers other activities, such as detection and filtering of disfluencies, sentence segmentation, speaker diarization, abbreviation restoration, etc.

This chapter overviews some of the proposed methods in Romanian literature to restore diacritics and diarize speech, then proposes an n-gram based method for capitalization and punctuation restoration for Romanian language.

4.2 OUTPUT RESTORATION FOR ROMANIAN LANGUAGE

The general architecture of an ASR system is shown in Figure 3.2, and thoroughly discussed in chapter 3, along with the implementation of such a system. Apart from the automatic speech recognition core, most systems nowadays also contain a speech pre-processing frontend, responsible with voice activity detection and speaker diarization. The input of this frontend is then recognized through the ASR system, then a post-processing framework analyzes the transcription and it`s responsible for its reformatting.

Voice activity detection (VAD) is needed in order to extract speech from other audio segments, such as music, noise, etc. Only the speech segments will be further processed in an ASR system. The voice activity detection block associates the output speech segments with timestamps relative to the initial speech signal. This timing information can be used in the end to associate speech transcriptions with the various parts of initial speech signal. A phone-based approach for VAD is proposed and used by the thesis author in Chapter 6. With diarization, we identify multiple non-competing speakers in a conversation, by generating speech segments associated with speaker information (speaker ids). This is critical if we need to answer the question “who spoke when?”, and associate speech transcriptions with the corresponding speakers, preserving time information along the way.

In the final step, a transcription post-processing framework should use the speaker information and the timestamps associated with the raw, unformatted transcriptions to organize them into paragraphs, insert punctuation marks and capitalize the text, and additionally, if required, format dates, and numbers.

A pre-processing frontend for Romanian is proposed in [Buzo, 2014], and [Cucu, 2015] offers an overview of its work with integrating multiple text reformatting tools in a comprehensive transcription post-processing framework, for Romanian.

4.2.1 Speaker Diarization

The process of dividing the audio recording into homogenous segments and providing labels with information about them is called diarization. The advantages are multiple, and some were mentioned in the introduction section. By providing audio segmentation, the ASR output becomes more intelligible, because in most cases the segment's bounds coincide with the end of a sentence, and this poses great issues to a human reader. This process cannot be easily solved by a typical VAD system, as this systems many times fail in removing music intervals and generate insertions in the WER metric [Buzo, 2014].

Generally, most current diarization systems groups the speech segments into hierarchical clusters, according to speaker similarities, differing by the clustering algorithm used or evaluation metrics. Methods based on Bayesian Information Criterion (BIC) followed by Cross Likelihood Ratio clustering are presented in [Barras, 2006; Deleglise, 2009], and showed to perform well on broadcast news. Systems based on HMM-BIC (Hidden Markov Models - BIC) [Pardo, 2007] or T-test distance [Nguyen, 2008] obtain better results with meeting recordings, while methods based on E-HMM [Meignier, 2006] obtain better results on telephone conversation recordings [Buzo, 2014].

For Romanian, the first large vocabulary ASR system with diarization was developed by [Buzo, 2014], from Speech and Dialogue Research Laboratory [SPEED, 2015]. The ASR system

used is the one he one developed by the SpeeD Research Laboratory for the Romanian language [Cucu, 2011a]. “It uses an acoustic model trained with 54 hours of speech and represented by a HMM pool with 4,000 senones and 16 Gaussian components. The acoustic features used are the MFCC with the energy and the first and second order derivatives. The language model has 64,000 distinct words and is trained with 169 million words”. For diarization, the paper authors integrated the LIUM system [Meignier, 2010], based on extensive documentation and open source policy. The LIUM toolkit is a state-of-the-art system for speaker diarization composed of multiple steps. First, music and other regions are removed using a Viterbi decoding. Next, an acoustic segmentation followed by a Hierarchical Agglomerative Clustering (HAC) splits and then groups the signal into homogeneous parts according to speakers and background. In this step, each segment or cluster is modeled by a Gaussian distribution with a full covariance matrix and the Bayesian Information Criterion (BIC) is employed both as similarity measure and as stop criterion. Then, a Gaussian Mixture Model is trained for each cluster via the Expectation-Maximization algorithm. The signal is then re-segmented through a Viterbi decoding. The system finally performs another HAC, using the Cross-Likelihood Ratio (CLR) measure and GMMs trained with the Maximum A-Posteriori algorithm [Meignier, 2010]. This architecture is presented in Figure 4.1.

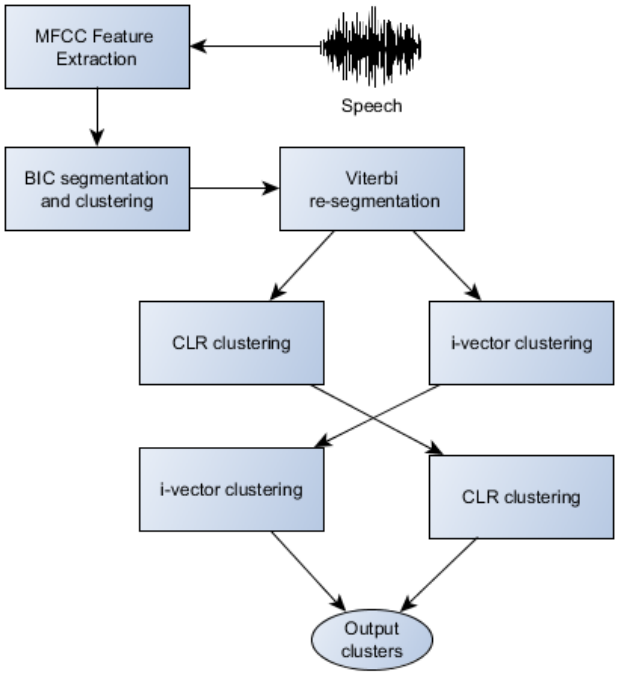

Figure 4.1 LIUM diarization system

In the proposed Romanian system, the LIUM system was modified and integrated in the groups ASR system, as follows: a) features extraction is removed, as this was already performed by the ASR, b) CLR clustering used in gender detection was removed and output clusters are passed to the Speaker Recognition component. Some further modifications in the ASR system were required so that the decoder could be aware of the diarization information and the speech start signals.

Overall, diarization significantly improves the intelligibility of the ASR output, as one can see in Table 4.1, and it`s better than an arbitrary statistical segmentation that can be used instead.

Table 4.1 Comparison of ASR output with / without diarization, on a Romanian news paragraph

Raw ASR output
pe douăzeci aprilie două mii treisprezece la palatul parlamentului din bucureşti a avut loc o conferinţă de presă la conferinţă au participat peste optzeci de persoane din marile oraşe ale ţării timişoara cluj-napoca iaşi şi altele premierul victor ponta şi preşedintele României traian băsescu au prezentat un plan comun de rezolvare a problemelor ţării printre altele s-a discutat despre restituirea unei tranşe de cinci virgulă douăzeci şi şapte la sută din datoria externă a României adică suma de cinci milioane o sută de mii de euro
ASR output with diarization
Speaker #1: Pe 20 aprilie 2013. La Palatul Parlamentului din Bucureşti. A avut loc o conferinţă de presă. La conferinţă au participat peste 80 de persoane din marile oraşe ale ţării, Timişoara cluj-napoca, Iaşi şi altele. Speaker #1: Premierul Victor Ponta şi preşedintele României, Traian Băsescu. Au prezentat un plan comun de rezolvare a problemelor ţării . Printre altele s-a discutat despre restituirea unei tranşe de 5,27%. Din datoria externă a României, adică suma de 5.100.000 de euro.
Ideal ASR output
Speaker #1: Pe 20 aprilie 2013, la Palatul Parlamentului din Bucureşti, a avut loc conferinţă de presă. La conferinţă au participat peste 80 de persoane din marile oraşe ale ţării: Timişoara, Cluj-Napoca, Iaşi şi altele. Speaker #1: Premierul Victor Ponta şi preşedintele României, Traian Băsescu, au prezentat un plan comun de rezolvare a problemelor ţării. Printre altele s-a discutat despre restituirea unei tranşe de 5,27% din datoria externă a României, adică suma de 5.800.000 de euro.

4.2.2 Diacritics restoration

In languages that use diacritical characters, if these special signs are stripped-off from a word, the resulted string of characters may not exist in the language, and therefore its normative form is, in general, easy to recover. However, this is not always the case, as presence or absence of a diacritical sign attached to a base letter of a word which exists in both variants, may change its grammatical properties or even the meaning, making the recovery of the missing diacritics a difficult task, not only for a program but sometimes even for a human reader. In Romanian, although the language contains only 5 diacritical characters (ă, â, î, ş, ţ), their occurrence is as high as 40% [Cucu, 2011a]. So every second word might contain at least one diacritical character and for large texts that lack diacritics, to insert them manually is highly time-consuming and error prone.

Diacritics restoration is a demanding text processing operation, and not a trivial task for a computer, although for a human reader it may seem easy to understand a text without diacritics. Restoring them is important, not only from a grammatically point of view, but also to solve ambiguous situations mentioned above, where words that can be written with several diacritics patterns, like the word fata / față (girl / face) can change the meaning of the sentence.

In Romanian literature, several diacritics restoration methods were developed, knowledge-based or statistical. Some methods are only interested in the character-level context, while others perform better if the full word-level context is given. The amount of training resources is also an important factor, as this generally approximates the cost of developing a diacritics restoration system, given the method. A statistical approach is described in [Mihalcea, 2002], and uses a character n-gram model and experiments with a memory-based learning system, with a decision based classifier, to achieve a precision of 98.3%. It used a medium sized corpus and no word level assumption, because a character n-gram model was used.

A knowledge-based diacritics restoration method, using part-of-speech tagging to disambiguate the different diacritical words hypotheses, is introduced in [Tufiş, 1999] and refined

in [Tufiş, 2008]. It achieves a WER of 2.25%, with more resources then [Mihalcea, 2002], but restoration accuracy is a little better.

In [Ungurean, 2008] the diacritics restoration system is regarded as a sequential filtering process based on unigrams and bigrams of diacritical words and trigrams of diacritical word-suffixes. This method needs only a medium size text corpus to train the various language models and to create a map connecting the non-diacritical word forms to all their diacritical word forms. An updated paper by the same author [Ungurean, 2011] reports a word error rate of 1.4%.

[Cucu, 2011a] proposed a statistical language modeling method for training and restoration, as shown in Figure 4.2.

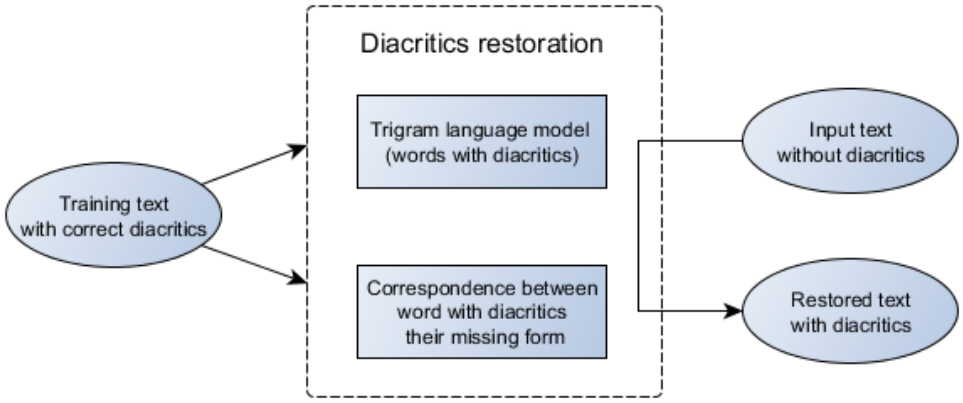

Figure 4.2 Diacritics restoration system architecture

Based on the training corpus containing correct diacritics, two higher-level structures are built: an n-gram language model and a probabilistic map (which links all non-diacritical word forms to all their possible diacritical word forms). This system achieved a WER of 1.5% with a tri-gram language model. This method is further enhanced in [Petrica, 2014], using an unreliable corpus of raw text data, acquired from the web. Through a process of filtering, the raw text corpus is divided into a trusted sub-corpus and an untrusted sub-corpus, with respect to the use of diacritics. The trusted corpus is used to train a diacritics restoration system. Using this system, diacritics are restored to the untrusted sub-corpus, which is then used in conjunction with the trusted corpus to train a language model for automatic speech recognition, obtaining a WER as low as 0.52.

4.3 CAPITALIZATION AND PUNCTUATION RESTORATION FOR ROMANIAN LANGUAGE

Capitalization, also known as true casing, is the process of restoring case information to badly-cased or non-cased text. Punctuation recovery or restoration is the process of inserting punctuation marks (at least periods and commas) in a punctuation-lacking text. In this section, we present an integrated capitalization and punctuation restoration solution for a Romanian ASR system. The solution implements both tasks in a single system (framework) and uses statistical information from a set of tri-gram language models as a post-processing stage. The integrated system is evaluated in terms of precision, recall and f-measure.

In ASR systems, researchers in the field tried to use prosodic information, disfluencies and overlapping speech to predict punctuation, and later they have supplemented these techniques with language models [Baldwin, 2009].

According to [Gravano, 2009], the approaches based on acoustic and prosodic information significantly outperform the methods based purely on n-gram models [Gravano, 2009]. This is concluded after multiple experiments with data-driven techniques for annotating transcribed

speech with sentence boundaries [Shriberg, 2000][Liu, 2006], and with sentence boundaries and other punctuation symbols, predominantly commas and question marks [Brown, 2002; Christensen, 2001; Favre, 2008]. At the same time, digitized text data is growing exponentially in volume, and the availability of massive amounts of written data, coupled with progress in computational power and storage capacity ("cloud model"), asks the question of the extent to which text-based models may be improved when increasing both the training data size and the n-gram order [Gravano, 2009]. Such text data is often produced automatically (ex. via speech recognition or optical character recognition) or in a hurry or unstructured manner (like instant messaging or web user forum data). Hence this data contains noise and needs to be first cleaned and processed in order to obtain any usable data for training, corpus creation, etc.

In this section we describe a set of experiments regarding language model generation, training and evaluation in the context of capitalization and punctuation recovery for the Romanian language. Although the methodology is not new, to the best of our knowledge this is the first such system developed for the Romanian language and these are the first re-capitalization and punctuation restoration results reported for this language. The n-gram language models are trained with data varying from 44 million to 290 million words of written Romanian text. The training and evaluation data consists of broadcast transcriptions and online news and was previously collected by the research group where the thesis author is affiliated. The restoration system was eventually integrated into our large-vocabulary automatic speech recognition for Romanian.

4.3.1 Related Work

Spoken language is similar to written text in many aspects, but differs due to the way these communication methods are produced. Current ASR systems are evaluated based on the WER (introduced in Chapter 3), which does not take into account the detection of structural information available in written texts. As a result, case and punctuation restoration was a relatively unexplored field until this decade [Baldwin, 2009].

One of the first systems to use a simple hidden Markov model with trigram probabilities to model the comma and restoration problem was "cyberpunc", a lightweight method for automatic insertion of intra-sentence punctuation into text [Beeferman, 1998]. It restored the punctuation of 54% of the sentences correctly. Further work was done in [Shieber, 2003] using syntactic information. This paper improves on previous study to achieve an accuracy of 58% for comma restoration. In both of these studies, sentence boundaries are assumed to be given at the input of the processing system. Because the above mentioned methods deal with punctuation restoration at the sentence level, this simplifies the task significantly, as the sentence boundaries are needed as a constraint, resulting in systems that are unable to process large quantities of raw unprocessed ASR text. Regarding case information, [Lita, 2003] proposed a language model-based case restoration method, and the truecaser agreement with the original reference text is about 98%. The high precision reported in the quoted paper can be used as an indicator that the case restoration task is simpler when compared to the punctuation restoration task.

Regarding punctuation marks, a large number can be considered for ASR output texts, including: comma, period or full stop, exclamation mark, question mark, semicolon etc. However, most of these marks rarely occur and are quite difficult to insert or evaluate. Therefore, most of the available studies focus either on full stop or on full stop and comma, which have higher corpus frequencies [Batista, 2008]. A number of recent studies also consider the question mark [Gravano, 2009], and even fewer consider other punctuation marks, such as exclamation marks.

A more recent study by [Gravano, 2009] is of particular interest, not only because it uses n-gram language models, but also because of the large amount of training data, from 58 million to 55 billion tokens. He concludes that a) increasing the n-gram order does not significantly improve capitalization results and b) increasing the size of the training data improves both

precision and recall for capitalization. These conclusions, combined with the adaptation of the training data set to domain specific data [Chelba, 2006], were taken into account in our development of the restoration methodology for capitalization and punctuation. Gravano's study obtained a mean precision and recall of ~81% / 77% for capitalization, ~46% / 42% for comma and ~56% / 48% for period, for broadcast news reference transcript.

The results from the above mentioned study [Gravano, 2009] served as a baseline benchmark of our system. We also note that all statistical capitalization and punctuation restoration systems presented in previous work utilize large amount of domain specific text corpora for training.

4.3.2 Methodology

Much of the prior research on punctuation restoration using n-gram models has been based largely on human transcriptions of speech, and so it has focused on retrieving / using textual information to train the language model. A language model describes possible word sequences, for the purpose of speech recognition and other language technologies. Statistical language modeling (SLM) attempts to capture regularities of spoken language in order to improve the performance of various natural language applications [Rosenfeld, 2000]. We use SLM to estimate the probability distribution of various linguistic units (such as word tokens) and sequences of linguistic units. The language model decomposes the probability of a sentence (s) into a product of conditional probabilities:

$$\Pr(s) = \Pr(w_1 ... w_n) = \prod_{i=1}^{n} \Pr(w_i \mid h_i) \tag{4.1}$$

where w_i is the i-th word in the sentence, and $h_i = \{w_1, w_2, \ldots w_{i-1}\}$ is called a history and, in this case, is a string of i tokens. An n-gram reduces the dimension of the estimation problem by modeling the language as a Markov source of order n-1:

$$\Pr(w_i \mid h_i) \approx P(w_{i-n+1}, \ldots, w_{i-1}) \tag{4.2}$$

where the approximation reflects a Markov assumption that only the most recent n−1 tokens are relevant when predicting the next token. We train with a value of n=3, as trigrams are a common choice with large training corpora (millions of tokens) [Rosenfeld, 2000], and [Gravano, 2009] shows that increasing the n-gram order does not help as much as increasing the training data set. A language model quality is measured by its effect on the specific language application for which it was designed, namely by improving the word error rate of that application. This is influenced by the quality of the n-gram language model. For under-resourced languages, like Romanian, significant efforts were made by the "SpeeD" group in recent years to increase the quality and size of the recorded speech corpus in Romanian, collect and pre-process new information from the Internet, in order to obtain better performing n-gram models after the training process [Cucu, 2014].

Figure 4.3 presents an overview of the system architecture and illustrates a) the role of the capitalization and punctuation restoration module, as a post-processing module for the transcripts resulted out of an automatic speech recognition process and b) the training processes which need to be employed to generate the n-gram language model.

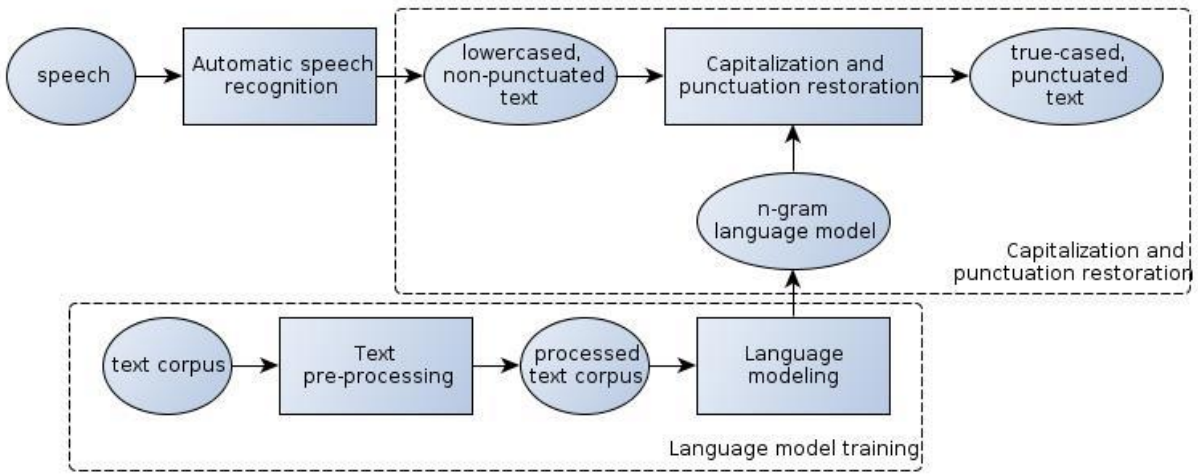

Figure 4.3 Capitalization and punctuation restoration module

The algorithm used in the capitalization and punctuation restoration module processes the input text line by line. The words on each line of input text are processed one by one, from left to right. The n^{th} word on the input line is appended to the existing sequences of n-1 words for that particular line. The word is appended to each existing sequence in all its possible capitalization forms: lowercased (e.g. popa), capitalized (Popa) and all-caps (POPA) and followed by all the possible punctuation marks took into account by our study: no punctuation mark, comma, period. For example, suppose that one of the existing sequences of words for the current line of text is "M-am întâlnit cu" and that the next input word is "popa". The current sequence of three words is expanded into the following 6 sequences of four words:

1. M-am întâlnit cu popa
2. M-am întâlnit cu Popa
3. M-am întâlnit cu popa <period>
4. M-am întâlnit cu Popa <period>
5. M-am întâlnit cu popa <comma>
6. M-am întâlnit cu Popa <comma>

In this example the word form "POPA" is not a valid word in Romanian so it is not taken into account when forming the sequences of four words. After the new sequences are generated, their probability is scored using the n-gram language model. Whenever the number of word-sequences of a certain length exceeds a given threshold, the list of sequences is pruned (the sequences with the lowest probabilities are discarded). After all the words on the input line were processed, the word sequence with the highest probability is sent to the output. This algorithm is inspired from a similar implementation within the CMU Sphinx ASR toolkit [Sphinx, 2015].

The success of the above algorithm is directly influenced by the quality of the n-gram language model. The language model needs to model as well as possible the probabilities for the words and punctuation tokens. The key features in the language modeling part are the n-gram order and the size, quality and adequacy of the training text corpus. The methodology we propose for text pre-processing and conditioning operations (for the Romanian language) are the following:

1. Diacritics and hyphens uniformization. Generally in Romanian texts there are several character codes (incorrectly) used for the same diacritical characters (e.g. ã, ă; ș, ş; etc.) and several hyphen character codes (wrongly) used to form compound words. For a correct computation of statistics for diacritical and hyphen words and word sequences, these characters have to be used in a consistent manner.

2. Diacritics restoration. Most Romanian texts are incorrectly written without diacritics. For the same reason (correct computation of statistics for diacritical words), these texts have to be conditioned: diacritics to be restored.
3. Replace punctuation marks with a corresponding token. As opposed to the ASR approach, in the case of punctuation restoration, the language model has to model the statistics of punctuation tokens as well. Because in this study we approached the restoration of commas and periods, all the other punctuation marks were mapped to one of these tokens, as described in Table 4.2.

Table 4.2 The correspondence between the punctuation marks and tokens in the LM

Punctuation mark	Token
, ( ) --	<COMMA>
: ; ! . ? ...	<PERIOD>

4.3.3 Evaluation setup

For a thorough evaluation of the proposed restoration methodology we used two large Romanian text corpora previously collected by the "SpeeD" research group over the Internet. As described in Table 4.3, for the training process we used the two corpora separately and together to create three different language models. Table 4.3 also illustrates the number of tokens in each corpus and the average number of tokens per paragraph.

The average number of tokens per paragraph is especially important for evaluation, because the algorithm automatically inserts period at the end of every processed line. In the case of short paragraphs (few words per line) this might artificially increase the punctuation score. We do not see this as a problem for our usage scenario because the average number of tokens in each paragraph is quite large (over 45).

Table 4.3 The language models and training corpora

Language model	Training corpora	Tokens	Token/Paragraph
TalkshowsLM	talkshows	45M	45
NewsLM	news	243M	67
MergedLM	talkshows + news	288M	63

For the evaluation process we used two held-out sets of data from the two Romanian corpora. The evaluation corpora contain 100k paragraphs each, with approximately 4M word tokens each. The evaluation corpora were pre-processed exactly as the training corpora (see Section 3) to become similar to the real output of a speech recognition system. In addition to this pre-processing operation, the evaluation data goes through an extra process to remove (lowercase) capitalization and eliminate all punctuation marks.

As performance figures, we used the standard criteria for evaluation of punctuation restoration and capitalization: precision, recall and f-measure:

$$precision = \frac{C}{C+I} \cdot 100 \tag{4.3}$$

$$recall = \frac{C}{C + D} \cdot 100 \tag{4.4}$$

$$f - measure = \frac{precision \cdot recall}{precision + recall} \cdot 100 \tag{4.5}$$

In these equations, *C* represents the number of correct tokens, *I* is the number of insertion errors and *D* is the number of deletion errors or missing tokens.

For capitalization, the correctly capitalized words are counted as correct (C), the words that are capitalized in the reference, but not in the hypothesis are counted as deletions errors (D) and the words that are wrongly capitalized in the hypothesis and are not capitalized in the reference are counted insertion errors (I). Table 4.4 shows an example performance measures for both capitalization and punctuation restoration.

Table 4.4 Example of evaluation procedure for punctuation restoration and capitalization

REF: Acesta este un exemplu , de calcul . HYP: Acesta , este un exemplu de calcul . I D C
REF: Acesta este un Exemplu de CALCUL HYP : Acesta Este un exemplu de CALCUL C I C D C C

4.3.4 Evaluation results and discussion

Table 4.5, Table 4.6 and Table 4.7 contain the results. As stated in previous section, the two test corpora were evaluated against all three trained language models.

Accuracy is measured for words only (capitalized and non-capitalized), excluding punctuation marks, in the entire hypothesis output:

$$accuracy = \frac{C}{T} \cdot 100 \tag{4.6}$$

where *C* is the number of correct words and *T* is the total number of words.

Table 4.5 Precision, Recall and F-measure for the talkshows evaluation corpus

Language Model	**Capitalization**		
	Precision	**Recall**	**F-measure**
TalkshowsLM	80%	69%	74%
MergedLM	76%	72%	73%
NewsLM	72%	71%	71%
Language Model	**Comma**		
	Precision	**Recall**	**F-measure**
TalkshowsLM	55%	46%	50%
MergedLM	56%	43%	48%
NewsLM	50%	38%	43%
Language Model	**Period**		
	Precision	**Recall**	**F-measure**
TalkshowsLM	68%	50%	57%
MergedLM	64%	54%	58%
NewsLM	59%	54%	56%

Table 4.6 Precision, Recall and F-measure for the news evaluation corpus

Language Model	Capitalization		
	Precision	Recall	F-measure
TalkshowsLM	80%	46%	58%
MergedLM	80%	66%	72%
NewsLM	80%	66%	72%
Language Model	**Comma**		
	Precision	**Recall**	**F-measure**
TalkshowsLM	49%	33%	39%
MergedLM	64%	48%	54%
NewsLM	67%	54%	59%
Language Model	**Period**		
	Precision	**Recall**	**F-measure**
TalkshowsLM	68%	34%	45%
MergedLM	68%	53%	59%
NewsLM	67%	54%	59%

Table 4.7 Word Accuracy for both corpuses

Language Model	Word Accuracy for talkshows test data	Word Accuracy for news test data
TalkshowsLM	94%	89%
MergedLM	93%	92%
NewsLM	93%	92%

Figure 4.4 and Figure 4.5 summarize the results and show a visual representation of all the corresponding values in the above tables.

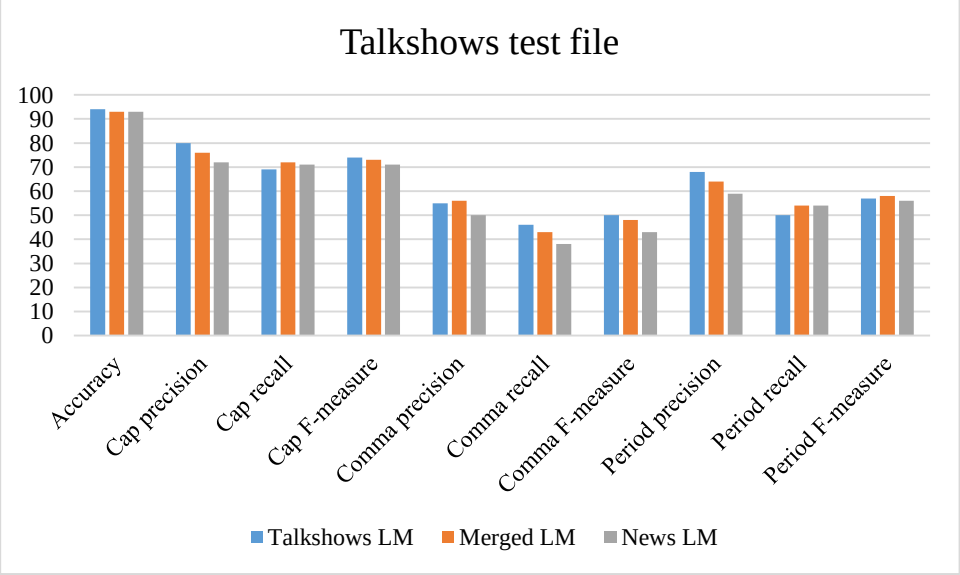

Figure 4.4 Visual representation of talkshows test results

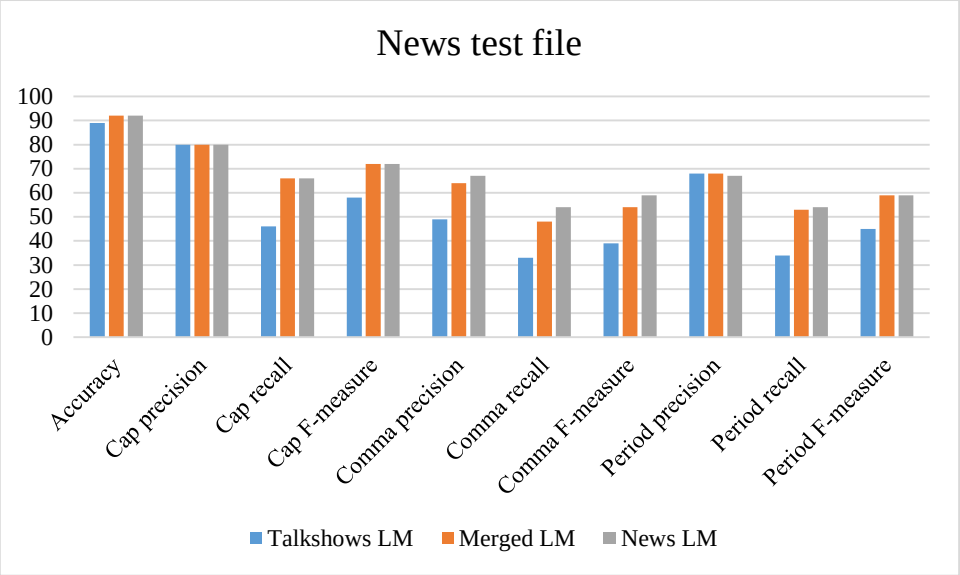

Figure 4.5 Visual representation of news test results

To assess the impact on metrics, we prepared three data sets with a varying numbers of tokens: 45M, 243M and 288M, as shown in previous paragraphs. We trained a *tri*-gram language model for each data set, considering two punctuation tokens. As the visual representation figures show, increasing the corpus size has a positive impact on most of the performance metrics, and can also state that corpus domain has an impact on results. We can conclude that further increasing the size of the training data, coupled with a more complex LM, from the same domain as the evaluation corpora, will presumably increase performance. Table 4.8 further illustrates the impact on readability of a paragraph from Romanian news, where the output from a transcribed news paragraph is compared against the hypothetical and ideal output.

Table 4.8 Comparison of ASR output with / without capitalization and punctuation restoration, on a Romanian news paragraph

Raw ASR output
iată ce spun telespectatorii noștri pe facebook în continuare îi rog să ne trimită propuneri pentru guvernul ponta
ASR output with capitalization and punctuation restoration
Iată ce spun telespectatorii noștri pe Facebook, în continuare îi rog să ne trimită propuneri pentru guvernul Ponta.
Ideal ASR output
Iată ce spun telespectatorii noștri, pe Facebook. În continuare, îi rog să ne trimită propuneri pentru Guvernul Ponta.

4.4 CHAPTER CONCLUSIONS

This chapter offered a quick overview over post-processing means of increasing the output intelligibility of an ASR system. Moreover, the thesis author presented a novel restoration approach to punctuation and capitalization for text in Romanian language, using text-based tri-gram language models.

Overall, our tests show a precision of 76-80% for capitalization restoration, 54-60% for comma and 64-68% for period recovery. Restoration of diacritics in Romanian is necessary for all

future test corpora, if missing. Otherwise, the algorithm will threat capitalized and uncapitalized words as different tokens and fail to restore capitalization for text without diacritics. Furthermore, our results suggest that test files from the same corpora domain offer better results, by a small margin. This margin can be reduced by using a larger training corpus in order to obtain better performing language models.

In conclusion, the punctuation and capitalization restoration tasks greatly improve the intelligibility of the ASR output (as shown in Table 4.8), even though its accuracy and precision are not 100%. The availability of large unstructured data over the internet, that can be downloaded and processed, makes this LM-based restoration principle a feasible method for this task.

To the best of my knowledge, these are the first re-capitalization and punctuation restoration results reported for Romanian language and at the time this thesis was written, we could not find a relevant study for capitalization and punctuation restoration for Romanian. Further work will focus on improving the language models, extending the study on other punctuation marks and on including more complex models based on acoustic/prosodic features from the audio signal.

CHAPTER 5

UNSUPERVISED SPEECH PROCESSING IN LOW RESOURCED LANGUAGES

5.1 OVERVIEW OF INFORMATION PROCESSING AND RETRIEVAL

Information retrieval is a wide, often loosely-defined term, where one ask for a document or piece of information, and an information retrieval system (IR) merely informs on the existence (or non-existence) and whereabouts of documents relating to his request [Robertson, 1976].

Since the 1940s the problem of information storage and retrieval has attracted increasing attention. It is simply to say we have vast amounts of information to which accurate and speedy access is becoming ever more difficult. One effect of this is that relevant information gets ignored since it is never uncovered, which in turn leads to much duplication of work and effort. With the advent of computers, a great deal of thought has been given to using them to provide rapid and intelligent retrieval systems.

In principle, information storage and retrieval is simple. Suppose there is a store of documents (in any format that can contain information, audio or not) and a person (user of the store) formulates a question (request or query) to which the answer is a set of documents satisfying the information need expressed by his question. He can obtain the set by reading all the documents in the store, retaining the relevant documents and discarding all the others. In a sense, this constitutes 'perfect' retrieval. This solution is obviously impracticable. A user either does not have the time or does not wish to spend the time reading the entire document collection, apart from the fact that it may be physically impossible for him to do so.

When high speed computers became available for non-numerical work, many thought that a computer would be able to 'read' an entire document collection to extract the relevant documents. It soon became apparent that using the natural language text of a document not only caused input and storage problems (it still does) but also left unsolved the intellectual problem of characterizing the document content. It is conceivable that future hardware developments may make natural language input and storage more feasible. But automatic characterization in which the software attempts to duplicate the human process of 'reading' is a very sticky problem indeed. More specifically, 'reading' involves attempting to extract information, both syntactic and semantic, from the text and using it to decide whether each document is relevant or not to a particular request. The difficulty is not only knowing how to extract the information but also how to use it to decide relevance. The comparatively slow progress of modern linguistics on the semantic front and the conspicuous failure of machine translation show that these problems are largely unsolved [Rijsbergen, 1995].

Intellectually it is possible for a human to establish the relevance of a document to a query. For a computer to do this we need to construct a model within which relevance decisions can be quantified. It is interesting to note that most research in information retrieval can be shown to have been concerned with different aspects of such a model [Rijsbergen, 1995].

Many Natural Language Processing techniques have been used in Information Retrieval, such as stemming, part-of-speech tagging, compound recognition, de-compounding, chunking, word sense disambiguation, DTW etc. [Brants, 2003]. It is interesting to see how this NLP techniques can be tailored to retrieve spoken documents from audio content, or discover word reoccurrences in a given audio corpus, as this might give a deeper level of understanding to an ASR system. Take, for example, Apple Siri or newly launched Microsoft Cortana. They are able to recognize speech, analyze then retrieve an answer to a question, and they are able to answer questions such as “how is the weather today?” or “when is my meeting scheduled for today?”. This was not possible without document retrieval and processing techniques, and lately this has become a major interest topic in the speech community.

This chapter was motivated by the challenge of searching and extracting useful information from speech data in a completely unsupervised setting. In many real world speech processing problems, obtaining annotated data is not cost and time effective. We therefore ask how much can we learn from speech data without any transcription.

5.2 SPOKEN CONTENT SEARCH

A number of content-based retrieval methods have been explored, including topic detection and tracking, spoken term detection, spoken document retrieval, spoken term discovery and so forth. Research in these directions was supported by multiple evaluation campaigns. In 2006, the U.S. National Institute of Standards and Technology (NIST) created the STD (Spoken Term Detection) evaluation toolkit to facilitate research and development of technology for retrieving information from speech data [Fiscus, 2007]. In recent years, numerous workshops hosted benchmarking initiatives to evaluate new algorithms for multimedia access and retrieval, such as MediaEval (MediaEval, 2011-2015), or as special sessions at relevant conferences in the field of speech communication (ZeroSpeech Challenge, InterSpeech 2015, OpenKWS).

In the following sections we look at two similar approaches for audio retrieval and discovery of speech related data (words reoccurrences, speech queries), in the context of under resourced languages. We approach Spoken Term Discovery and Spoken Term Detection tasks with acoustic modelling of phones obtained from feature types discussed in Chapter 2, and a robust phone recognizer described in Chapter 3. We then use unsupervised methods to search and discover "words" defined as recurring speech fragments (Discovery task) or audio content within audio content (Detection task).

As opposed to the ASR task, there is no general way to evaluate such systems, and due to this fact, the thesis author attended or submitted his systems to popular evaluation campaigns, in the spoken document retrieval domain: The Zero Resource Speech Challenge [ZeroSpeech, 2015] and MediaEval 2015 QUESST - Query by Example Search on Speech Task [MediaEval, 2015].

The ZeroSpeech 2015 evaluation campaign targets the unsupervised discovery of linguistic units from raw speech in an unknown language. Such a task is done within the first year of life by human infants through mere immersion in a language speaking community, but remains very difficult to do by machine, where the dominant paradigm is massive supervision with large human-annotated datasets. The idea behind this challenge is to push the envelope on the notion of flexibility in speech recognition systems by setting up the rather extreme situation where a whole language has to be learned from scratch. At this campaign, the aim of Spoken Term Discovery task is the unsupervised discovery of "words" defined as recurring speech fragments. The systems should take raw speech as input and output a list of speech fragments (timestamps referring to the original audio file) together with a discrete label for category membership. The evaluation will use the suite of F-score metrics described in [Schatz, 2013], which enables detailed assessment of the different components of a spoken term discovery pipeline (matching, clustering, segmentation, parsing) and so will support a direct comparison with NLP models of unsupervised word segmentation.

The Query by Example Search on Speech Task (QUESST) at MediaEval 2015 involves searching for audio speech content within audio content, using an audio content query. This task is particularly interesting for speech researchers in the area of spoken term detection or zero/low-resource speech processing. The task consists in determining how likely it is that a query appears within an audio file. Given an audio file and a spoken query, systems will have to produce a score. The higher the score the more likely is that the query appears in the audio file. The task data is a set of audio files from multiple languages (some resource-limited, some recorded in challenging acoustic conditions, and some containing heavily accented speech), which will are provided to researchers. In addition, two sets of spoken queries (for development and test) will are provided for researchers to build their systems. No transcriptions, language tags or any other meta-data are provided for the development and test corpora (except for a timing of query location inside an utterance). The task therefore requires researchers to build a language-independent audio-within-audio search system.

Both systems and approaches submitted to this evaluation campaigns, by the thesis author, are discussed in the sections to follow. Evaluation metrics specific to each task are discussed and presented, along with results obtained with the proposed systems. Baseline, or best performing systems submitted by competing researchers in the field are presented, for comparison.

5.2.1 Dynamic time warping technique

We make a quick introduction to Dynamic time warping (DTW) in this section, as this technique is used for finding an optimal alignment between two given (time-dependent) sequences under certain restrictions. Originally, DTW has been used to compare different speech patterns in automatic speech recognition [Sakoe, 1978], and it has also been applied to many other fields like bioinformatics, econometrics and handwriting recognition. Basically, in fields such as data mining and information retrieval, DTW has been successfully applied to automatically cope with time deformations and different speeds associated with time-dependent data [Müller, 2007].

DTW is used to compare two (time-dependent) sequences, $X = \{x_1, x_2, \dots, x_N\}$ and $Y = \{y_1, y_2, \dots, y_M\}$ as shown in Figure 5.1, where aligned points are indicated by the arrows.

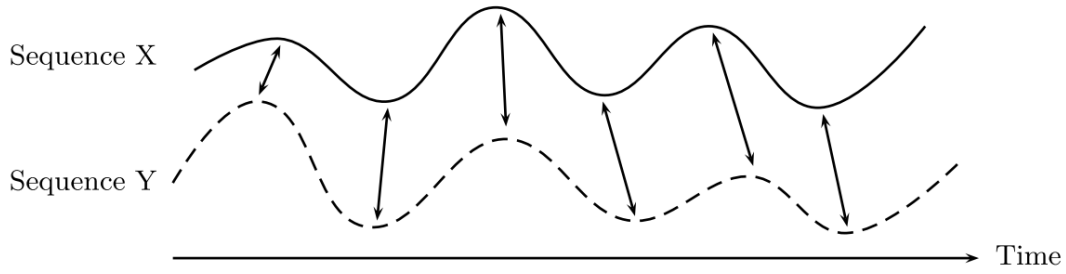

Figure 5.1 Time alignment of two time-dependent sequences using DTW

These sequences may be discrete signals (time-series) or, more generally, feature sequences sampled at equidistant points in time. To compare two different features *x, y*, we need a local cost measure, sometimes also referred to as local distance measure. Typically, *c(x, y)* is small (low cost) if *x* and *y* are similar to each other, otherwise *c(x, y)* is large (high cost).

Evaluating the local cost measure for each pair of elements of the sequences *X* and *Y*, one obtains the cost matrix, defined by *C(n, m)*, as shown in Figure 5.2. The goal is to find an alignment between *X* and *Y* having minimal overall cost, or distance. In this matrix, each cell (*i, j*) represents the distance between the *i*-th element of sequence *X* and the *j*-th element of sequence *Y*. The distance metric used depends on the application, but a common metric is the Euclidean distance.

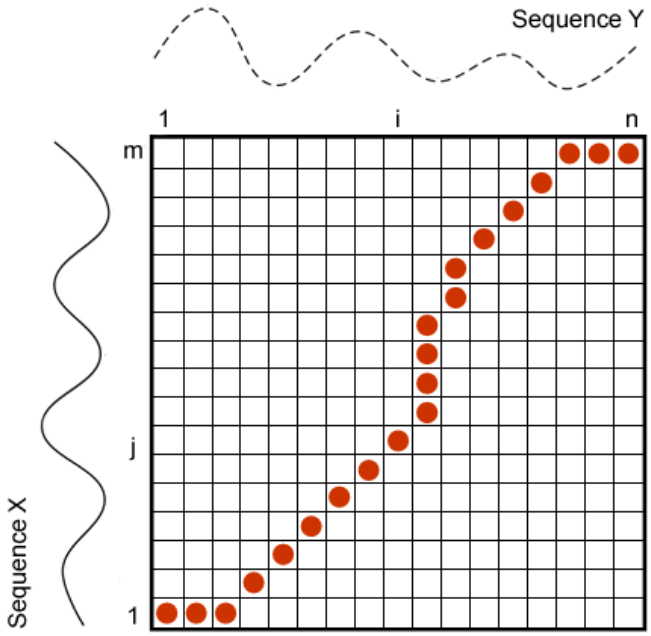

Figure 5.2 Cost matrix for elements of the sequences X and Y

Finding the best alignment between two sequences can be seen as finding the shortest path to go from the bottom-left cell to the top-right cell of that matrix. The length of a path is simply the sum of all the cells that were visited along that path. The further away the optimal path wanders from the diagonal, the more the two sequences need to be warped to match together.

The brute force approach to finding the shortest path would be to try each path one by one and finally select the shortest one. However it's apparent that it would result in an explosion of paths to explore, especially if the two sequences are long. To solve this problem, DTW uses two things: constraints and dynamic programming.

DTW can impose several kinds of reasonable constraints, to limit the number of paths to explore:

- Monotonicity: The alignment path doesn't go back in time index. This guarantees that features are not repeated in the alignment.
- Continuity: The alignment doesn't jump in time index. This guarantees that important features are not omitted.
- Boundary: The alignment starts at the bottom-left and ends at the top-right. This guarantees that the sequences are not considered only partially.
- Warping window: A good alignment path is unlikely to wander too far from the diagonal. This guarantees that the alignment doesn't try to skip different features or get stuck at similar features.
- Shape: Aligned paths shouldn't be too steep or too shallow. This prevents short sequences to be aligned with long ones.

The total cost $c_p(X, Y)$ of a warping path p between X and Y with respect to the local cost measure c is defined as:

$$c_p(X,Y) = \sum_{l=1}^{L} c(x_{nl}, y_{ml}) \tag{5.1}$$

where L is the total number of sequences of warping paths p. Furthermore, an optimal warping path between X and Y is a warping path p^* having minimal total cost among all possible warping paths. The DTW distance $DTW(X, Y)$ between X and Y is then defined as the total cost of p^*:

$$DTW(X,Y) = c_p^*(X,Y) = \min\{c_p(X,Y) \,|\, p \text{ is an } (N,M) \text{ warping path} \tag{5.2}$$

To determine an optimal path p^*, one could test every possible warping path between X and Y. Such a procedure, however, would lead to a computational complexity that is exponential in the lengths N and M, and all algorithms proposed to solve din path are based on dynamic programming. Once the algorithm has reached the top-right cell, we can use backtracking in order to retrieve the best alignment. If we're just interested in comparing the two sequences however, then the top-right cell of the matrix just happens to be the length of the shortest path. We can therefore use the value stored in this cell as the distance between the two sequences.

5.2.2 Spoken Term Discovery Related Work

With the increasing availability of spoken documents in different languages, some of those languages even considered under-resourced in the speech community, there is a growing need for unsupervised methods of information extraction. An appropriate method for this task, spoken term discovery systems identify recurring speech fragments from the raw speech, without any knowledge of the language at hand [Park, 2008], to build classes of similar speech fragments.

Current approaches to spoken term discovery rely on variants of dynamic time warping (DTW) to efficiently perform a search within a speech corpus, with the aim of discovering occurrences of repeating speech (further called "terms" or "motifs") [Park, 2008; Jansen, 2010; Flamary, 2011; Muscariello, 2012]. Applications employing automatically discovered terms have quickly appeared, having a wide focus, ranging from topic segmentation [Malioutov, 2007] to document classification [Dredze, 2010] or spoken document summarization [Harwath, 2013]. Besides the immediate applications it can have in languages with little or no resources, spoken term discovery can also have relevance to cognitive models of infant language acquisition [Jansen, 2013].

In order for the obtained terms to be a viable source of information for any downstream application, they need to be of good quality and to sufficiently cover the target corpus. For this reason, the speech research community has worked towards improving the unsupervised term

discovery process through different methods. Among the various approaches proposed, we mention the use of linguistic information in the input features [Zhang, 2010; Muscariello, 2012], the optimization of the search process [Jansen, 2011], or the introduction of linguistic constraints during DTW search [Ludusan, 2014].

Since spoken term discovery works in an unsupervised manner, the extraction of informative features is an important aspect. Zhang and colleagues [Zhang, 2010] were the first to explore the use of Gaussian posteriorgram representations for unsupervised discovery of speech patterns. They demonstrated the viability of using their approach, by showing that it provides significant improvement towards speaker independence. The investigation into the use of posteriorgrams for spoken term discovery was extended in [Muscariello, 2012], where the authors employed two types of posteriorgrams: both supervised and unsupervised ones, the former being trained either on the target language or on a different language. They showed that for one of their system settings, the posteriorgrams always outperformed the Mel Frequency Cepstral Coefficients (MFCC) features, while for the other setting only the target language supervised posteriors brought improvements over the MFCC baseline.

Taking advantage of the existence of open source recognition systems and the availability of acoustic models trained on different languages [Schwarz, 2015], we built upon the study conducted by Muscariello and colleagues [Muscariello, 2012]. We investigate a larger range of phone-based posteriorgrams, as well as combinations of posteriorgrams coming from different languages. Furthermore, we use the phoneme recognizer output to build an additional feature, a binary phoneme feature vector. The latter feature defines the presence (value 1) or absence (value 0) of a phoneme at a certain time instant, as returned by the speech recognizer. We compare the results given by a spoken term discovery system employing these linguistically-enhanced features with those obtained with an identical system using classical spectral features.

5.2.3 Spoken Term Detection Related Work

The Spoken Term Detection task is similar to the discovery one, only here, a query term is searched into an audio database. If Spoken Term Discovery can be compared with a library, where every book is classified in similar domains automatically, then Spoken Term Detection task is the search part, where the correct book is retrieved from this massive index.

In an ideal information retrieval scenario, the end user should be able to perform open vocabulary search and retrieval in any language, over a large collection of spoken documents, in a front-end application, with results being returned in a matter of seconds. For this reason, most of the systems employ some sort of pre-indexing of the speech corpus, prior to search, without the advanced knowledge of the query terms. Thus, a typical STD system is illustrated in Figure 5.3 and mainly consists of two components: in the pre-indexing phase, a speech recognition subsystem transcribes speech signals into intermediate representations, usually word or sub-word lattices, followed by a detection subsystem that searches for occurrences of the search terms, using a pattern matching or search algorithm (such as DTW). The later subsystem comprises (i) a term detector that searches the indexed content for all potential occurrences of a search term, and (ii) a decision making component that determines if a potential occurrence is reliable enough to be hypothesized as a term match. It is important to note that the recognition subsystem is run only once on the audio database and that the detection subsystem has access only to the decoded content (or lattices), hence the pre-indexing phase.

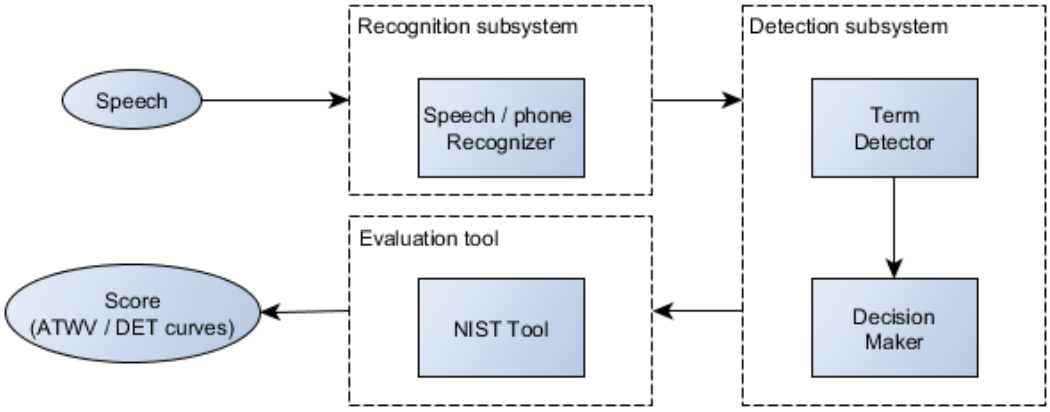

Figure 5.3 Illustration of a typical STD system, where the US NIST Tool is used to evaluate system performance. Adapted from [Dong, 2012]

Much of the prior work, done to date, focused on languages and domains where transcribed speech and phonetic lexicon resources are widely available. Thus, they relied on large amounts of training data, including recordings (for acoustic modeling) and text data (for language modeling) in the target languages. As such, the best current methods make heavy use of word-based speech recognition during the indexing process to build word lattices. The good accuracy of the Automatic Speech Recognition (ASR) systems for high-resourced languages has also assured a high quality STD. As such, these systems have some constraints, and assume well-trained recognizers for the input language, with a search vocabulary to be well covered by the language models used during indexing (low Out-Of-Vocabulary for the query terms). Hence, the recent efforts are concentrated mainly on handling Out-Of-Vocabulary (OOV) words for which the pronunciation is unknown and the language model is unavailable [Parlak, 2008; Parada, 2010; Wang, 2010]. In a recent paper [Wade, 2009], the authors made use of the classic lattice approach to build possible alternatives from both the query term and search indices, suggesting that lattice representations of search indices and queries can still improve STD performance.

To compensate for the languages where resources are scarce (low resourced languages), many state of the art systems also make use of phonetic search and data fusion techniques. Approaches based on subword units (phones), are widely used nowadays to solve the OOV issue. In this approach, subword representations of search terms are searched for within subword lattices that are generated by a subword-based ASR system. Authors in [Ng, 2000; Burget, 2006; Hori, 2007] made significant work with phonetic units for content based retrieval from speech, through a method of confusion networks applied to phones, outperforming lattice-based methods especially for OOV queries.

Regarding multilingual STD, there are a few previous studies [Lee, 2009; Motlicek, 2010]. The first uses an out-of-language module based on confidence measures to detect only the English speech segments. The latter proposes a method for a switch between Chinese and English languages using code-switched lattice-based structures for word/subword units. An alternative solution is to build acoustic and language models that are shared across languages, like [Lin, 2009].

Less work has been done involving methods for speech search by example. In [Murao, 2005], the authors describe a method for example-based query generation for general search. [Buzo, 2013], proposed a query-by-example approach to multilingual Spoken Term Detection for under-resourced languages, based on ASR. The approach overcomes the main difficulties met under these conditions, providing a new method for building multilingual acoustic models with few annotated data. The acoustic models are obtained by adapting well trained phonemes to the ones from the envisaged languages. The mapping is made according to the International Phonetic

Alphabet phoneme classification and a confusion matrix. The weighting of query length and alignment spread are incorporated in the Dynamic Time Warping technique to improve the searching method.

Using data fusion techniques to combine results from diverse ASR systems, one can improve robustness across a variety of talkers, channels, environments and target terms. Various Hybrid approaches which fuse word and subword approaches at the lattice level have also been proposed [Yu, 2004; Meng, 2008].

This study is closely related to works that provide multilingual acoustic models, even though they are mainly used in ASR. All of the methods presented, in this section, show promising performance on the utterance retrieval task. STD remains a challenging task going forward. Unfortunately, state-of-the-art ASR systems are far from being reliable when it comes to transcribing unconstrained speech recorded in uncontrolled environments. Considering the heterogeneous nature of the large spoken databases, it is no surprise that speech retrieval research is mainly about compensating for ASR deficiencies [Can, 2011].

5.3 UNSUPERVISED SPOKEN TERM DISCOVERY EXPERIMENTS

The current experiments employ an open-source spoken term discovery system, called MODIS [Catanese, 2013], based on the systems proposed in [Muscariello, 2012]. The functioning of the system follows the so called seed discovery principle, i.e. to search for matches of a short audio segment in a larger segment, with the search being performed by means of a segmental variant of the dynamic time warping (DTW) algorithm. In this framework the shorter segment is called seed, while the larger one is called buffer.

The algorithm inspects the acoustic sequences present in the buffer to assess whether it contains any repetition of the seed and a matching decision is taken by comparing the DTW score of the path with a DTW similarity threshold. If the computed score is lower than the threshold, the algorithm considers that a match was found. In that case, the seed will be extended and the process repeated using the longer seed, until the dis-similarity between the segments reaches the set threshold. When that happens, the term candidate is stored in the motif library, provided it has passed any length constraints imposed by the system. The algorithm continues parsing the speech looking for matches with respect to the motif library. If no match is found with respect to the motifs in the library, the DTW search process described previously is repeated. When further matches of the same term are found, the corresponding cluster model is updated accordingly. Once the corpus has been parsed in its entirety, found motifs are compared to each other in term of their overlap and overlapping elements are merged into one single term. This process is illustrated in Figure 5.4.

The algorithm has several important parameters that must be set: the “seed size”, the minimum stretch of speech matched against the buffer, the minimum term “size” the algorithm will find, the “buffer size” in which the seed is searched and the “similarity threshold” ϵ_{DTW}. The latter influences the level of similarity between the members of the same term class: the lower the threshold, the more similar the terms will be. The choice of this parameter has to be a compromise between a small number, but highly homogeneous terms, and a larger number of terms with a higher heterogeneity.

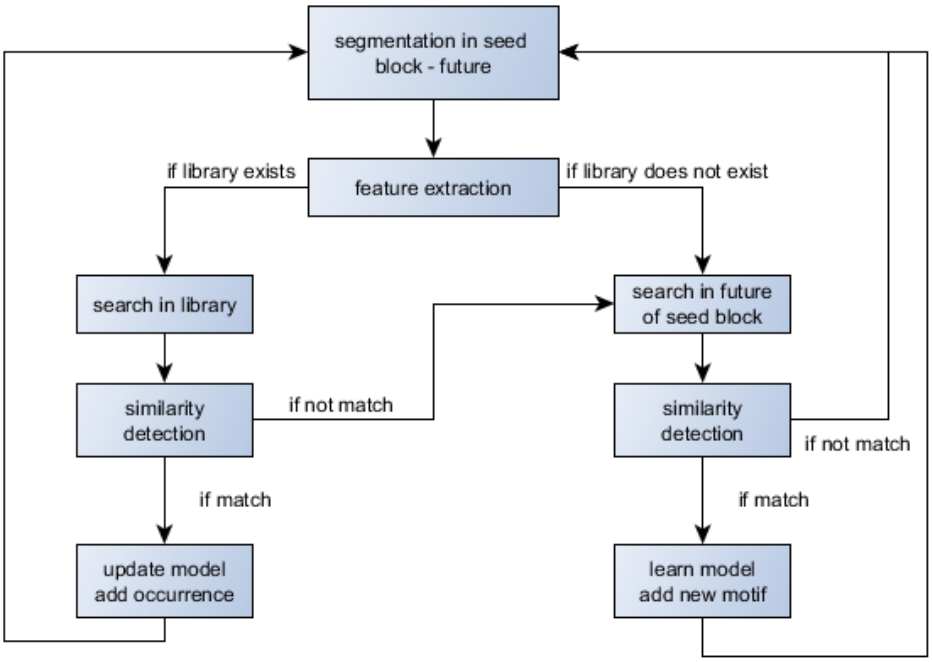

Figure 5.4 Audio motif discovery algorithm architecture, as proposed by [Catanese, 2013]

MODIS was designed to work with several types of input features. For this it implements two different distances: the Euclidean distance, generally used together with spectral representations, and the distance defined in the below equation, when posterior probabilities features are employed:

$$d(a,b) = -\log\left(\sum_{i=0}^{N}(a_i \cdot b_i)\right) \tag{5.3}$$

The terms involved in the above equation represent the two feature vectors for which the distance is calculated (*a* and *b*) and their length (*N*).

5.3.1 Experimental Setup

This section describes the experimental setup and the features (MFCCs and Posteriorgrams) involved in spoken term discovery experiments. Both types of features were discussed in Chapter 2, along with the toolkits necessary for a successful extraction. We reiterate here their most important properties and quickly review the necessary toolkit user in our experimental setup.

We decided to use for our baseline system MFCC features, a standard spectral representation in speech applications. The features were extracted using the HTK toolkit [Young, 2006], for all the datasets used thought the spoken term discovery experiments. The speech signal was analyzed using 25ms long frames with a 10ms frame rate and the original wider frequency band (16 kHz) was used for feature extraction. HTK was configured to compute 39 features (MFCC + Energy + Deltas + Accelerations) per frame. Critical bands' energy was obtained in conventional way.

In our phone-based posterior approach, the state-of-the-art phoneme recognizer based on long temporal context from BUT [Schwarz, 2009] was implemented in our toolkit, to obtain highly accurate phone-based posteriorgrams as features. A phone posteriorgram is defined by a probability vector representing the posterior probabilities of a set of pre-defined phonetic classes for a speech frame, with entries summing up to one. By using a phonetic recognizer, each input speech frame is converted to its corresponding posteriorgram representation.

The BUT toolkit uses a hybrid Hidden Markov Model - Artificial Neural Network (HMM/ANN) approach, to extract as much information about phoneme from as long temporal context as possible. The baseline phoneme recognizer is based on the Temporal Pattern (TRAP) system, with Split Temporal Context (STC) optimizations (Left-Right Context). This is based on the theoretical study that significant information about phoneme is spread over few hundreds milliseconds and that an STC system can process two parts of the phoneme independently. The trajectory representing a phoneme feature can then be decorrelated by splitting them into two parts, to limit the size of the model, in particular the number of weights in the neural-net (NN). The system uses two blocks of features, for left and right contexts (the blocks have one frame overlap). Before splitting, the speech signal is filtered by applying the Hamming window on the whole block, so that the original central frame is emphasized. Dimensions of vectors are then reduced by DCT and results are sent to two neural networks. The posteriors from both contexts are, in the final stage, merged, after the front-end neural networks are able to generate a three-state per phoneme posterior model [Schwarz, 2009].

Regarding Temporal Pattern Processing, spectrum-based techniques form the basis of most feature extraction methods in current ASR systems. A drawback of the spectral features is that they exhibit rapid degradation in performance in realistic communication environments and supplementary techniques need to be applied to address this problem. In a TRAP system, the conventional spectral feature vector in ASR is substituted by a 1 sec long temporal vector of critical band logarithmic spectral energies from a single frequency band, to capture the temporal evolution of the band-limited spectral energy in a vicinity of the underlying phonetic class [Hermansky, 1999].

The results from the toolkit are incorporated in a recognizer module, able to output transcribed speech signals into strings of unconstrained acoustic units (phonemes) and deliver these strings together with temporal labels, which we can further process in our voice activity detector (VAD) tool, or binary phoneme vectors. We use the phoneme recognizer output to build an additional binary phoneme feature vector (Figure 5.5). This vector represents the speech as a sequence of binary values, 1 indicating that at that time instant, a particular phone was found by the recognizer and 0 signaling the opposite case. The phoneme recognizers were trained using four languages: English, with data from the TIMIT corpus [Garofolo, 1993], and three other languages from the SpeechDat-E corpus [Pollak, 2000] (Czech, Russian and Hungarian). The performances of these systems are detailed in [Schwarz, 2009] and summarized in Table 5.1, where *ERR%* represents the system error rate and *NN* is the number of neurons in all nets.

Table 5.1 BUT Recognizer systems used

Rec. system	phones	ERR%	NN
CZ-8k	45	24.24	1500
HU-8k	61	33.32	1500
RU-8k	52	39.27	1500
EN-16k	39	24.24	500

In a first step, the segments given in the output of the recognizer were processed to keep only speech zones, using an embedded voice activity detector module. Intervals for the VAD module were obtained either from the corpus annotation released with the challenge files, for the evaluation set, or calculated from the output of the phoneme recognizer, for the development set. These speech masks were applied on posteriorgrams, MFCCs and phoneme vectors to discard noise and other non-speech events from speech files. The three types of non-speech tokens in the BUT systems: “int” (intermittent noise), “Spk” (speaker noise) and “pau” (silent pause) [Matjka, 2005] were all mapped to silence in our VAD module. The complete feature extraction procedure is illustrated in Figure 5.6.

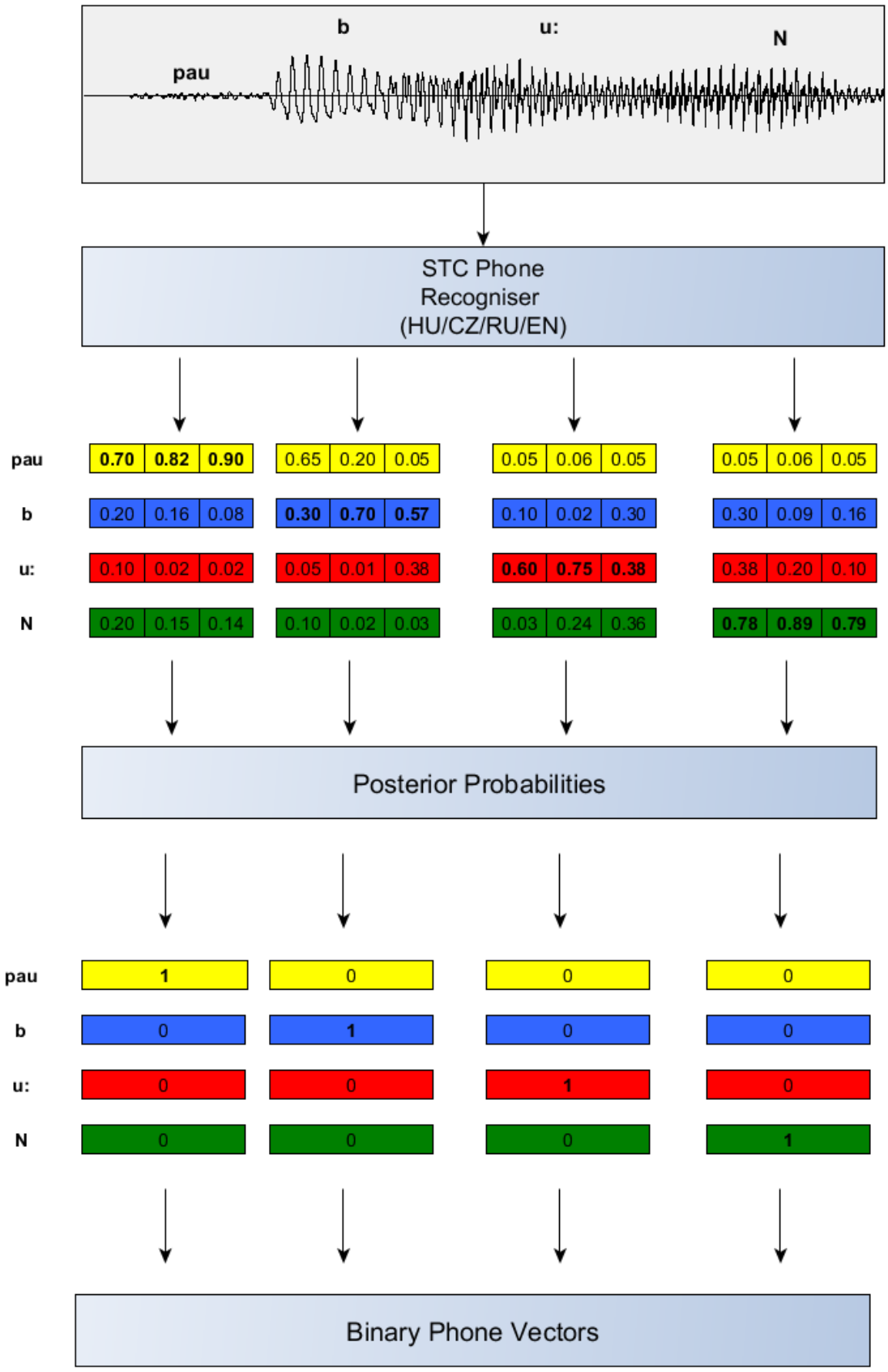

Figure 5.5 Posterior probabilities and binary phone vectors feature types

As features for term discovery we use both MFCCs and information coming from the phone recognizer: posteriorgrams and phoneme vectors. As mentioned in previous paragraphs, we employed the BUT recognizer and acoustic models for the following languages: *Czech (CZ), English (EN), Hungarian (HU) and Russian (RU)*. Down sampling to 8kHz was necessary for all speech files, except for the EN system, to match the recognizer acoustic models used for phoneme training. Besides extracting posteriorgrams and vectors of phonemes for each individual language, we created two combinations of such features, by concatenating the individual vectors into one, large, super-vector. The two combinations tested are the following: *All*, containing the vectors of all four languages, and *AllE*, obtained by concatenating the *CZ*, *HU* and *RU* models outputs. The latter was tested in order to see the effect of a combination of language posteriorgrams on the English data, without using knowledge from that language. The vector of phonemes feature represents the speech as a sequence of binary values, with 1 indicating that at that time instant, a particular phone was found by the recognizer and value 0 representing the opposite case.

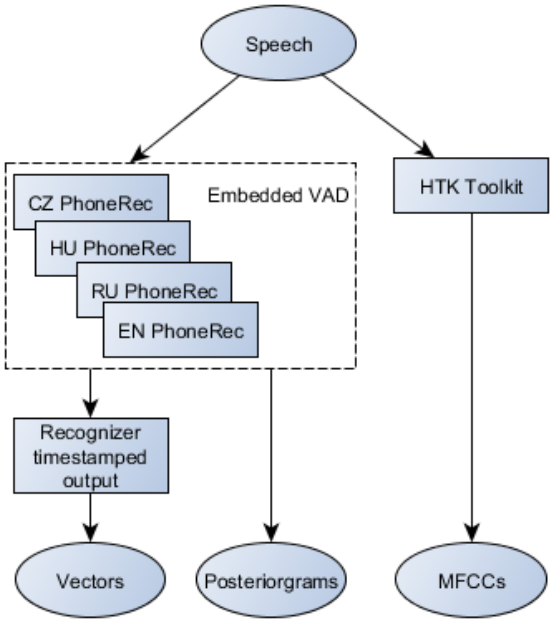

Figure 5.6 Feature extraction module used for audio motif discovery experiments

For the spoken term discovery experiments we varied the similarity threshold, while keeping the rest of the parameters constant. The seed length was set to 0.25 s and the minimum term size considered was 0.5 s in order to able to find entire words. The buffer length to 600 s, as some of the materials used here had fewer repetitions of the same word. The model chosen to represent the term clusters was the median model, while self-similarity matrix checking [Muscariello, 2011] was employed for the matching between the model of the cluster and the seed. Regarding the distances used, the posteriorgram features employed the distance presented in equation 5.3, while the MFCCs and the phoneme vectors used the Euclidean distance.

Before proceeding to the experiments, good values for the term discovery similarity threshold had to be determined, for each of the different features employed. Besides the fact that the features used are quite diverse, the term discovery software uses also different distance functions for them. These differences made even more important the finding of a correct setting for the DTW threshold. For this reason, we employed the sample set released with the challenge as a development set, on which we searched for the optimum threshold value, given a certain evaluation metric. As optimization metric we chose the matching F-score [Ludusan, 2014b], presented in next section, as it characterizes the quality of the matching process, and it rewards systems having both a high precision and a high recall.

5.3.2 Datasets

We used for this chapter experiments the datasets released with the ZeroSpeech 2015 challenge [Versteegh, 2015]: an English dataset and one containing an unknown, surprise language. The English set contains recordings from the Buckeye corpus [Pitt, 2007], while the surprise language, identified later as being Xitsonga, one of the eleven official languages of South Africa, had its material drawn from the NCHLT speech corpus of the South African languages [Vries, 2014].

The Buckeye corpus contains spontaneous speech, recorded between an interviewer and an interviewee, discussing issues of local interest. It was completely transcribed orthographically and annotated at the phone and word level. While the corpus contains around 38 hours of recordings, coming from 40 speakers, for the challenge only a subset of the corpus was used. It was divided into two datasets: a sample set containing recordings from 2 speakers (one female, one male), totaling almost 2 hours, and an evaluation set more than 10 hours long, with data coming from 12 speakers (six females, six males).

A subset of the NCHLT Xitsonga Speech Corpus was used for the challenge. It contains more than 4 hours of recordings, coming from 24 speakers (12 females and 12 males). The corpus contains read speech, recorded through a smartphone-based interface and it was transcribed and annotated at the word and phone-level.

5.3.3 Evaluation metrics

Besides the two datasets, the above challenge offers on its website (www.zerospeech.com) also an evaluation software based on the measures introduced in [Ludusan, 2014b], toolkit which was used to evaluate the results obtained in this section. This allowed us to compare against baseline systems.

Several measures are implemented in the evaluation package, ranging from metrics on the quality of the matching process, to those characterizing the clustering stage and some which compute natural language processing metrics, like token and type F-scores. We focus here on the matching metrics, as they indicate the performance of the DTW search. We have chosen this measure because all the other measures are directly affected by the matching quality and we expect that a good first matching stage would also translate into better performance downstream.

Precision, recall and F-score are computed from the set of discovered motif pairs, with respect to all matching substrings in the dataset. Precision is defined as being the proportion of discovered substrings pairs that belong to the list of gold pairs, weighed by the type frequency. Similarly, recall is computed as the proportion of gold motif pairs discovered by the algorithm. Matching F-score is defined as the harmonic mean between precision and recall.

From a formal point of view, we define a set of found structures (X), which are compared to the set of gold structures (Y) using average precision, recall and F scores:

$$Precision = \sum_{t \in types(X)} \omega(t, X) \cdot \frac{match(t, X \cap Y)}{match(t, X)} \tag{5.4}$$

$$Recall = \sum_{t \in types(X)} \omega(t, X) \cdot \frac{match(t, X \cap Y)}{match(t, Y)} \tag{5.5}$$

$$F - score = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \tag{5.6}$$

In most of the cases, X and Y will be sets of fragments (i, j) or of pairs of such fragments. We will always sum over fragment types, as defined through their phonemic transcriptions T, with

a weight w defined as the normalized frequency of the types in the corpus. The function $match(t, X)$ counts how many tokens of type t are in the set X [Ludusan, 2014b].

5.3.4 Results and analysis

The results obtained are presented in terms of matching F-score, computed over all speakers in the respective datasets. Since the optimal value of the DTW threshold was set on the sample set, part of the English dataset, we are particularly interested in the performance obtained by the system on a different language. We expect that good results on another language, not seen by the system, will further validate the generalizability of the approach. We report results for the matching precision, recall and F-score and for all the features/combinations of features we tested. By doing so, we expect to have a better insight into the role that each feature plays in the term discovery process.

The matching F-score results on the two tested languages are illustrated in Figure 5.7. It shows the performance of our baseline (MFCC) on the first column and that of the systems using either posteriorgrams or phoneme vectors, computed with the different single language acoustic models or combinations of them, as input features.

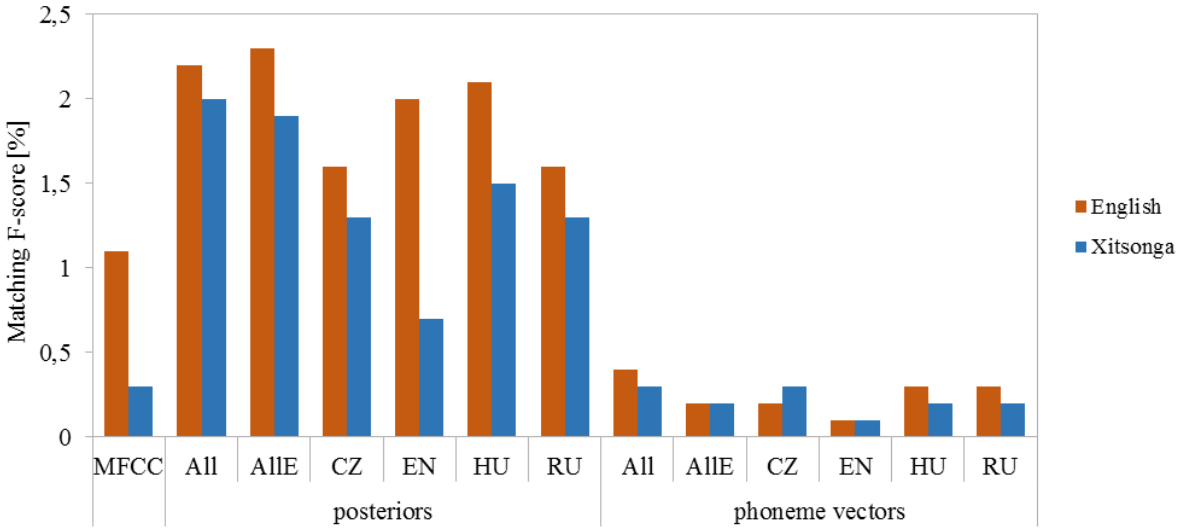

Figure 5.7 Matching F-score obtained using the posteriorgrams and the phoneme vectors features (individual and combination of languages), on the English and Xitsonga datasets

When comparing the performance of the different features, we can see a clear advantage of posteriorgrams over MFCCs and phoneme vectors, for both languages. Furthermore, we observe an important increase in performance also on the Xitsonga dataset, although the DTW threshold was set on a totally different language (English). Phoneme vectors instead seem not to have enough discriminative power for spoken term discovery. It shows that the hard decision taken by recognizer introduces a significant amount of error, from which the system cannot recover even when multi-language resources are employed.

Regardless of the feature used (posteriorgrams or phoneme vectors), we can see the advantage of using combined features. These features give either the best metric values or they are close to the best one, being the most consistent ones, overall.

Next, we looked more in detail into the systems employing posteriorgrams as input features. Table 5.2 shows the precision, recall and F-score for each individual feature setting, on the two languages. It appears that the systems having in input posteriorgrams of combinations of languages behave better both in terms of precision and recall. Again, for both languages, we obtain either the best performance or close to it, in the case of multi-language posteriorgrams.

Table 5.2 Matching Precision, Recall and F-score obtained on English and Xitsonga when MFCCs (baseline) and Posteriorgrams are used as input features (bold represents the best overall result)

System	English			Xitsonga		
	P [%]	R [%]	F [%]	P [%]	R [%]	F [%]
MFCC	1.2	1.1	1.1	6.4	0.2	0.3
CZ	4.5	1.0	1.6	8.6	0.7	1.3
EN	6.1	1.2	2.0	5.7	0.4	0.7
HU	6.3	1.3	2.1	10.4	0.8	1.5
RU	3.6	1.1	1.6	6.9	0.7	1.3
AllE	**7.2**	**1.4**	**2.3**	**12.5**	1.0	1.9
All	4.5	1.5	2.2	8.7	**1.1**	**2.0**

Results presented in this section were published in [8], and the thesis author contributed with the feature extraction system, the embedded VAD module and advice on calibrating the system threshold for the DTW metric.

5.4 UNSUPERVISED SPOKEN TERM DETECTION EXPERIMENTS (QBYE STD)

The audio search problems are addressed by Spoken Term Detection approaches, which identify all of the occurrences of a specified "term" in a given corpus of speech data. For the detection task, a term is considered a sequence of one or more words, and no terms will include more than five words (Patty, 2006). A particular feature that discriminates Spoken Term Detection from other ASR-based tasks, such as speech transcription or keyword spotting, is that queries may contain words that are not limited to the system vocabulary. So Spoken Term Detection systems must cope with these so-called out-of-vocabulary (OOV) words.

Traditionally, most systems used large vocabulary continuous speech recognition tools to produce word transcripts [Mamou, 2007]. These transcripts are further indexed and query terms are retrieved from the index. Most of the time, query terms that are not part of the recognizer's trained vocabulary cannot be retrieved, decreasing the evaluation recall, with a significant drawback that such approaches return no results on queries containing out-of-vocabulary terms [Mamou, 2007]. Thus, more advanced systems provide also phonetic transcripts, against which query terms can be matched phonetically. Such systems suffer from lower accuracy, but are a first step towards a language independent method of search. Some of the more advanced systems match phones from multiple language resources to improve the search. Current approaches to spoken term detection rely on variants of dynamic time warping (DTW) algorithm, to efficiently perform a search within a given speech corpus and detect the location of all query occurrences or terms [Park, 2008; Jansen, 2010; Flamary, 2011; Muscariello, 2012]. Despite these different approaches, all spoken term discovery systems can be logically broken down and implemented in two phases: indexing and searching [Patty, 2006]. In the indexing phase, the system must process the speech data without knowledge of the terms. The extraction of reliable features plays a very important role in this phase, for speech representation. In the searching phase, the system uses the terms and the speech parameterization (features), the index, and optionally the audio to detect term occurrences and their location. Theoretically, a perfect system, with the best methods, would detect the exact locations of all the query occurrences in the audio documents, and would yield no false detections.

In practice, however, acoustic, language and phonetic models are not available for all languages. Such languages are called under-resourced. In this case, it is not possible to process a query in a text form, because, due to the lack of phonetic models, mapping between the

pronunciation and the written form of the words are not available. Therefore, Spoken Term Detection is approached by query-by-example search. This means that queries are given in the form of recording of spoken terms and the task becomes searching for audio queries in audio contents.

In this section, I attempt to solve the Spoken Term Detection problem for under-resourced languages by phone recognition with a multilingual acoustic model, having special focus on low-resource languages. The Power Normalized Cepstral Coefficients (PNCC) features are used for improved robustness to noise. I investigate whether the use of multi-language resources as input features help the process of term detection. The proposed multilingual acoustic model (AM) is trained, at first, with three languages (Albanian, English and Romanian), then introduce additional languages (Czech, Hungarian, Russian) and features. Then I evaluate our system on the ground truth and evaluation metrics proposed by the MediaEval 2014 Multimedia Benchmark Initiative (MediaEval, 2014). The 2014 database features speech audio in many unknown languages, most of them under resourced. But the acoustic environment is clean in the 2014 database, so our proposed method is also tested against the 2015 database and ground truth metrics, which features very challenging acoustic environments (random noise, reverberation, low volume, etc).

5.4.1 Experimental Setup

The proposed system uses a multilingual acoustic model with a scalable Dynamic Time Warping (DTW) search algorithm. To solve the QbyE STD problem for under-resourced languages, we use an indexer with a multilingual phoneme recognizer with acoustic models from multiple languages. I also experimented with multiple features types, to see which performs better and in what conditions for this task.

The final system implementation is based on the architecture proposed by NIST, illustrated in Figure 5.8. We separate the indexing and the searching modules, rather than searching the corpus directly for each query term, to make the search faster, provided that the indexing method simplifies the search [Buzo, 2013]. So the approach consists of two stages:

- The indexing, i.e. the phone recognition of the content data
- The searching, i.e. finding a similar string of phones in the indexed content that matches the one of the query by using a DTW based searching algorithm.

Phone recognition is used for indexing, thus all the speech contents are transformed in strings of phonemes. As stated in the introduction section, for under-resourced languages where resources are scarce, phone recognition makes an ideal choice, as we do not have enough data to construct complex *n*-gram language models.

For the 2014 database experiments, we started with two speech feature types, to parametrically represent speech: the common Mel Frequency Cepstral Coefficients (MFCC) and the Power Normalized Cepstral Coefficients (PNCC), the later theoretically offering improved robustness to noise. Both type of features are explained in detail in Chapter 2, here we reiterate their most important characteristics and why they are used in speech.

The MFCC features are widely used and well known, we used them as baseline features. MFCC`s are implemented in the Sphinx Toolkit used for development of the system. They are based on the known variation of the human ear's critical bandwidths with frequency. Filters spaced linearly at low frequencies and logarithmically at high frequencies have been used to capture the phonetically important characteristics of speech.

The second set of features, Power Normalized Cepstral Coefficients (PNCC), are relatively new and their development was motivated by a desire to obtain a set of features for speech recognition that are more robust to acoustical variability, hence they perform better in noisy

environments. Their computational complexity is comparable to that of MFCC coefficients. Major new features of PNCC processing include the use of a power-law non-linearity that replaces the traditional log non-linearity used in MFCC coefficients, a noise-suppression algorithm based on asymmetric filtering that suppress background excitation, and a module that accomplishes temporal masking [Kim, 2012]. Experimental results demonstrate that PNCC processing provides substantial improvements in recognition accuracy compared to MFCC and PLP processing for speech, in the presence of various types of additive noise and in reverberant environments, with only slightly greater computational cost than conventional MFCC processing, and without degrading the recognition accuracy that is observed while training and testing using clean speech [Kim, 2012].

Regarding the Acoustic Model (AM), in our first approach using HMM based phone recognizers, we wanted to compare the effect of using multilingual resources against monolingual models, and in order to achieve this, we built six acoustic models described in Table 5.3. We start with acoustic models for each language and use the IPA classification for mapping common phones in language models (LM) trained with all data. Mapping common phones is motivated by the high number of phones obtained in AM4, where phonemes from different languages are trained separately and they are seen as different entities. This allowed us to lower the number of phonemes to 98 for the multilingual model AM5. We used a moderate number of training data for individual languages, to have a balanced training data set among different languages. For comparison, we trained and additional acoustic model for Romanian (AM6), with a big amount of data (64h) and a relatively small set of phonemes (34).

Table 5.3 Training data used for HMM approach for QbyE STD task

ID	Language	LM no. phonemes	Training data [h]
AM1	Romanian	34	8.7
AM2	Albanian	36	4.1
AM3	English	75	3.9
AM4	Multilingual separate phones	145	16.7
AM5	Multilingual common phones (IPA)	98	16.7
AM6	RomanianBig	34	64

Unlike previous years, in 2015 the audio database features a more challenging acoustic environments, by introducing noise and reverberation. We expected PNCC features to perform better in this scenario.

As increasing the training database in comparison with last year would go beyond the context of this type of tasks (which aims at low-resourced languages), we tried introducing new languages in the training phase to see how they perform, along with a neural network based phoneme recognizer, from BUT, used for motif discovery experiments in previous section, and detailed in Chapter 2. This phone recognizer uses a split temporal context (STC) based feature extraction, with neural network classifiers to output phone posteriorgrams, while Viterbi algorithm is used for phoneme string decoding. We can use the output of this tool in our DTW search algorithm as input features, to do the matching.

In order to use additional languages to build features for the phoneme recognizer, we used the pre-trained systems available at [BUT, 2015] and described in Table 5.4.

Table 5.4 Trained systems used for STC approach for QbyE STD task

ID	Language	LM no. phonemes	WER[%]
AM7	Czech	45	24.24
AM8	Hungarian	61	33.32
AM9	Russian	52	39.27

The languages used for training the systems described in Table 5.4 are from the SpeechDat-E Eastern European Speech Database [Speechdat-E, 2015]. Another incentive to use these systems is the existence of trained non-speech events mapped to the following tokens, which should prove useful with these years challenging acoustic environment:

- "int" for intermittent noise
- "spk" for speaker noise
- "pau" for silent pause

The STC approach, described in detail in Chapter 2, is based on the theoretical study that significant information about phoneme is spread over few hundreds milliseconds and that an STC system can process two parts of the phoneme independently. The trajectory representing a phoneme feature can then be decorrelated by splitting them into two parts, to limit the size of the model, in particular the number of weights in the neural-net (NN). The system uses two blocks of features, for left and right contexts (the blocks have one frame overlap). Before splitting, the speech signal is filtered by applying the Hamming window on the whole block, so that the original central frame is emphasized. Dimensions of vectors are then reduced by DCT and results are sent to two neural networks. The posteriors from both contexts are, in the final stage, merged, after the front-end neural networks are able to generate a three-state per phoneme posterior model [10]. The above described features were used in this work as input to our search algorithm, which is described in the following paragraph.

The proposed search method uses a Dynamic Time Warping Algorithm (DTW) to align a string (a query) within a given content. Originally, DTW has been used to compare different speech patterns in automatic speech recognition, as stated in the introduction. In fields such as data mining and information retrieval, DTW has been successfully applied to automatically cope with time deformations and different speeds associated with time-dependent data.

The search is not performed on the entire content, but only on a part of it by the means of a sliding window proportional to the length of the query, where both query and contents are string of phonemes. The term is considered detected if the DTW scores above a threshold. In addition to the classical DTW string algorithm, we include, in the distance formula, the effect of query length and DTW match spread. Their effect is weighted in order to find an optimal configuration. The accuracy of the ASR used for indexing plays an important role, because the searching algorithm must compensate for the rather high Phone Error Rate (PhER), thus it must be robust.

Since the length of the content is usually greater than the length of the query, the comparison is made within a sliding window whose length is proportional to the query length. For each window, the alignment is given by the score *s*:

$$s = (1 - PhER) \tag{5.7}$$

where *s* is a score of similitude. Detection is based on a threshold which is determined empirically.

It is also worth mentioning that the evaluation toolkits used for this experiments also look for different types of searches:

- Type 1 search, exact match. Occurrences of single/multiple word queries in utterances should exactly match the lexical representation of the query. An example of this case is the query "white horse" that should match the utterance "My white horse is beautiful" but should not match to "The whiter horse is faster".
- Type 2 search, re-ordering and small lexical variations.
- Type 3 search, conversational queries in context. This type of search is another step towards realistic use-case scenarios. The spoken query not only contains relevant terms, but also useless (filler) items.

But standard DTW score is not good at scoring the above type of searches, as it cannot catch variations in query types, and might penalize better, more compact, string alignment. This is better explained in Table 5.5.

Table 5.5 Standard DTW scoring issues

	DTW search	Score
Sliding window content	… v a m e s u l p l e …	0.6
Query 1	m e s u ' '	
Sliding window content	… v a m e s u l p l e …	0.6
Query 2	m ' s ' l p	
Sliding window content	… v a m e s u l p l e …	0.6
Query 3	m e ' '	
Sliding window content	… v a m e s u l p l e …	0.6
Query 4	v a m e s u l ' ' '	

The detection method is refined by introducing a penalization for the short queries and the spread of the DTW match. Penalizations are motivated by the assumption that for two queries of different length that match their respective contents by the same phone error rate (PhER), the match of the longer query is more probable to be the right one (Query 4 example in Table 5.5). The formula for the score s is now given by equation [Buzo, 2013]:

$$s = (1 - PhER)(1 + \alpha \frac{L_Q - L_{Qm}}{L_{QM} - L_{Qm}})(1 + \beta \frac{L_w - L_S}{L_Q}) \tag{5.8}$$

where L_Q is the length of the query, L_{QM} and L_{Qm} are the maximum and minimum values respectively for L_Q, L_W is the window length, L_S is the spread of the DTW match, while α and β are tuning parameters that control the amount of penalization.

The penalizations in Equation 5.8 are motivated by the assumption that for two queries of different length that match their respective contents by the same phone error rate (*PhER*), the match of the longer query is more probable to be the right one. Similarly the more compact DTW matches are assumed to be more probable than the longer ones. This algorithm is suitable for queries searches of type 1 and 2, because the DTW handles inherently the small variations from the query, but it is not suitable for queries of type 3 where word order may be inverted.

The above methodology is summarized in Figure 5.8, the proposed system architecture.

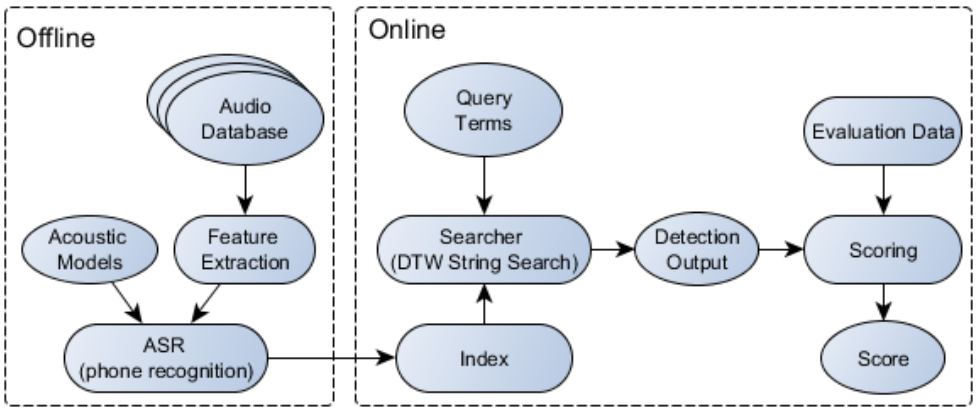

Figure 5.8 The proposed QbyE STD system architecture

5.4.2 Datasets

The MediaEval QUESST 2014 search dataset consists of 23 hours or around 12.500 spoken documents in the following languages: Albanian, Basque, Czech, non-native English, Romanian and Slovak. The languages were chosen so that relatively little annotated data can be found for them, as would be the case for a "low resource" language. The recordings were PCM encoded with 8 KHz sampling rate and 16 bit resolution (down-sampling or re-encoding were done when necessary to homogenize the database). The spoken documents (6.6 seconds long on average) were extracted from longer recordings of different types: read, broadcast, lecture and conversational speech. Besides language and speech type variability, the search dataset also features acoustic environment and channel variability.

The 2015 QUESST database is even more difficult, as speech is recorded in challenging acoustic conditions, and the data was artificially noised and reverberated. In total there were about 19 hours of audio, 11662 spoken documents in the following languages: Albanian, Chinese, Czech, Portuguese, Romanian, and Slovak. There were 450 queries recorder in isolation by different speakers, some non-native of the language.

This databases are free for research purposes, and together with the evaluation toolkit, can be used for a good comparison of different approaches to the spoken term detection task.

5.4.3 Evaluation metrics

The development and evaluations datasets used are part of the 2014 and 2015 QUESST task from Mediaeval [MediaEval, 2015] evaluation campaigns. This allowed us to use metrics tested in relevant workshops in the field. Two separate sets of queries are provided, for development and evaluation, along with a single set of audio files. The set of development queries and the set of audio files are distributed including the ground truth and the scoring scripts. More on the datasets used in the following section.

The metrics used to evaluate QbyE STD performance obtained with different acoustic models on the development data set, are made using the Maximum Term Weighted Value (MTWV), along with Detection Error Tradeoff (DET) curves. TWV is defined as a weighted combination of the miss and false alarm error rates, averaged over the set of queries, as follows (Fiscus, 2007):

$$TWV(\theta) = 1 - \frac{1}{|Q|}\sum_{\forall q \in Q}(P_{miss}(q,\theta) + \beta \cdot P_{fa}(q,\theta)) = 1 - (P_{miss}(\theta) + \beta \cdot P_{fa}(\theta)) \tag{5.9}$$

where the weight factor $\beta > 0$ is defined as:

$$\beta = \frac{C_{fa} \cdot (1 - P_{target})}{C_{miss} \cdot P_{target}} \tag{5.10}$$

and $-\beta < \mathrm{TWV}(\theta) < 1$ with 1 for a perfect system. C_{miss}, $C_{fa} > 0$ are the costs of miss and false alarms, and $0 < P_{target} < 1$ is the prior probability of a target trial, assumed to be constant across queries. ATWV is also the reference metric in NIST Spoken Term Detection evaluations. In the STD 2006 evaluation campaign they used $\mathrm{TWV}(\theta_{act})$ for system hard decisions, known as Actual Term-Weighted Value. As usual, the Maximum Term Weighted Value (MTWV) is the highest value that can be attained by applying a single threshold to system scores.

We use a graphical performance assessment using a Detection Error Tradeoff (DET) curve that plots miss probability (pMiss) versus false alarm probability (pFA). Miss and false alarm probabilities are functions of the detection threshold, θ. This (θ) is applied to the system's

detection scores, which are computed separately for each search term, then averaged to generate a DET line trace [Fiscus, 2007].

The results obtained on the development database with different speech features use a secondary metric, the normalized cross entropy cost (C_{nxe}).

This metric has been used for several years in the language and speaker recognition fields to calibrate system scores, and correlate quite well with TWV metrics. C_{nxe} is based on system scores, in contrast to TWV, which evaluates system decisions. C_{nxe} measures the fraction of information, with regard to the ground truth, that is not provided by system scores, assuming that they can be interpreted as log-likelihood ratios. A perfect system would get $C_{nxe} \approx 0$ and a non-informative system would get $C_{nxe} = 1$ [Rodriguez-Fuentes, 2013]. If we assume that the system under evaluation, *S*, submits a set of log-likelihood ratios llr_t for a set of trials *T(S)*, with the prior probability P_{tar}, then the empirical cross entropy, in information bits, is:

$$C_{xe} = \frac{1}{\log 2} \cdot \left(\frac{P_{tar}}{\left|T_{true}(S)\right|} \sum_{t \in T_{true}(S)} C_{\log}(llr_t) + \frac{1 - P_{tar}}{\left|T_{false}(S)\right|} \sum_{t \in T_{false}(S)} C_{\log}(llr_t) \right) \tag{5.11}$$

where the logarithmic cost function is:

$$C_{\log}(llr_t) = \begin{cases} -\log(sigmoid(llr_t + \log it(P_{tar}))) & t \in T_{true}(S) \\ -\log(sigmoid(-(llr_t + \log it(P_{tar})))) & t \in T_{false}(S) \end{cases} \tag{5.12}$$

The empirical cross entropy of a system can be normalized ($llr_t = 0 \ \forall t$) to obtain a trivial system, and we obtain the prior entropy, system that always gives non-informative scores:

$$C_{xe}^{prior} = \frac{1}{\log 2} \cdot \left(P_{tar} \cdot \log \frac{1}{P_{tar}} + (1 - P_{tar}) \cdot \log \frac{1}{1 - P_{tar}} \right) \tag{5.13}$$

Finally, the normalized empirical cross entropy is defined as (Rodriguez-Fuentes, 2013):

$$C_{nxe} = \frac{C_{xe}}{C_{xe}^{prior}} \tag{5.14}$$

5.4.4 Results and analysis

In Figure 5.9, a comparison of the results obtained with different acoustic models on the development data set are shown. We use a graphical performance assessment using a Detection Error Tradeoff (DET) curve that plots miss probability (pMiss) versus false alarm probability (pFA).

We can see that the Romanian acoustic model obtained the best results among individual languages, probably because it was trained with double the amount of data. AM4 multilingual performed slightly better than the monolingual acoustic models. The number of phonemes for this acoustic model is relatively high by combining data from all languages, thus it increases the uncertainty during recognition. Improvement is shown in multilingual model AM5, where using the IPA classification to merge phones helped and results show an improvement among the evaluation metrics. For comparison, the acoustic model AM6, trained with a bit amount of data, obtained the best results, even though it is trained with only one language (Romanian). It seems more data is needed to consolidate multilingual acoustic models, otherwise the larger phoneme set, even with merging, increase the detection uncertainty.

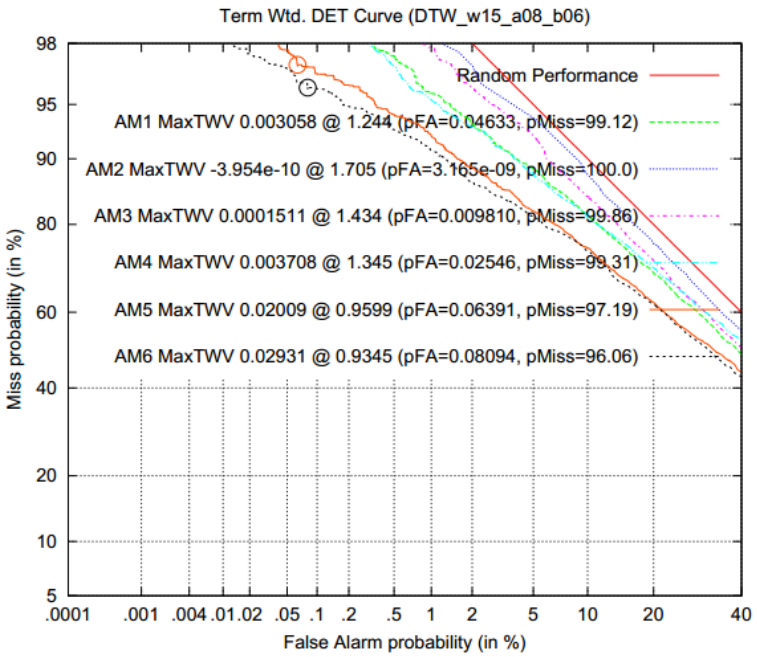

Figure 5.9 QbyE STD results for the 2014 database

Results obtained on the development database with different speech features (PNCC and MFCC) are shown in Table 5.6.

Table 5.6 PNCC and MFCC performance comparison using actual and minimum C_{nxe} for 2014 dataset

ID	PNCC		MFCC	
	AC_{nxe}	$MinC_{nxe}$	AC_{nxe}	$MinC_{nxe}$
AM1	1.032	0.986	1.032	0.986
AM2	1.055	0.997	1.055	0.997
AM3	1.03	0.994	1.03	0.994
AM4	1.015	0.972	1.016	0.971
AM5	1.016	0.969	1.016	0.969
AM6	1.032	0.986	1.032	0.986

Results show almost no difference between the two types of features. The same conclusion is drawn even when comparing by TWV metric. In general speech recognition, PNCCs obtain better accuracy in noise conditions, but, most probably, the noise in the MediaEval 2014 database is not significant. Therefore, the use of PNCC did not bring any improvement in these scenario.

For the 2015 evaluation, where the audio database is very challenging, results are presented in Figure 5.10, Table 5.7 for MFCC vs PNNC and Table 5.8 for Posteriorgram models, as this types of features should be more content aware and theoretically provide better performance.

Table 5.7 PNCC and MFCC performance comparison using actual and minimum C_{nxe} for 2015 dataset

ID	MFCC		PNCC	
	AC_{nxe}	$MinC_{nxe}$	AC_{nxe}	$MinC_{nxe}$
AM1	1.0061	0.9944	1.0061	0.9943
AM2	1.0059	0.9947	1.0058	0.9947
AM3	1.0055	0.9944	1.0047	0.9933
AM4	1.0047	0.9933	1.0047	0.9933
AM5	1.0037	0.9923	1.0037	0.9923
AM6	1.0057	0.9935	1.0057	0.9935

Table 5.8 Posteriorgram performance comparison using actual and minimum C_{nxe} for 2015 dataset

ID	AC_{nxe}	$MinC_{nxe}$
AM7	1.0055	0.9945
AM8	1.0048	0.9935
AM9	1.0056	0.9941

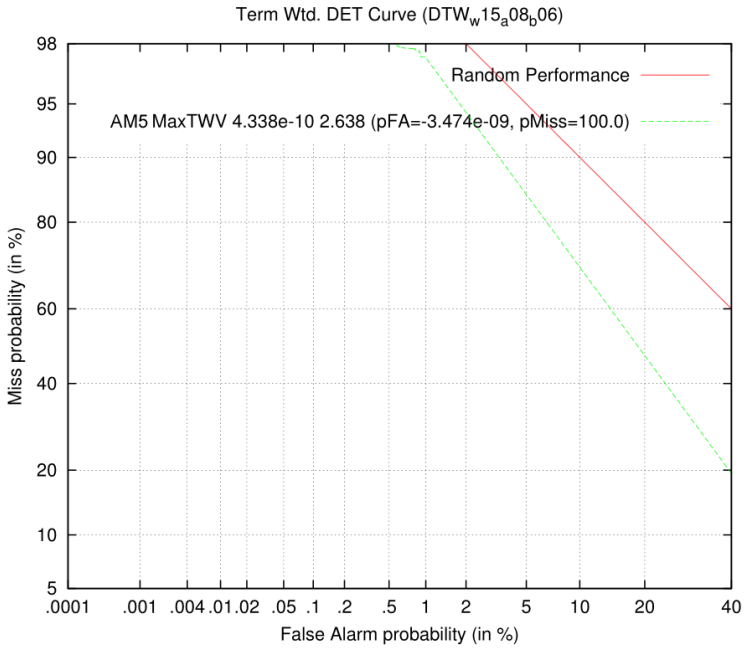

Figure 5.10 QbyE STD results for the 2015 database

Figure 5.10 plots only AM5 model, as the rest of the model are of the scale, in terms of TWV values, thus produce random results. Posterior models (AM8/AM9) seem to offer minimal performance improvements in some scenarios, so we cannot draw a conclusion that posteriors are better suited for the STD task in difficult acoustical environments. The same can be said about PNCC features, that did not offer performance improvements, probably because acoustical models are not trained on noisy databases, and the phoneme recognizer cannot recover relevant phone classes.

By applying the embedded VAD (we cut on frames from queries and audio content that had a high probability of corresponding to silence or noise tokens) used in motif discovery experiments, we can further limit the search interval and improve the metrics, as shown in Table 5.9 (using posterior models).

Table 5.9 Posteriorgram performance comparison using actual and minimum C_{nxe} and embedded phoneme VAD for 2015 dataset

ID	AC_{nxe}	$MinC_{nxe}$
AM7	1.0015	0.9823
AM8	1.0023	0.9843
AM9	1.0025	0.9816

5.5 CHAPTER CONCLUSIONS

Running our proposed architectures for both discovery and detection task on relevant challenges or workshops in the field, allowed the thesis author to compare the performance of different systems within and across relevant studies in the field.

We have first presented an investigation into the use of multi-language resources for the task of spoken term discovery, by evaluating against the ZeroSpeech challenge dataset. Similarly to previous studies employing posteriorgrams as input features for term discovery, we have shown that they improve performance with respect to using MFCCs. We have also explored the use of combined features, by concatenating posteriorgrams coming from different languages and we observed that they generally improve over the single language features. Since individual language features might give good results for one metric and worse for other metrics, one can use the combination of posteriorgrams from different languages for more robust overall results, a desirable trait for a system working on an unknown language. Also, exploring the use of phoneme identity information (binary vectors) in spoken term discovery, showed that, contrary to its usefulness in spoken term detection, this type of information is not sufficient for the current task. Still, the use of combined features seems to give also in this case an overall better performance over single language features.

Regarding the spoken term detection, we evaluated the proposed methods against the MediaEval QUESST Task (QbyE STD), with a two-step process. A multilingual ASR is used as a phone recognizer for indexing the database, while a DTW based algorithm is used for searching a given query in the content database. We tested three types of features (MFCC, PNCC and Posteriorgrams) with two approaches to the phoneme recognizer (statistical HMMs and a neural STC approach). The results show no big improvement between each approach and feature types, in part because the used audio database features realistical audio scenarios, with very challenging acoustic environments, and the proposed phonetizers return a lot of “noise” tokens or repeated phonemes, which reflected further upon our DTW algorithm. As this information retrieval tasks are designed to get as close as possible to a practical use case scenario, in which a user would like to retrieve, using speech, utterances in any language and conditions, much work is to be done in this direction, to be able to achieve relevant systems that can be used in production applications.

CHAPTER 6

CONCLUSIONS

6.1 GENERAL CONCLUSIONS

The main objectives of this thesis are in the field of automatic speech recognition, mainly to bring optimizations in the field of spoken language recognition. Throughout research, the thesis author identified several directions where research was needed, and the main target can be split into several tasks: ASR tuning, post-processing frameworks, unsupervised audio search and discovery, for information retrieval in speech systems. This thesis presents the successive steps which were employed by the author in order to approach each task.

As mentioned in the thesis introduction, current state-of-the-art paradigm for continuous speech recognition is the hidden Markov model (HMM), in particular, the HMM-based acoustic model used in conjunction with an *n*-gram model. The commercial availability of speech recognition, and the need for web-based language techniques have provided an important incentive for development of real systems and applications. The availability of very large on-line corpora has enabled statistical models of language at every level, from phonetics to discourse.

Therefore, *Chapter 1* of this thesis started with an introduction into the field of speech recognition and machine learning, it summarized the main tasks needed for obtaining an ASR system and then highlighted some of the components and methods for post-processing optimizations. In the context of merging historically distinct fields (speech recognition, computational linguistics, natural language processing), this chapter offered a brief overview of the current issues and directions going forward, such as spoken language recognition and

unsupervised learning. The study continued in *Chapter 2*, when special attention is given to the current speech features extraction and analysis methods, as current state-of-the-art ASR techniques do not use directly the time-domain waveform to model the speech signal.

Besides statistical approaches, we saw a renewed interest in Neural Networks, to ingest lots of data into training the ASR systems, and then feeding new data to those systems, with the incentive of receiving better predictions in response. Artificial neural networks are mathematical models of the low-level circuits in the human brain, and have been a familiar concept since the 1950s. The notion of using ANNs to improve speech-recognition performance has been around since the 1980s, and a model known as the hybrid ANN-Hidden Markov Model (ANN-HMM) showed promise for large-vocabulary speech recognition. They were not widely used until recently, because of performance issues. With the invention of discriminative training, which refines the model and improves accuracy, the conventional, context-dependent Gaussian mixture model HMMs outperformed ANN models when it came to large-vocabulary speech recognition. But with the ever increasing of the available computing power and available training data, scientists found novel ways to parallelize training in neural networks, for speaker-independent speech recognition, and improve performance.

Split temporal context model is such a proposed neural net system, addressed in *Chapter 3*, to obtain a robust phoneme recognizer to be used in unsupervised search experiments. In this proposed architecture, the trajectory representing a phoneme feature can then be decorrelated by splitting it into two parts, to limit the size of the model, in particular the number of weights in the neural-net. The system uses two blocks of features, for left and right contexts (the blocks have one frame overlap), that are in the in the final stage, merged. *Chapter 3* also summarized the integration of the components and toolkits necessary to build a continuous recognition system. Information for improving the models and the training set, along with decreasing word error rate (the primary evaluation metric) and sentence error rate were provided. Knowledge from this chapter was essential in tuning the phoneme recognizers, for the unsupervised speech processing techniques presented later in the thesis.

Lately, by using deep neural networks and using massive amounts of data, Google announced that it lowered the recognition error rate to just 8% [Venturebeat, 2015]. This pushed the boundaries of just “plain recognition”, and along with the commercial success of speech enabled devices (phones, cars, etc.), just simple recognition is no longer sufficient for a device to be “smart”. For an ASR system to go beyond just a plain raw recognizer, one must enhance the capabilities of an automatic speech recognition system by including post-processing frameworks to analyze the output, restore diacritics, punctuation, in order to increase readability of raw text, the standard output of a recognizer.

In the above context, *Chapter 4* offered a quick overview over post-processing means of increasing the output intelligibility of an ASR system. Moreover, the thesis author presented a novel restoration approach to punctuation and capitalization for text in Romanian language, using just text-based tri-gram language models. Overall, our tests show a precision of 76-80% for capitalization restoration, 54-60% for comma and 64-68% for period recovery. This margin can be reduced by using a larger training corpus in order to obtain better performing language models. To the best of my knowledge, these are the first re-capitalization and punctuation restoration results reported for Romanian language and at the time this thesis was written, we could not find a relevant study for capitalization and punctuation restoration for Romanian.

Besides the above objectives, another task attracted my interest: audio document retrieval and discovery. With the ever-increasing amounts of vast digital audio data being created and broadcasted daily from various sources, a pressing need exists for intelligent information extraction and retrieval methods, in the speech community. There are various applications for these methods, from document retrieval containing speech data like broadcast news, telephone

conversations and roundtable meetings to audio query searches. In recent years, numerous workshops hosted benchmarking initiatives to evaluate new algorithms for multimedia access and retrieval, such as MediaEval (MediaEval, 2011-2015), or as special sessions at relevant conferences in the field of speech communication (ZeroSpeech Challenge, InterSpeech 2015, OpenKWS). Also, this can have applications in languages with little or no resources, and has considerable relevance for cognitive modelling of human infant's language acquisition [Jansen et al., 2013].

These studies are presented in *Chapter 5*, for two similar subtask, Spoken Term Discovery and Spoken Term Detection. For the first subtask, the aim is to recover word boundaries (or motifs), as well as to construct a lexicon of terms. The second task aimed at low resource languages, to search for audio content (queries) within audio content (database), independent of the language at hand. As currently there is no standard evaluation metrics or databases for this types of tasks, the thesis author submitted its proposed systems to satellite workshops relevant in the field, and used their standard datasets and proposed evaluation metrics. As results show, there is much work to be done in this direction, as most of the participant teams struggle to propose systems that can be used in real life scenarios, either because of computational cost, or the relevance of their results. This is partly because these tasks have been designed to perform search on language-independent audio in a low-resource scenario, to get as close as possible to a practical use case. Moreover, audio files are stressed by noise, reverberation or channel mismatch. This conditions pose difficult recognition scenarios for largely trained and annotated ASR systems, let alone for DTW based systems proposed in these thesis or in literature. This chapter in the thesis was motivated by the challenge of searching and extracting useful information from speech data, in a completely unsupervised setting. In many real world speech processing problems, obtaining annotated data is not cost and time effective, so we asked ourselves how much can we learn from speech data without any transcription, or knowing the language at hand.

6.2 PERSONAL CONTRIBUTIONS

My personal contributions can be concentrated in Chapters 3, 4 and 5 of this thesis and are summarized as follows:

a) Research into the theoretical aspects and discussion of the main issues in preprocessing of the speech signal and it`s digital representation. Ways of parametrically representing speech with features are also presented, as speech features where extensively used in Chapter 5.
b) Overview over the current state of the art in speech recognition and natural language processing, in Chapter 3;
c) Research of the theory and processes behind automatic speech recognition, to build and design of a small vocabulary, automatic speech recognition system, presented in Chapter 3. Knowledge from this chapter was essential in tuning phoneme recognizers modules, for the unsupervised speech processing techniques presented in Chapter 5.
d) Analysis of ASR recognition results, providing information for improving the models and the training set, along with decreasing word error rate (the primary evaluation metric).
e) Research into techniques and theoretical aspects for automatic phoneme recognition from spoken speech, using neural network based approaches (TRAP, STC). These were presented in the second part of chapter 3, with the goal to extract as much information about phoneme from as long temporal context as possible, to be used in pattern recognition applications, evaluated in Chapter 5.

f) Overview some of the proposed methods in Romanian literature to restore diacritics and diarize speech, found in post-processing modules, to enhance intelligibility for a human reader using an ASR system.
g) Enhance the capabilities of an automatic speech recognition system by including a new module in the post-processing framework to statistically restore capitalization and punctuation of raw text resulted from the output of the system. The proposed module uses an n-gram based method for capitalization and punctuation restoration for Romanian language. To the best of my knowledge, these are the first re-capitalization and punctuation restoration results reported for Romanian language and at the time this thesis was written, as we could not find a relevant study for capitalization and punctuation restoration for Romanian.
h) Overview over information processing and retrieval, in the context of NLP.
i) Set up and integrate a robust phone recognizer toolkit to be used for pattern matching in audio content.
j) Set up a system and propose a methodology for Spoken Term Discovery and Detection in the context of low-resourced languages (use MODIS algorithm for spoken discovery, along with parameter tuning, enhance the proposed STD NIST architecture); Results with proposed techniques are evaluated and presented.

6.3 FUTURE WORK

The author of this thesis is interested in continuing the research directions started in these thesis, which were not concluded. Thus, regarding the ASR system, it can be enhanced by using more training data, for a better speaker discrimination.

Regarding the proposed punctuation and capitalization module, evaluation results suggest that test files from the same corpora domain offer better results, by a small margin. This margin can be reduced by using a larger training corpus in order to obtain better performing language models. Thus, further work will focus on improving the language models, extending the study on other punctuation marks and on including more complex models based on acoustic or prosodic features from the audio signal.

For the unsupervised speech processing tasks (Spoken Term Discovery and Detection), there is a lot of work to be done, as this is a relatively new field of study. First, for the discovery task, as future research directions we plan to extend the current study by investigating other language combinations that were not tested here. We also plan to explore the use of posteriorgrams coming from training an acoustic model with the phonemes of several languages, a sort of "universal" acoustic model.

The "universal" model can also be applied to the detection task, to allow better discrimination between phonemes from different languages. This universal model should incorporate noise into the training data, to reduce the mismatch problem between the training and the decoded data. Here, we could also improve the VAD module, to include a frequency band based energy VAD, as statistical VAD did not help much in noisy conditions. Finally, for our multilingual approach, some sort of data fusion technique must be also investigated, to merge relevant data from all the models. It remains to be seen if increasing the training database is relevant for the study of low resource conditions.

PUBLICATIONS LIST

1. **Caranica A.**, Burileanu C., "An automatic speech recognition system with speaker-independent identification support", *ATOM-N 2014, Proceedings of SPIE*, ISSN 0277-786X, Constanța, Romania, 2015
2. **Caranica A.**, Cucu H., Buzo A., Burileanu C., "On the design of an automatic speaker independent digits recognition system for Romanian language", *Journal of Control Engineering and Applied Informatics, ISSN 1454-8658*, 2015, accepted for publication.
3. **Caranica A.**, Cucu H., Buzo A., Burileanu C., "Exploring Spoken Term Detection with a Robust Multi-Language Phone Recognition System", Rev. TJFE, ISSN 0254-0770, 2015.
4. **Caranica A.**, Cucu H., Buzo A., Burileanu C., "Capitalization and Punctuation Restoration for Romanian Language", *UPB Scientific Bulletin, Series C, ISSN 2286 – 3540*, 2015
5. **Caranica A.**, Buzo A., Cucu H., Burileanu C., "SpeeD @ MediaEval 2015: Multilingual phone recognition approach to Query by example STD", *Working Notes Paper, MediaEval 2015*, Wurzen, Germany, 2015.
6. Cucu H., Buzo A., **Caranica A.**, Burileanu C., "On Formatting Transcriptions of Romanian Speech", *Rev. TJFE, ISSN 0254-0770*, 2015.
7. Cucu H., **Caranica A.**, Buzo A., Burileanu C., "On Transcribing Informally-Pronounced Numbers in Romanian Speech", *TSP 2015*, Prague, Czech, 2015.
8. Ludusan B., **Caranica A.**, Buzo A., Burileanu C., Dupoux E., "Exploring multi-language resources for unsupervised spoken term discovery", SpeD 2015, ISBN: 978-1-4673-7559-7, Bucharest, Romania. 2015.

REFERENCES

[Acero, 1990] A. Acero and R. M. Stern, "Environmental Robustness in Automatic Speech Recognition," in Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing (Albuquerque, NM), vol. 2, Apr. 1990, pp. 849–852.

[Adami, 2010] A. G. Adami, "Automatic Speech Recognition: From the Beginning to the Portuguese Language", Universidade de Caxias do Sul, Centro de Computação e Tecnologia da Informação, 2010.

[Anguera, 2012] Xavier Anguera, "SPEAKER INDEPENDENT DISCRIMINANT FEATURE EXTRACTION FOR ACOUSTIC PATTERN-MATCHING", Telefonica Research, 2012.

[Baldwin, 2009] T. Baldwin, M. P. A. K. Joseph, "Restoring Punctuation and Casing in English Text", Springer (LNCS), 2009.

[Baker, 1975] Baker, J., "The DRAGON system – an overview," IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 23, no. 1, pp. 24-29, 1975.

[Baker, 1989] Baker, J., "DragonDictate – 30K: Natural language speech recognition with 30,000 words," *Eurospeech 1989*, pp. 161-163, 1989.

[Barras, 2006] C. Barras, X. Zhu, S. Meignier, and J.L. Gauvain, "Multi-stage speaker diarization of broadcast news," IEEE Transactions on Audio, Speech and Language Processing, vol. 14, no. 5, pp. 1505–1512, 2006.

[Batista, 2008] F. Batista, D. Caseiro, N. Mamede, I. Trancoso, "Recovering Capitalization and Punctuation Marks for Automatic Speech Recognition: Case Study for Portuguese Broadcast News, Speech", in Speech Communication, 2008.

[Batista, 2012] F. Batista, H. Moniz, I. Trancoso, N. Mamede, "Bilingual experiments on automatic recovery of capitalization and punctuation of automatic speech transcripts", in Audio, Speech, and Language Processing, IEEE, 2012

[Beeferman, 1998] D. Beeferman, A. Berger, and J. Lafferty. "Cyberpunc: A lightweight punctuation annotation system for speech", in Proceedings of 1998 IEEE ICASSP'98, Seattle, USA, 1998.

[Beigi, 2011] H. Beigi, "Fundamentals of Speaker Recognition", 978-0-387-77592-0, 2011, Springer Science + Business Media, LLC, USA.

[Bernacki, 2005] M. Bernacki, P. Wlodarczyk, "Principles of training multi-layer neural network using backpropagation", Academic Website, http://galaxy.agh.edu.pl/~vlsi/AI/backp_t_en/backprop.html, accesed august 2015.

[Boldea, 2003] M. Boldea, "Contribuții la recunoașterea automată a vorbirii continue în limba română", Teză de doctorat, Universitatea "Politehnică" din Timișoara, 2003.

[Bourlard, 1994] Bourlard H., Morgan N., "Connectionist speech recognition. A Hybrid Approach", *Kluwer Academic Publishers*, 1994.

[Burget, 2006] Burget L., et al., "Indexing and search methods for spoken documents," in ICTSD, (2006).

[Burileanu, 1999] Burileanu, D., "Contribuţii la sinteza vorbirii din text pentru limba română", PhD Thesis, Bucharest, Romania, 1999.

[BUT, 2015] BUT Phoneme recognizer based on long temporal context, Brno University of Technology, http://speech.fit.vutbr.cz/software/phoneme-recognizer-based-long-temporal-context, accessed august 2015.

[Buzo, 2011] A. Buzo, "Recunoaşterea Limbajului Vorbit în Reţele de Telecomunicaţii Mobile", PhD Thesis, Bucharest, Romania, 2011.

[Buzo, 2013] Buzo A., Cucu H., Safta M., Burileanu C., "Multilingual Query by Example Spoken Term Detection for Under-Resourced Languages", in Speech Technology and Human - Computer Dialogue (SpeD), (2013).

[Buzo, 2014] A. Buzo, H. Cucu, L. Petrică, D. Burileanu and C. Burileanu, "An Automatic Speech Recognition Solution with Speaker Identification Support", *Proceedings of COMM*, pp. 119-122, 2014.

[Buzo, 2014b] A. Buzo, H. Cucu, C. Burileanu, "SpeeD @ MediaEval 2014: Spoken Term Detection with Robust Multilingual Phone Recognition", in Working Notes Proceedings of the MediaEval 2014 Workshop, Barcelona, Spain, 2014, ISSN: 1613-0073.

[Brants, 2003] T. Brants, "Natural Language Processing in Information Retrieval", Google Inc, 2003.

[Brown, 2002] E.W. Brown, A.R. Coden, "Capitalization recovery for text", IR Techniques for Speech Applications, 2002.

[Can, 2011] Can D., Saraçlar M., "Lattice Indexing for Spoken Term Detection", IEEE TRANSACTIONS ON AUDIO, SPEECH, AND LANGUAGE PROCESSING, VOL. 19, NO. 8, (2011).

[Catanese, 2013] L. Catanese, N. Souviraà-Labastie, B. Qu, S. Campion, G. Gravier, E. Vincent, and F. Bimbot, "MODIS: an audio motif discovery software," in Proc. of INTERSPEECH, 2013.

[Cawley, 1996; Kleijn, 1998] Cawley G. "The Application of Neural Networks to Phonetic Modelling". PhD. Thesis, University of Essex, England, 1996 (http://www.sys.uea.ac.uk/~gcc/thesis.html).

[Chelba, 2006] C. Chelba and A. Acero, "Adaptation of maximum entropy capitalizer: Little data can help a lot", Computer Speech & Language, 20(4):382–399, 2006

[Chen, 1999] Chen, S.F., Goodman, J.T.: An empirical study of smoothing techniques for language modeling. Computer Speech and Language 13 (1999) 359-393.

[Chen, 2000] Chen, S.F., Rosenfeld, R.: A survey of smoothing techniques for ME models. IEEE Transactions on Speech and Audio Processing 8 (2000) 37-50.

[Chen, 2005] B. Chen, Learning Discriminant Narrow-Band Temporal Patterns in Automatic Recognition of Conversational Telephone Speech, Ph.D. thesis, University of California, Berkeley, Berkeley, CA, USA, 2005.

[Chen, 2015] V. Chen, "Unsupervised Learning and Modeling of Knowledge and Intent for Spoken Dialogue Systems", *Ph.D. Thesis Proposal*, CMU, 2015.

[Christensen, 2001] H. Christensen, Y. Gotoh, S. Renals, "Punctuation annotation using statistical prosody models", in ISCA Workshop on Prosody in Speech Recognition and Understanding, 2001.

[Clark, 2010] A. Clark, C. Fox and S. Lapping, „The Handbook of Computational Linguistics and Natural Language Processing", Wiley-Blackwell, 2010.

[Cucu, 2011a] H. Cucu, "Towards a speaker-independent, large-vocabulary continuous speech recognition system for Romanian," *PhD Thesis, University Politehnica of Bucharest*, 2011.

[Cucu, 2011b] H. Cucu, L. Besacier, C. Burileanu, A. Buzo, "Enhancing Automatic Speech Recognition for Romanian by Using Machine Translated and Web-based Text Corpora," *in Proc. Int. Conf. Speech and Computer (SPECOM)*, Kazan, Russia, 2011, pp. 81-88.

[Cucu, 2014] H. Cucu, A. Buzo, L. Petrică, D. Burileanu and C. Burileanu, "Recent Improvements of the SpeeD Romanian LVCSR System," *in Proc. Int. Conf. Communications (COMM)*, Bucharest, Romania, 2014, pp. 111-114.

[Cucu, 2015] Cucu H., Buzo A., Caranica A., Burileanu C., "On Formatting Transcriptions of Romanian Speech", Rev. TJFE, ISSN 0254-0770, 2015.

[Dahl, 2012] Dahl G. E., Yu D., Deng L. and Acero A. (2012). "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition." IEEE Trans. Audio Speech Lang. Processing, vol. 20, no. 1, pp. 30–42, 2012.

[Denes, 1993] P. B. Denes and E. N. Pinson, "The speech chain." W. H. Freeman Company, 2nd Edition, 1993.

[Deleglise, 2009] P. Deleglise, Y. Esteve, S. Meignier, and T. Merlin, "Improvements to the LIUM French ASR system based on CMU Sphinx: what helps to significantly reduce the word error rate?" In Proc. Interspeech pp. 2123-2126, 2009.

[Deller, 2000] Deller J. R., Hansen J. H. L., Proakis J. G., "Discrete – time processing of speech signals", IEEE Press, 2000.

[Dredze, 2010] M. Dredze, A. Jansen, G. Coppersmith, and K. Church, "NLP on spoken documents without ASR," in Proc. of EMNLP, 2010, pp. 460–470.

[Domokos, 2009] Domokos J., "Contributii la recunoasterea vorbirii continue si la procesarea limbajului natural", PhD thesis, Universitatea Tehnica din Cluj Napoca, 2009.

[Dong, 2012] Dong W., Simon K., Joe F., Ravichander V., Nicholas E., and Raphaël T. “Direct posterior confidence for out-of-vocabulary spoken term detection”, ACM Trans. Inf. Syst. 30, 3, Article 16 (2012).

[Dumitru, 2008] Dumitru C. O., Gavat I. (2008). Progress in Speech Recognition for Romanian Language, Advances in Robotics, Automation and Control, pp. 472, Vienna, Austria

[Favre, 2008] B. Favre, R. Grishman, D. Hillard, H. Ji, D. Hakkani-Tur, M. Ostendorf, “Punctuating Speech for Information Extraction”, in ICASSP, 2008

[Fiscus, 2007] Fiscus J.G., Ajot J., Garofolo J.S., Doddingtion G., “Results of the 2006 spoken term detection evaluation,” in Proceedings of ACM SIGIR Workshop on Searching Spontaneous Conversational. Citeseer, pp. 51–55, (2007).

[Flamary, 2011] R. Flamary, X. Anguera, and N. Oliver, “Spoken WordCloud: Clustering recurrent patterns in speech,” in Proc. of International Workshop on Content-Based Multimedia Indexing, 2011, pp. 133–138.

[Furui, 1986] Furui, S., “Speaker-independent isolated word recognition using dynamic features of speech spectrum,” IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 34, no. 1, pp. 52-59.

[Furui, 2009] Furui, S. (2009). 40 Years of Progress in Automatic Speaker Recognition, Proc. of the ICB 2009 Conf, 1050-1059.

[Gauci, 2008] Gauci, O., Debono, C., Micallef, P.: “A maximum log-likelihood approach to voice activity detection.” In: International Symposium on Communications, Control and Signal Processing (ISCCSP-2008), pp. 383–387 (2008)

[Gales, 1993] M.J.F. Gales and S.J. Young. “HMM recognition in noise using parallel model combination.” Proceedings Eurospeech, pages 837–840, 1993.

[Gales, 1995] M.J.F. Gales. “A fast and flexible implementation of parallel model combination.” Proceedings ICASSP, pages 133–136, 1995.

[Garofolo, 1993] J. Garofolo, L. Lamel, W. Fisher, J. Fiscus, D. Pallett, N. Dahlgren, and V. Zue “TIMIT: acoustic-phonetic continuous speech corpus,” Philadelphia: Linguistic Data Consortium, 1993.

[Gravano, 2009] Gravano A., Jansche M., Bacchiani M. (2009). “Restoring punctuation and capitalization in transcribed speech”, in ICASSP 2009.

[Goodman, 2001] Goodman, J.T.: “A Bit of Progress in Language Modeling. Computer Speech and Language” (2001) 403-434.

[Harwath, 2013] D. Harwath, T. Hazen, and J. Glass, “Zero resource spoken audio corpus analysis,” in Proc. of IEEE ICASSP, 2013, pp. 8555–8559

[Henderson, 2010] J. B. Henderson, “Artificial Neural Network”, chapter 9, The Handbook of Computational Linguistics and Natural Language Processing, edited by Alexander Clark, Chris Fox, and Shalom Lappin, Wiley-Blackwell, 2010.

[Herley, 2006] C. Herley, “ARGOS: Automatically extracting repeating objects from multimedia streams,” IEEE Transactions on Multimedia, vol. 8, no. 1, pp. 115–129, 2006.

[Hermansky, 1990] Hermansky, H., “Perceptual Linear Predictive (PLP) Analysis of Speech,” Journal of the Acoustical Society of America, vol. 87, no. 4, pp. 1738-1752, 1990.

[Hermansky, 1994] H. Hermansky and N. Morgan. “RASTA processing of speech.” IEEE Transactions SAP, 2:578–579, 1994.

[Hermansky, 1999] H. Hermansky and S. Sharma, “Temporal patterns (traps) in asr of noisy speech,” in Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing (ICASSP), Phoenix, Arizona, USA, Mar 1999.

[Horzyk, 2013] A. Horzyk, “Artificial Neural Networks”, Academic website, AGH University of Science and Technology in Cracow, Poland, http://home.agh.edu.pl/~horzyk/lectures/ai/ann.php, accessed august 2015.

[Holmes, 1997] J. N. Holmes, W. J. Holmes and P. N. Garner, "USING FORMANT FREQUENCIES IN SPEECH RECOGNITION", British Crown Copyright, 1997.

[Hori, 2007] Hori T., Lee Hetherington I., Timothy J. Hazen, Glass J., “Open-vocabulary spoken utterance retrieval using confusion networks,” in ICASSP, (2007).

[HTK, 2015] HTK Speech Recognition Toolkit, http://htk.eng.cam.ac.uk/, accessed august 2015.

[Huang, 2001] Huang X., Acero A., Hon H.-W. “Spoken Language Processing. Guide to Algorithms and System Development”, Prentice Hall, 2001.

[Huang, 1993] Huang X., Alleva F., Hwang M., Rosenfeld R. (1993). An overview of the SPHINX-II speech recognition system. HLT '93 Proceedings of the workshop on Human Language Technology.

[Jansen, 2010] A. Jansen, K. Church, and H. Hermansky, "Towards spoken term discovery at scale with zero resources," in Proc. of INTERSPEECH, 2010, pp. 1676–1679.

[Jansen, 2011] A. Jansen and B. Van Durme, "Efficient spoken term discovery using randomized algorithms," in Proc. of IEEE ASRU, 2011, pp. 401–406.

[Jansen, 2013] A. Jansen, E. Dupoux, S. Goldwater, M. Johnson, S. Khudan-pur, K. Church, N. Feldman, H. Hermansky, F. Metze, R. Rose, M. Seltzer, P. Clark, I. McGraw, B. Varadarajan, E. Bennett, B. Borschinger, J. Chiu, E. Dunbar, A. Fourtassi, D. Harwath, C. Lee, K. Levin, A. Norouzian, V. Peddinti, R. Richardson, T. Schatz, and S. Thomas, "A summary of the 2012 JHU CLSP workshop on zero resource speech technologies and models of early language acquisition," in Proc. of ICASSP, 2013, pp. 8111–8115.

[Jelinek, 1998] Jelinek, F., Statistical Methods for Speech Recognition, The MIT Press, Cambridge, MA, 1998.

[Juang, 2005] Juang, B.H, Rabiner, L. (2005). Automatic Speech Recognition - A Brief History of the Technology Development, Elsevier Encyclopedia of Language and Linguistics.

[Jurafsky, 2008] Jurafsky, D., Martin J. H. (2008). "Speech and Language Processing, 2nd Edition", *Ch. 9. Prentice Hall*, USA.

[Kacur, 2006]]Kačur J. (2006). HTK vs. SPHINX for SPEECH Recognition, Department of telecommunication, FEI STU, Bratislava Slovakia.

[Kelly, 2010] F. Kelly, N. Harte, "A COMPARISON OF AUDITORY FEATURES FOR ROBUST SPEECH RECOGNITION", In Proc. Of 18th European Signal Processing Conference (EUSIPCO-2010).

[Kim, 2009] C. Kim and R. M. Stern, "Feature extraction for robust speech recognition using a power-law nonlinearity and power-bias subtraction," INTERSPEECH 2009,Sept 2009.

[Kim, 2010] C. Kim, "Signal Processing for Robust Speech Recognition Motivated by Auditory Processing", PhD Thesis, Language Technologies Institute School of Computer Science Carnegie Mellon University, 2010

[Kleijn, 1998] Kleijn K., Paliwal K. "Speech Coding and Synthesis." Elsevier Science B.V., The Netherlands, 1998.

[Koehn, 2010] Koehn, P., "Statistical Machine Translation", Cambridge University Press, 2010.

[Kohonen, 1984] Kohonen, T. (1984), "Self-Organization and Associative Memory". Berlin: Springer-Verlag.

[Lamere, 2003] Lamere, P., Kwok, P., Gouvêa. E., Raj, B., Singh, R., Walker W., Warmuth. M., Wolf, P. (2003). The CMU SPHINX-4 Speech Recognition System. Proc. of ICASSP 2003.

[Lemmetty, 1999] S. Lemmetty. "Review of Speech Synthesis Technology", PhD Thesis, Helsinki University of Technology, 1999.

[Lin, 2009] H. Lin, L. Deng, D. Yu, Y. Gong, A. Acero and C.H. Lee, "A study on multilingual acoustic modeling for large vocabulary ASR," ICASSP: Proceedings of the Acoustics, Speech, and Signal Processing, Taipei, Taiwan, pp. 4333–4336, (2009).

[Liu, 2006] Y. Liu, E. Shriberg, A. Stolcke, D. Hillard, M. Ostendorf, M. Harper, "Enriching speech recognition with automatic detection of sentence boundaries and disfluencies," IEEE Trans. Audio Speech Lang. Process., vol. 14(5), pp. 1526–1540, 2006.

[Lita, 2003] L. V. Lita, A. Ittycheriah, S. Roukos, N. Kambhatla, "tRuEcasIng", in ACL, Japan 2003.

[Ludusan, 2014a] B. Ludusan, G. Gravier, and E. Dupoux, "Incorporating prosodic boundaries in unsupervised term discovery," in Proc. of Speech Prosody, 2014, pp. 939–943.

[Ludusan, 2014b] B. Ludusan, M. Versteegh, A. Jansen, G. Gravier, X.-N. Cao, M. Johnson, and E. Dupoux, "Bridging the gap between speech technology and natural language processing: an evaluation toolbox for term discovery systems," in Proc. of LREC, 2014, pp.560–567.

[Ma, 2009] Ma, G., Zhou W., Zheng J., You X and Ye W. (2009). A Comparison between HTK and SPHINX on Chinese Mandarin. In Artificial Intelligence, 2009. JCAI '09, pp 394 – 397.

[Mamou, 2007] Mamou J., Ramabhadran B., Siohan O., Vocabulary Independent Spoken Term Detection, SIGIR '07 Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval, 615-622, (2007).

[Malioutov, 2007] I. Malioutov, A. Park, R. Barzilay, and J. Glass, "Making sense of sound: Unsupervised topic segmentation over acoustic input," in Proc. of ACL, 2007, pp. 504–511.

[Matjka, 2005] P. Matjka, P. Schwarz, and P. Chytil, “Phonotactic language identification using high quality phoneme recognition,” in Proc. of Eurospeech, 2005, pp. 2237–2240.

[MediaEval, 2015] MediaEval Benchmarking Initiative for Multimedia Evaluation, accessed august 2015, http://multimediaeval.org/mediaeval2015/quesst2015/

[Meng, 2008] MENG, S., Y U, P., L IU, J., , SEIDE, F. “Fusing multiple systems into a compact lattice index for Chinese spoken term detection.” In Proc. ICASSP’08. Las Vegas, Nevada, USA, 4345–4348, (2008).

[Mihalcea, 2002] Mihalcea, R., Nastase, V., “Letter Level Learning for Language Independent Diacritics Restoration,” CoNLL 2002, Taipei, Taiwan, pp. 105-111, 2002.

[Militaru, 2014] Militaru D., Gavat I. (2014). A Historically Perspective of Speaker-Independent Speech Recognition in Romanian Language, Sisom & Acoustics 2014, Bucharest, Romania.

[Modis, 2015] Motif discovery software. https://gforge.inria.fr/projects/motifdiscovery/, accessed august 2015.

[Morgan, 1995] Morgan N., Bourlard H., “An Introduction to Hybrid HMM/Connectionist Continuous Speech Recognition”, IEEE Signal Processing Magazine (May 1995).

[Motlicek, 2010] Motlicek P., Valente F., “Application of out-of-language detection to spoken term detection,” ICASSP: Proceedings of the Acoustics, Speech, and Signal Processing, USA, pp. 5098-5101, (2010).

[MSU, 2003] Michigan State University (MSU), “Linguistics 401”, Online Course, section 2, December 3, 2003 https://www.msu.edu/course/lin/401/fs03-s2/comp-lecture.pdf, accesed august 2015.

[Meignier, 2006] S. Meignier, D. Moraru, C. Fredouille, J.-F. Bonastre, and L. Besacier, “Step-by-step and integrated approaches in broadcast news speaker diarization,” Computer Speech and Language, vol. 20, no. 2-3, pp. 303–330, 2006.

[Meignier, 2010] S. Meignier, T. Merlin, “LIUM SpkDiarization: An Open Source Toolkit For Diarization,” in Proc. CMU SPUD Workshop, 2010.

[Moreno, 1996] P. J. Moreno, B. Raj, and R. M. Stern, “A vector Taylor series approach for environment-independent speech recognition,” in IEEE Int. Conf. Acoust., Speech and Signal Processing, May. 1996, pp. 733–736.

[Müller, 2007] Müller, M., “Information Retrieval for Music and Motion”, Springer-Verlag Berlin Heidelberg, 2007.

[Murao] Murao H., Kawaguchi N., Matsubara S., Inagaki Y., “Example-based query generation for spontaneous speech,” IEICE - Trans. Inf. Syst., (2005).

[Muscariello, 2011] A. Muscariello, G. Gravier, and F. Bimbot, “Zero-resource audio-only spoken term detection based on a combination of template matching techniques,” in Proc. of INTERSPEECH, 2011, pp. 921–924.

[Muscariello, 2012] A. Muscariello, G. Gravier, and F. Bimbot, “Unsupervised motif acquisition in speech via seeded discovery and template matching combination,” IEEE Transactions on Audio, Speech, and Language Processing, vol. 20, no. 7, pp. 2031–2044, 2012.

[Ng, 2000] Ng K., Zue V., “Subword-based approaches for spoken document retrieval,” in PhD. Thesis, (2000).

[Nguyen, 2008] T. Hieu Nguyen, E. S. Cheng, and H. Li, “T-test distance and clustering criterion for speaker diarization,” In Proc. Interspeech pp.36-39, 2008.

[NIST, 2014] NIST, Speech Recognition Scoring Toolkit (SCTK), 2014. Online, National Institute of Standards and Technology, http://www.nist.gov/itl/iad/mig/tools.cfm

[Oh, 2007] Oh, Y.R., Yoon, J.S., Kim, H.K., “Acoustic model adaptation based on pronunciation variability analysis for non-native speech recognition,” Speech Communication, 49, pp.59-70, 2007.

[O’Shaughnessy, 1987] D. O’Shaughnessy. “Speech Communication, Human and Machine.” Addison-Wesley, 1987.

[Patty, 2006] Patty L., Spoken Term Detection Evaluation Plan, National Institute of Standards and Technology, (2006).

[Pardo, 2007] J. M. Pardo, X. Anguera, and C. Wooters, “Speaker diarization for multiple-distant-microphone meetings using several sources of information,” IEEE Transactions on Computers, vol. 56, no. 9, pp. 1212–1224, 2007.

[Parada, 2010] Parada C., Sethy A., Ramabhadran B., “Balancing false alarms and hits in Spoken Term Detection,” ICASSP: Proceedings of the Acoustics, Speech, and Signal Processing, USA, pp. 5286-5289, (2010).

[Parlak, 2008] Parlak S., Saraclar M., “Spoken Term Detection for Turkish Broadcast News,” ICASSP: Proceedings of the Acoustics, Speech, and Signal Processing, USA, pp. 5244 - 5247, (2008).

[Pașca, 2012] M. Pașca, „METHODS OF PERFORMANCE IMPROVEMENT FOR ROMANIAN CONTINUOUS SPEECH RECOGNITION TASKS", PhD Thesis, 2012.

[Petrica, 2014] L. Petrică*, H. Cucu, A. Buzo, and C. Burileanu, "A Robust Diacritics Restoration System Using Unreliable Raw Text Data", SLTU, 2014.

[Pitt, 2007] M. Pitt, L. Dilley, K. Johnson, S. Kiesling, W. Raymond, E. Hume, and E. Fosler-Lussier, "Buckeye corpus of conversational speech (2nd release)," Columbus, OH: Department of Psychology, Ohio State University, 2007.

[Poritz, 1988] Poritz, A., "Hidden Markov models: a guided tour," ICASSP 1988, pp. 7–13.

[Park, 2008] A. Park and R. Glass, "Unsupervised pattern discovery in speech," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 16, no. 1, pp. 186–197, 2008.

[Price, 2006] Price J. and Eydgahi A. "Design of Matlab®-Based Automatic Speaker Recognition Systems". *In the 9th International Conference on Engineering Education*, 2006.

[Pollak, 2000] P. Pollák, J. Boudy, K. Choukri, H. van den Heuvel, K. Vicsi, A. Virag, R. Siemund, W. Majewski, P. Staroniewicz, H. Tropf, J. Kochanina, A. Ostroukhov, M. Rusko, and M. Trnka, "SpeechDat (E) - Eastern European telephone speech databases." in Proc. of Workshop on Very Large Telephone Speech Databases, 2000.

[Pujol, 2006] P. Pujol, D. Macho, and C. Nadeu, "On real-time mean-and-variance normalization of speech recognition features," in IEEE Int. Conf. Acoust., Speech and Signal Processing, vol. 1, May 2006, pp. 773–776.

[Rabiner, 1993] Rabiner, L.R., Juang, B.H.: Fundamentals of Speech Recognition, Prentice-Hall, NJ, 1993.

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

[Rabiner, 2007] Rabiner L. R., Schafer R. W. (2007). Introduction to Digital Speech Processing, Foundations and Trends in Signal Processing, Vol. 1, Nos. 1–2 (2007) 1–194.

[Renalds, 2010] Renalds S., Hain, T., "Speech Recognition," Handbook of Computational Linguistics and Natural Language Processing, 2010.

[Reynolds, 2009] Reynolds D., „Gaussian Mixture Models", MIT Lincoln Laboratory, Lexington, 2009.

[Rijsbergen, 1995] C. J. van Rijsbergen, "Information Retrieval", Department of Computing Science, University of Glasgow, 1995.

[Robertson, 1976] ROBERTSON, S.E. and SPARCK JONES, K., 'Relevance weighting of search terms', Journal of the American Society for Information Science, 27, 129-146 (1976).

[Rosenfeld, 2000] R. Rosenfeld, "Two decades of statistical language modeling: where do we go from here?", Proceedings of the IEEE, 2000.

[Sakoe, 1978] Sakoe, H. and Chiba, S., Dynamic programming algorithm optimization for spoken word recognition, IEEE Transactions on Acoustics, Speech and Signal Processing, 26(1) pp. 43- 49, 1978.

[Sam, 2010] Sam, S., Castelli, E., Besacier. L., "Unsupervised Acoustic Model Adaptation for Multi-Origin Non Native ASR," Interspeech 2010, pp. 254-257, Tokyo, Japan, 2010.

[Schatz, 2013] Schatz, T., Peddinti, V., Bach, F., Jansen, A., Hynek, H. & Dupoux, E. "Evaluating speech features with the Minimal-Pair ABX task: Analysis of the classical MFC/PLP pipeline". In Interspeech (pp 1781-1785), 2013.

[Schwarz, 2004] P. Schwarz, P. Matejka, J. Cernock, "Towards Lower Error Rates in Phoneme Recognition", in Proc. TSD2004, Brno, Czech Republic, 2004.

[Schwarz, 2006] P. Schwarz, P. Matejka, J. Cernocky, "Hierarchical Structures of Neural Networks for Phoneme Recognition", submitted for publication to ICASSP2006.

[Schwarz, 2009] P. Schwarz, "Phoneme Recognition based on Long Temporal Context", PhD Thesis, Brno University of Technology, 2009.

[Shieber] S. M. Shieber, X. Tao. "Comma restoration using constituency information", in Proceedings of the 3rd International Conference on Human Language Technology Research, Edmonton, Canada, 2003.

[Shriberg, 2000] E. Shriberg, A. Stolcke, D. Hakkani-Tur, and G. Tur, "Prosodybased automatic segmentation of speech into sentences and topics," Speech Comm., vol. 32(1-2), pp. 127–154, 2000.

[Speechdat-E, 2015] Eastern European Speech Databases for Creation of Voice Driven Teleservices, http://www.fee.vutbr.cz/SPEECHDAT-E/, accessed august 2015.

[SpeeD, 2015] “Speech and Dialogue Research Group”, *University POLITEHNICA of Bucharest*. (http://speed.pub.ro)

[Sphinx, 2015] CMU Sphinx Project by Carnegie Mellon University, http://cmusphinx.sourceforge.net/, accessed august 2015.

[Standford, 2015] “Machine learning”, *Coursera online courses, Engineering & Computer Science*, accessed August 2015. (http://online.stanford.edu/course/machine-learning-1)

[Stadtschnitzer, 2008] Stadtschnitzer, M., Van Pham, T., Chien, T.T.: "Reliable voice activity detection algorithms under adverse environments." In: International Conference on Communications and Electronics, pp. 218–223 (2008)

[Stevens, 1937] Stevens, S.S., Volkmann, J.E., Newman, E.B. “A Scale for the Measurement of the Psychological Magnitude Pitch”. Journal of the Acoustical Society of America 8(3), 185–190 (1937).

[Stevens, 1940] Stevens, S.S., Volkmann, J.E. “The Relation of Pitch to Frequency”. Journal of Psychology 53(3), 329–353 (1940).

[Sigh, 2002] R. Singh, R. M. Stern, and B. Raj, “Signal and feature compensation methods for robust speech recognition,” in Noise Reduction in Speech Applications, G. M. Davis, Ed. CRC Press, 2002, pp. 219–244.

[Stuttle, 2003] M. N. Stuttle, “A Gaussian Mixture Model Spectral Representation for Speech Recognition”. *PhD Thesis*, Hughes Hall and Cambridge University Engineering Department, July 2003

[Tan, 2007] Tan, T.-P., Besacier. L., “Acoustic Model Interpolation for Non-Native Speech Recognition,” ICASSP 2007, pp. 1009-1012, Honolulu, USA, 2007.

[Tan, 2008] Tan, T.-P., Automatic Speech Recognition for Non-Native Speakers. PhD Thesis, University Joseph Fourier, Grenoble, France, 2008.

[Thomas, 2011] S. Thomas, P. Nguyen, G. Zweig and H. Hermansky, “MLP BASED PHONEME DETECTORS FOR AUTOMATIC SPEECH RECOGNITION”, in Proc. Of Acoustics, Speech and Signal Processing (ICASSP), 2011.

[Tufiş, 1999] Tufiş, D., Chiţu, A., “Automatic Insertion of Diacritics in Romanian Texts,” International Workshop on Computational Lexicography 1999, pp. 185-194, Pecs, Hungary, 1999.

[Tufiş, 2008] Tufiş, D., Ceauşu. D., “DIAC+: A Professional Diacritics Recovering System,” LREC 2008, Marrakech, Morocco, 2008.

[Ungurean, 2008] Ungurean, C., Burileanu, D., Popescu, V., Negrescu, C., Dervis, A., “Automatic Diacritics Restoration for a TTS-based E-mail Reader Application,” University Politehnica of Bucharest Scientific Bulletin, Series C, vol. 70, no. 4, Bucharest, Romania, 2008.

[Ungurean, 2011] Ungurean, C., Burileanu, D., “An advanced NLP framework for high-quality Text-to-Speech synthesis,” SpeD 2011, Braşov, Romania, 2011.

[Varga, 1990] A. Varga and R.K. Moore. “Hidden markov model decomposition of speech and noise.” In Proceedings ICASSP, pages 845–848, 1990.

[Venturebeat, 2015] http://venturebeat.com/2015/05/28/google-says-its-speech-recognition-technology-now-has-only-an-8-word-error-rate/, accesed august 2015.

[Vries, 2014] N. de Vries, M. Davel, J. Badenhorst, W. Basson, F. de Wet, E. Barnard, and A. de Waal, “A smartphone-based asr data collection tool for under-resourced languages,” Speech Communication, vol. 56, pp. 119–131, 2014.

[Versteegh, 2015] M. Versteegh, R. Thiolliere, T. Schatz, X.N. Cao, X. Anguera, A. Jansen, and E. Dupoux. "The Zero Resource Speech Challenge 2015." In Proc. of INTERSPEECH, 2015.

[Wade, 2009] Wade S., Christopher M. White, Timothy J. Hazen, “A Comparison of Query-by-Example Methods for Spoken Term Detection”, MIT/Lincoln Laboratory, (2009).

[Waibel, 1989] Waibel A., Hanazawa T., Hinton G., Shikano K. and Lang K. J., (1989). Phoneme recognition using time-delay neural networks. IEEE Transactions on Acoustics, Speech and Signal Processing, vol. 37, pp. 328-339.

[Wang, 2010] Wang D., King S., Frankel J., Bell P., “Stochastic pronunciation modeling and soft match for out-of-vocabulary spoken term detection,” ICASSP: Proceedings of the Acoustics, Speech, and Signal Processing, (2010).

[Witten, 1982] Witten I. "Principles of Computer Speech", Academic Press Inc., 1982.

[Woodland, 1998] P.C. Woodland, T. Hain, S.E. Johnson, T.R. Niesler, A. Tuerk, E.W.D. Whittaker, and S.J. Young. "The 1997 HTK broadcast news transcription system". In Proc. 1998 DARPA Broadcast News Transcription and Understanding Workshop, 1998.

[Woodland, 1999] P.C. Woodland. "Speaker adaptation: techniques and challenges." In Proceedings ASRU, 1999.

[Wu, 1993] Wu J. and Chan C. (1993). Isolated Word Recognition by Neural Network Models with Cross-Correlation Coefficients for Speech Dynamics. IEEE Trans. Pattern Anal. Mach. Intell., vol. 15, pp. 1174-1185.

[Young, 2006] S. J. Young, G. Evermann, M. J. F. Gales, T. Hain, D. Kershaw, G. Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev, P. C. Woodland, "The HTK Book, version 3.4", CMU, 2006.

[Yu, 2004] YU, P. SEIDE, F. "A hybrid word / phoneme-based approach for improved vocabulary-independent search in spontaneous speech." In Proc. ICSLP'04. Jeju, Korea, 293–296, (2004).

[ZeroSpeech, 2015] The Zero Resource Speech Challenge, accessed august 2015, http://www.lscp.net/persons/dupoux/bootphon/zerospeech2014/website/index.html

[Zhang, 2001], I. C. B. a. L. H. C. X. Zhang, M. G. Heinz, "A phenomenological model for the responses of auditory-nerve fibers: I. nonlinear tuning with compression and suppression," vol. 109, pp. 648–670, Feb 2001.

[Zhang, 2010] Y. Zhang and J. Glass. "Towards multi-speaker unsupervised speech pattern discovery." In Proc. ICASSP, pages 4366–4369, 2010

[Zhang, 2013] Y. Zhang, "Unsupervised Speech Processing with Applications to Query-by-Example Spoken Term Detection", PhD Thesis, Massachusetts Institute of Technology, 2013.