

University “Politehnica” of Bucharest

Faculty of Electronics, Telecommunications and Information Technology

Multilingual Automatic Speech Recognition System

Diploma Thesis

submitted in partial fulfillment of the requirements for the Degree of
Engineer in the domain *Electronics and Telecommunications*, study program
Technologies and Systems of Telecommunications

Thesis Advisor

Dr. Ing. Andi BUZO

Dr. Ing. Horia CUCU

Student

Ioana CALANGIU

University "Politehnica" of Bucharest
Faculty of Electronics, Telecommunications and Information Technology
Department Telecommunications

Department Director's Approval:

Prof. Dr. Ing. Silviu CIOCHINĂ



DIPLOMA THESIS
of student Calangiu Ioana, group 441G

1. Thesis title: Multilingual Automatic Speech Recognition System

2. The student's original contribution will consist of (not including the documentation part): *Design of the automatic speech recognition system; gathering the necessary resources for creating a wide database: building the acoustic, language and phonetic models for 2 to 3 different languages and improve the recognition accuracy at an acceptable level; testing the system under different conditions; configuration tuning (each configuration depends on a series of parameters; finding the optimum setup of these parameters); study of current methods used for spoken word recognition; defining the requirements of the system (with a focus on its capabilities); conceptual analysis (schematic blocks regarding its architecture); implementation of the system; in the end, testing the system and drawing conclusions regarding its performances and drawbacks.*

3. The project is based on knowledge mainly from the following 3-4 courses: *Decision and Estimation in Information Processing, Object-Oriented Programming, Digital Signal Processing*

4. The construction part of the project remains in the property of: *UPB*

5. The Intellectual Property upon the project belongs to: *UPB*

6. The research is performed at the following location: *UPB*

7. The thesis project was issued at the date: 15.10.2013

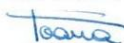
Thesis Advisor:

Ș.I. Dr. Ing. Horia CUCU



STUDENT:

Ioana CALANGIU



Statement of Academic Honesty

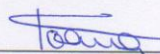
I hereby declare that the thesis "*Multilingual Automatic Speech Recognition System*", submitted to the Faculty of Electronics, Telecommunications and Information Technology in partial fulfillment of the requirements for the degree of Engineer of Science in the domain *Electronics and Telecommunications*, study program *Technologies and Systems of Telecommunications*, is written by myself and was never before submitted to any other faculty or higher learning institution in Romania or any other country.

I declare that all information sources I used, including the ones I found on the Internet, are properly cited in the thesis as bibliographical references. Text fragments cited "as is" or translated from other languages are written between quotes and are referenced to the source. Reformulation using different words of a certain text is also properly referenced. I understand plagiarism constitutes an offence punishable by law.

I declare that all the results I present as coming from simulations and measurements I performed, together with the procedures used to obtain them, are real and indeed come from the respective simulations and measurements. I understand that data faking is an offence punishable according to the University regulations.

Bucharest, 3.07.2014

Ioana CALANGIU



(student's signature)

Table of Contents

1	CHAPTER INTRODUCTION	13
1.1	THESIS MOTIVATION	13
1.2	THE FIELD OF SPEECH RECOGNITION	13
1.3	THESIS OBJECTIVES AND OUTLINES	15
2	CHAPTER NECESSARY RESOURCES FOR BUILDING AN ASR	17
2.1	RECOGNITION FORMALISM	17
2.2	LANGUAGE MODELING	18
2.2.1	N-GRAM MODELS	19
2.2.2	APPROACH OF THE DATA SPARSENESS PROBLEM	19
2.2.2.1	Back-off methods	19
2.2.2.2	Smoothing methods	21
2.2.3	EVALUATING THE PERFORMANCE OF THE LANGUAGE MODEL	21
2.2.3.1	Perplexity	22
2.2.3.2	Out Of Vocabulary words	22
2.2.3.3	N-gram hits	23
2.3	FSG GRAMMAR	23
2.4	PHONETIC MODELING	23
2.5	ACOUSTIC MODELING	24
2.5.1	FEATURE EXTRACTION	25
2.5.2	HMM FRAMEWORK	29
2.5.3	CHOOSING THE BASIC UNIT	32
2.6	ASR EVALUATION	33
2.7	SPEECH RECOGNITION TOOLS	34
3	CHAPTER AUTOMATIC SPEECH RECOGNITION SYSTEMS	35
3.1	CURRENT STAGE FOR AUTOMATIC RECOGNITION SYSTEM FOR ALBANIAN	35
3.2	SPEECH DATABASE ACQUISITION FOR ALBANIAN ASR	35
3.2.1	ACQUISITION TOOLS FOR AUDIO CLIPS FOR ALBANIAN ASR	35
3.2.2	ACQUISITION TOOLS FOR TRANSCRIPTIONS FOR ALBANIAN ASR	36
3.2.2.1	Diacritics	38
3.3	BUILDING AND ACOUSTIC, A LANGUAGE AND A PHONETIC MODEL FOR A SMALL ALBANIAN DATABASE	38
3.3.1	BUILDING THE LANGUAGE MODEL	38
3.3.2	BUILDING THE PHONETIC MODEL	39
3.3.2.1	Graphemes-to-phonemes method description	39
3.3.2.2	Phones list	39
3.3.3	BUILDING THE ACOUSTIC MODEL	41
3.3.3.1	Speech unit selection	41

3.4 DATA PREPARATION FOR TRAINING AND TESTING AN ASR	41
3.4.1 TRAINING	41
3.4.1.1 The hierarchical training strategy	44
3.4.2 TESTING	44
3.5 FINAL RESULTS FOR ALBANIAN ASR	45
3.6 AUTOMATIC SPEECH RECOGNITION SYSTEM FOR ENGLISH	46
3.6.1 BUILDING THE LANGUAGE MODEL	47
3.6.2 BUILDING THE PHONETIC MODEL	47
3.6.2.1 Phones list	47
3.6.3 BUILDING THE ACOUSTIC MODEL	49
3.7 AUTOMATIC SPEECH RECOGNITION SYSTEM FOR ROMANIAN	49
3.7.1 BUILDING THE LANGUAGE MODEL	49
3.7.2 BUILDING THE PHONETIC MODEL	50
3.7.2.1 Phones list	50
3.7.3 BUILDING THE ACOUSTIC MODEL	51
3.8 DEMO APPLICATION	52
<u>4 CONCLUSIONS</u>	<u>55</u>
<u>BIBLIOGRAPHY</u>	<u>57</u>
<u>ANNEX 1</u>	<u>59</u>

Table of Figures

Figure 1.1 A source-channel model for speech recognition system	14
Figure 2.1 Necessary resources for building an ASR.....	18
Figure 2.2 Necessary sets for training a language model	20
Figure 2.3 Matlab figure illustrating the energy of vowels	25
Figure 2.4 The stages for extracting the MFC coefficients	26
Figure 2.5 Frequency bands on Mel scale[12].....	27
Figure 2.6 HMM-based phone model with 5 states[15]	30
Figure 3.1 Example of Albanian .txt file in raw form	37
Figure 3.2 Cleaned Albanian text	38
Figure 3.3 transcription file for language model	38
Figure 3.4 Example of output during decoding	42
Figure 3.5 GUI for Demo application	52
Figure 3.6 The language word graph.....	52
Figure 3.7 FSG Grammar for Demo application	53
Figure 3.8 GUI for Romanian Recognition	53

Table of Tables

Table 3-1 MediaEval 2013 database	38
Table 3-2 Albanian Phoneme set.....	40
Table 3-3Structure of decoding process	43
Table 3-4 Albanian Speech database.....	45
Table 3-5 Albanian Text copora.....	45
Table 3-6 Albanian Acoustic Models.....	45
Table 3-7 Albanian Language Models	46
Table 3-8 Albanian experimental results.....	46
Table 3-9 TIMIT Text corpora	47
Table 3-10Language model evaluation for TIMIT database	47
Table 3-11Phoneme set in English	48
Table 3-12 Results for TIMIT database	49
Table 3-13 Romanian database.....	49
Table 3-14 number of words for Romanian database.....	50
Table 3-15List of phones in Romanian	51
Table 3-16 Evaluation for Romanian database.....	51

1 CHAPTER Introduction

1.1 Thesis motivation

From human prehistory to the new media of the future, speech communication has been and will be the dominant mode of the human social bonding and information exchange. In addition to human-human interaction, this human preference for spoken language communication finds a reflection in human-machine interaction as well. Designing a machine that mimics human behavior, in particular the capability of speaking naturally and responding properly to spoken language, has intrigued engineers and scientists for centuries. Homer W. Dudley was the pioneering electronic and acoustic engineer in this field, by creating the first electronic voice synthesizer for Bell Labs in the 1930s and leading the development of a method of sending secure voice transmissions during World War Two.

New machine learning algorithm can lead to significant advances in automatic speech recognition. The biggest single advance occurred nearly four decades ago with the introduction of the Expectation-Maximization (EM) algorithm for training Hidden Markov Models (HMMs). Through the EM algorithm, it became possible to develop speech recognition systems for real world tasks using richness of Gaussian mixture models (GMM) to represent the relationship between the acoustic input and the HMM states. In these systems the acoustic input is created by concatenating Mel Frequency Cepstral Coefficients (MFCCs), computed from the raw waveform, and their first- and second-order temporal differences. This pre-processing of the input signal is designed to discard the large amount of information in waveforms that is considered irrelevant.

The field of Automatic Speech Recognition (ASR) exploded in the last decades, since people tend to be more and more busy and look after hands-free and eyes-free interfaces to devices. The object of ASR is to capture an acoustic signal representative of speech and determine the words that were spoken by pattern matching. To do this, a set of acoustic and language models have to be stored in a computer database, that represent the actual patterns. These models result after training and are then compared to the capture signals

1.2 The field of speech recognition

Recognition and understanding of spontaneous unrehearsed speech remains an elusive goal. To understand speech, a human considers not only the specific information conveyed to the ear, but also the context in which the information is being discussed. For this reason, people can understand spoken language even when the speech signal is corrupted by noise. However, understanding the context of speech is, in turn, based on broad knowledge of the world. And this has been the source of the difficulty and over forty years of research.

Automatic speech recognition is the recognition of the information embedded in a speech signal and its transcription in terms of a set of characters. The ASR process addresses the problem of mapping an acoustic signal to a sequence of words. When the input acoustic signal contains speech uttered by different speakers, the ASR task can be regarded as a two-step process : speaker diarisation (who spoke when?) and speech-to-text transcription (what did he say?).

The task of speech recognition can be formulated through a source-channel model. The speaker's mind decides the source word sequence W that is delivered through his/her text generator. The source is passed through a noisy communication channel that consists of the speaker's vocal apparatus to produce the speech waveform and the speech signal processing component of the speech recognizer. At the last stage, the decoder aims to decode the acoustic signal X into a word sequence \hat{W} , which should be as close as possible to the original sequence W .

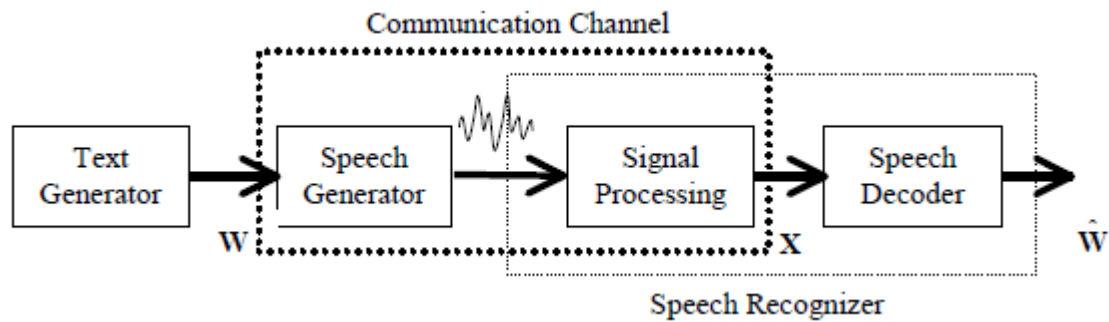


Figure 1.1 A source-channel model for speech recognition system[8]

The speech signal is processed in the signal processing model that extracts feature vectors for the decoder. The decoder uses both acoustic and language models to generate the word sequence that has the maximum probability for the input feature vectors. Acoustic models refer to the representation of the information and knowledge about acoustics, phonetics, environment variability, gender, different pronunciations and dialect differences among speakers etc. Language models refer to a system's intuition of what constitutes a valid word and what words are most likely to occur.

Several problems appear when building an ASR, and mostly depend on the type of language. For a vast number of languages, called *low-resourced language*, there are no text and speech resources available. These language are spoken by a large number of people, but no prior work of collecting and organizing speech and/or text resources has been made. In this case, the task of implementing an ASR includes gathering the necessary resources for creating a wide database.

Other languages, like French and Romanian, are categorized as *rich-morphology language*. Compared to English, a *poor-morphological language*, these languages have a large vocabulary. For example the word *to learn* in Romanian has six morphologically different forms : „învăţ”, „înveţi”, „învăţă”, „învăţăm”, „învăţaţi”, „învăţă”. In French it has four : „apprends”, „apprend”, „apprenons”, „apprenez”, „apprennent”. The right morphological variant depends on various factors, like constraints or grammatical gender. In English, the same verb has only two forms : „learn” and „learns”. German and Turkish are some of the so-called *agglutinative languages*. Agglutination is a process in which complex words are formed by stringing together morphemes, each with a single grammatical or semantic meaning. This process translates into a very large vocabulary, which makes the task of speech recognition even more challenging.

The *size of vocabulary* is also an important factor which settles the difficulty when designing an ASR. The task of recognizing a set of commands, with a limited number of words, is much simpler than a spontaneous recognizing task (with 64k words vocabulary). Nevertheless, a large vocabulary does not always translate into a more difficult ASR task. The *linguistic uncertainty* of the possible speech utterances plays a significant role. For instance, an ASR targeted to recognize tourism

related words (which can form a 64k words vocabulary) is not as difficult as a spontaneous speech recognition task with an equals-size vocabulary. The low linguistic uncertainty, also called perplexity, of the tourism-specific ASR task makes it less difficult.

After years of research and development, accuracy of automatic speech recognition remains one of the most important research challenges. A number of well-known factors determine accuracy: those most noticeable are variations in context, in speaker and in environment. Acoustic modeling plays a critical role in improving accuracy and is arguably the central part of any speech recognition system.

One of the factors that influences the accuracy of the speech recognition system is the *acoustic environment* in which the speaker is placed, along with any *transmission channel*. In this cases, it is a demanding task to separate the different acoustic signals found in an environment, which can be other talkers or environmental noise. This factor is also influenced by microphones, which can have a great impact on the speech recognition accuracy. In laboratories, the research is done with high-quality, head-mounted microphones. Other types of microphones can cause problems due to movements of the speaker's head relative to the microphone. In a similar manner to speaker-independent training, we can build a system by using a large amount of data collected from a number of environments; this is referred to as *multistyle training*. Nevertheless, despite the progress being made in the field, environment variability remains as one of the most severe challenges facing today's state-of-the-art speech systems.

The accuracy of a speech recognition process is also influenced by the speaker characteristics. By speaker characteristics, one refers to the speaker accent, the gender, the speech rate, different pronunciations or even dialect differences. Every individual speaker is different. As such, one person's speech patterns can be entirely different from those of another person. Even if these interspeaker differences could be excluded, the same speaker is unable to precisely produce the same utterance. Along with these, the speaking style also plays an important role in designing an ASR. The speaking style refers to how fluent, natural or conversational the speech is. The interspeaker variability could be dealt with by simply designing *speaker-dependent* ASR systems. Nevertheless, even if this would translate into a small error rate, the drawback is that a new acoustic model should be trained for every new speaker. Consequently, *speaker-independent* ASR systems are more flexible, since they can be used to recognize the speech of any speaker.

Another factor to be taken into consideration is style variability. To deal with acoustic realization variability, a number of constraints can be imposed on the use of the speech recognizer. For instance, there are *isolated speech recognition systems*, in which users have to pause between each word. Because the pause provides a clear boundary for the word, we can easily eliminate errors such as *Ford or* and *Four Door*. In continuous speech recognition, the error rate is usually much higher than in the case of isolated speech recognition. If a person whispers, or shouts, to reflect his or her emotional changes, the variation increases more significantly.

1.3 Thesis objectives and outlines

The main objective of this thesis was to develop a speaker-independent large-vocabulary continuous speech recognition system for Albanian, a under-resourced language. This system should be able to recognize general Albanian continuous speech produced by any speaker with a decent performance. Several stages were followed to achieve this final goal :

1. The acquisition of a phonetic, speech and text resources. A speech database is needed to train the acoustic model, while a text database is required to create a general purpose language model. A phonetic model links the acoustic model, a spectral representation of sounds and words, to the language models, that is a representation of the grammar or syntax of the task.
2. The development of specific tools to process the necessary resources presented above.
3. The design, implementation and evaluation of an Albanian large vocabulary continuous speech recognition system using state-of-the-art techniques : the HMM framework for acoustic modeling and the n-gram paradigm for language modeling.

The thesis is organized in four chapters as follows.

Chapter 1 presents a brief summary of the main issues in the field of speech recognition and the important factors that influence the accuracy of a large vocabulary continuous speech recognition system.

Chapter 2 presents a brief summary of the main issues in the field of speech recognition. The second chapter introduces the reader the concepts of acoustic, phonetic and language modeling, which represent the engines of a continuous speech recognition system. This chapter ends with presenting some metrics computed in order to evaluate the ASR.

In *Chapter 3*, in the first sections, we focus on presenting the steps and difficulties in developing an ASR from zero. Our targeted language is Albanian, a *low-resourced language*. In the beginning of this chapter are illustrated acquisition tools for a speech database (audios clips and transcripts), The chapter continues with the stages of training the acoustic and the language models for a small Albanian database. After that it describes our efforts to extend this database, in order to have more accurate models, and smaller word error rates. Chapter 3 ends with the final results we obtained. The last part of Chapter 3 is dedicated to Romanian and English speech recognition systems, for which there are speech databases available. There are presented problems we encountered on the way, and the solutions we have come up with. Moreover, a demo is described in order to evaluate each of the three targeted languages.

Chapter 4 summarizes the main conclusions of this thesis and underlines the author's contributions. This chapter ends with briefly discusses regarding future developments for increasing the systems' accuracy.

2 CHAPTER Necessary resources for building an ASR

2.1 Recognition formalism

The process of automatic speech recognition is the translation of spoken words into text. This speech-to-text task can be characterized in a probabilistic framework. Probability theory and statistics provide the mathematical language to analyze and describe ASR systems. The speech-to-text task can be formulated in a probabilistic manner :

What is the most likely sequence of words W^ in a certain Language L , given the speech utterance X ?*

The formal representation uses the arg max function, in order to select the argument which maximizes the probability of the word sequence is :

$$W^* = \arg \max_W p(W/X) \quad (2.1)$$

The Equation 2.1. points to the most probable sequence of words as the one with the highest posterior probability, given the speech utterance. This posterior probability is computed using Bayes rule, so the most probable word sequence becomes:

$$W^* = \arg \max_W \frac{p(X/W) * p(W)}{p(X)} \quad (2.2)$$

$p(X)$, the probability of the speech utterance is independent of the sequence of words W , and can be ignored. The problem of recognition is simply reduced to :

$$W^* = \arg \max_W p(X/W) * p(W) \quad (2.3)$$

Equation 2.3. points out two terms that can be directly estimated : a) the apriori probability of the word sequence $p(W)$ and b) the probability of the acoustic data, given the word sequence, $p(X/W)$. The first factor can be estimated using a language model, while the second factor can be estimated with the help of an acoustic model. The two models can be built independently, but will be used together in order to decode the spoken data, as shown by equation 2.3. The general architecture of an ASR is presented in Figure 2.1. In can be seen that the speech recognition process is mainly described by two essential phases : a) extracting useful information from speech signal and b) compressing these representations for efficient transmission and storage. [1]

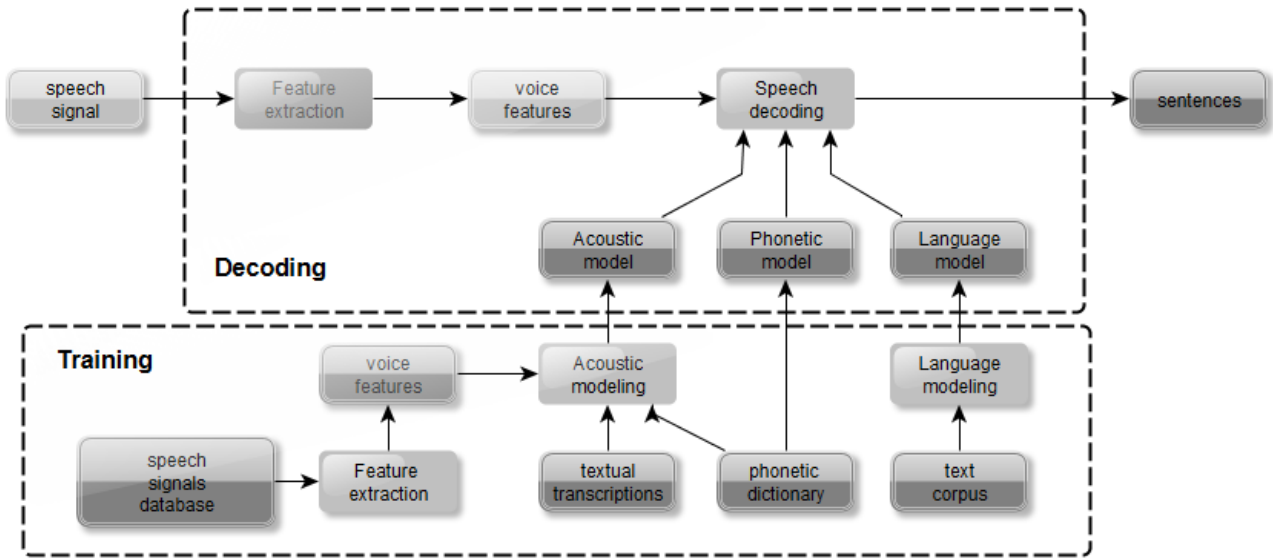


Figure 2.1 Necessary resources for building an ASR

Besides the acoustic model and the language model, which have been already mentioned above, the general architecture of the ASR also includes a phonetic model. Its purpose is to connect the acoustic model to the language model.

Figure 2.1. also shows that the system performs, in the early phase, a feature extraction. This block has the role to extract specific acoustic features which are further used to create the acoustic model. Consequently, the same feature extraction block is used in the decoding process.

Section 2.2. continues with the analysis and description of the several blocks in Figure 2.1.

2.2 Language modeling

The language model (grammar) is used in the decoding phase to describe how likely, in a probabilistic sense, is a sequence of language symbols that can appear in the speech signal. A statistical language model assigns a probability to a sequence of n words $P(w_1, w_2, \dots, w_n)$ by means of probability distribution. The main purpose of the grammar is to estimate the probability that a word sequence $W = w_1, w_2, \dots, w_n$, is a valid sentence in the researched language. The probability of these word sequences help the acoustic model in the decision process. [1]

In other words, a language model is used to restrict word search. It defines which word could follow previously recognized words and helps to significantly restrict the matching process by stripping words that are not probable.

The probability of the word sequence $W = w_1, w_2, \dots, w_n$ can be decomposed as follows

$$\begin{aligned}
 p(W) &= p(w_1, w_2, \dots, w_n) = p(w_1)p(w_2|w_1)p(w_n|w_1, w_2, \dots, w_{n-1}) \\
 &= \prod_{i=1}^n p(w_i|w_1, w_2, \dots, w_{i-1})
 \end{aligned} \tag{2.4}$$

where $p(w_i|w_1, w_2, \dots, w_{i-1})$ is the probability that w_i will follow, given that the word sequence w_1, w_2, \dots, w_{i-1} was present previously.

Now, the task of estimating the probability of a word sequence has been split into several tasks of estimating the probability of one word given a history of preceding words. In Eq. 2.4 the choice of w_i thus depends on the entire part history of the input. For a vocabulary of size v there are v^{i-1} different histories and in order to specify $p(w_i|w_1, w_2, \dots, w_{i-1})$ completely, v^i values would have to be estimated. In reality, the probabilities $p(w_i|w_1, w_2, \dots, w_{i-1})$ are impossible to estimate for even moderate values of i , since most histories w_1, w_2, \dots, w_n are unique or have occurred only a few times. A practical solution is to assume that $p(w_i|w_1, w_2, \dots, w_{i-1})$ depends only on some equivalence classes. The equivalent class can be simply based on the several previous words $w_{i-N+1}, w_{i-N+2}, \dots, w_{i-1}$. This leads to an n -gram language model. The trigram is a particular case, and has proven to be very powerful, since most words have a strong dependence on the previous two words. [2]

2.2.1 N-gram models

The n -gram model, which characterizes the word relationship within a span of n words, is a very powerful statistical representation of a grammar. Its effectiveness in building a word search was strongly validated by the famous word game of Claude Shannon which consisted in a competition between a human and a computer. In this competition, both the human and the computer were asked to sequentially guess the next word in an arbitrary sentence. The human guessed based on native experience with language, while the computer based its answers on maximum likelihood principle, using the accumulated word statistics. This experiment showed that, when n , the number of preceding words, exceeds 3, the computer was very likely to make a better guess of the next word in the sentence than the human. Unigrams are terrible at this game, but is easy to understand why. Currently, n -gram models are indispensable in large vocabulary speech recognition systems.[3]

2.2.2 Approach of the data sparseness problem

The text available for building a model is called a training corpus. For n -gram models, the amount of training data used is typically many millions of words.

Data sparseness is a problem which may appear even in the cases when there is a large training corpus put on disposal. No matter the size of the training corpus, there may always appear n -grams in the decoding phase, which were not found in this text.

2.2.2.1 Back-off methods

Sometimes it helps to use less context than more. In the cases when a trigram appears a very large number of times, it can be considered a very good estimator. But sometimes, a trigram does not appear that often, so a better solution is to back-off and to use a bigram. If the bigram is not trust worthy, as well, a unigram may provide a more useful information. The interpolation method proposes the mix of unigrams, bigrams and trigrams, in order to get benefits from all of them. In practice, it was proven to have really good results.

There are two kinds of interpolation :

a) Simple linear interpolation :

$$\hat{P}(w_i|w_{i-1}, w_{i-2}) = \lambda_1 P(w_i|w_{i-1}, w_{i-2}) + \lambda_2 P(w_i|w_{i-1}) + \lambda_3 P(w_i) \quad (2.5)$$

Where $\sum_i \lambda_i = 1$, for them to be probabilities. The task is simply to compute the probability of a word, when given the previous two, by interpolating the three models.

b) Lambdas conditional on context :

$$\begin{aligned} \hat{P}(w_i|w_{i-1}, w_{i-2}) &= \lambda_1(w_{i-2}^{i-1}) P(w_i|w_{i-1}, w_{i-2}) + \\ &\quad \lambda_2(w_{i-2}^{i-1}) P(w_i|w_{i-1}) + \\ &\quad \lambda_3(w_{i-2}^{i-1}) P(w_i) \end{aligned} \quad (2.6)$$

This method also mixes the three models, but λ s are here dependant on what the previous words were. This translates in the possibility to train a richer and more complex context conditioning for deciding how to mix the trigrams, the bigrams and the unigrams.[4]

The next encountered step is to set lambdas and this is done by using a held-out corpus.

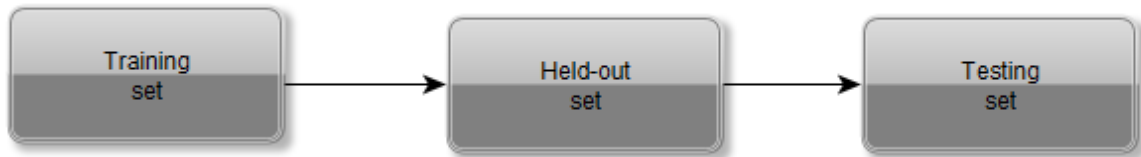


Figure 2.2 Necessary sets for training a language model

Lambdas are chosen to maximize the likelihood of held-out data. The first step is to train some n-grams, using the training set. Then look after λ s to use to interpolate those n-grams such that to give the highest probability of this held-out set.

So far, the case of switching from a bigram to a unigram has been approached, when the bigram has few or even 0 appearances. But what about the case when the actual word does not appear at all in the training set? Here can be discussed two situations. The first one is the case of a command menu. It is characterized by a fixed vocabulary V , and no other words can be said, except the ones included in the menu. This is called a closed vocabulary task and shall be discussed in more detail in a Section 2.3. The second situation deals with unseen words in the training set, called out of vocabulary words or OOV. This task is known as open vocabulary task and consequently, these words cannot be predicted by the language model.

In such situations, firstly we create a special token $\langle \text{UNK} \rangle$ (i.e. "unknown") and a fixed lexicon L of size v . At text normalization phase, we take the most unimportant words, with the lowest

probabilities and change them to <UNK>. Next step is to train the probabilities of UNK like a normal word.

So, instead of having in the training set : $W, W, W, w, W..$ where w is a really low probability word, we will have : $W, W, W, UNK, W..$ In the decoding process, if a word appears which has not been seen before, that word is replaced with UNK and its bigram and trigram probabilities are given from the UNK word in the training set.

2.2.2.2 Smoothing methods

The main idea of these methods is that they extract a part of the probability assigned to n-grams seen in the training phase, and redistribute it to unknown n-grams. As a result, they tend to make distributions more uniform, by adjusting low probabilities such as zero probabilities upward, and high probabilities upward. Smoothing methods have proven to be very effective, since they attempt to improve the accuracy of the model as a whole. Whenever a probability is estimated from a small number of occurrences, smoothing has the potential to significantly improve the estimation so that it has a better generalization capability.

The Good-Turning method deals with infrequent n-grams. The basic idea is to look after how many times the *n-grams* appear in the training data. On this basis, divide the *n-grams* into groups, depending on their frequency, such that the parameter can be smoothed based on the n-gram frequency.

To wrap this up, if a *n-gram* occurs r times, we should pretend that it occurs r^* times :

$$r^* = (r + 1) \frac{n_{r+1}}{n_r} \quad (2.7)$$

where n_r represents the number of *n-grams* that appear exactly r times in the training data. In order to convert it to probability :

$$P(a) = \frac{r^*}{N} \quad (2.8)$$

where $N = \sum_{r=0}^{\infty} n_r r^* = \sum_{r=0}^{\infty} (r + 1) n_{r+1} = \sum_{r=0}^{\infty} n_r r$, so N is equal with the number of counts in the distribution.[2]

The Good-Turing method is not very reliable for large values of r , for which n_r is typically 0. This drawback can be overcome by leaving aside the counts for frequent *n-grams*.

2.2.3 Evaluating the performance of the language model

A good language model is a model that assigns a higher probability to „real” or „frequently observed” sentences than to „ungrammatical” or „rarely observed” sentences. The process starts with training the parameters of the model on a training set and then, test the model’s performance on unseen data. In order to be a fair evaluation, this data should be really different from the training data. An evaluation metric is used to see how well the model does on the test data.

2.2.3.1 Perplexity

The best evaluation for comparing two models, for example A and B, is to put each model in a task, run the task, and get an accuracy for A and B. In the end, the only thing left to do is to compare the accuracy for A and B. This is called extrinsic evaluation of *n-gram* models(in-vivo). The drawback of this kind of evaluation is that it is time-consuming, can take days or even weeks.[5]

Instead, one can use intrinsic evaluation, that is perplexity. The *perplexity* of the test data is the most widely-used metric to evaluate the performance of *n-gram* smoothing. In information theory, the perplexity is a measurement of how well a probability distribution or probability model predicts a sample. So, the intuition of perplexity comes down to the simple question : How well can the model predict the next word in a sentence? The best language model is one that best predicts an unseen test set.

Translated in a mathematical language, perplexity is the probability of the test set, normalized by the number of words. If we consider the sentence has N-words:

$$PP(W) = P(w_1, w_2, \dots, w_N)^{\frac{-1}{N}} = \sqrt[N]{\frac{1}{P(w_1, w_2, \dots, w_N)}} \quad (2.9)$$

Where $P(w_1, w_2, \dots, w_N)$ is the probability of a string of words. The longer the sentence, the less probable it is going to be. Another mathematical expression of the perplexity is obtained through the chain rule :

$$PP(W) = \sqrt[N]{\prod_{i=1}^N \frac{1}{P(w_i | w_1, w_2, \dots, w_{i-1})}} \quad (2.10)$$

A particular case of Equation 2.10 for bigrams has the following expression:

$$PP(W) = \sqrt[N]{\prod_{i=1}^N \frac{1}{P(w_i | w_{i-1})}} \quad (2.11)$$

Because of this inversion, minimizing perplexity is the same as maximizing probability.[6] To conclude this section, there is a strong correlation between the test-set perplexity and the word error rate. Smoothing algorithms leading to lower perplexity generally result in a lower error rate.[2]

2.2.3.2 Out Of Vocabulary words

In the case of unseen words in the training set, the evaluation of the performance of the language model is very difficult. As mentioned previously, these words are known as *out of vocabulary* OOV and cannot be predicted by the language model. The perplexity of such words is infinity and thus,

cannot be added to the perplexity of the other *n*-grams. Thus, the perplexity of the entire word sequence cannot be computed. In order to fully evaluate the performance of a language model, one must specify both the perplexity and the OOV.

$$OOV[\%] = \frac{\#OOVs}{\#words} \times 100 \quad (2.12)$$

2.2.3.3 N-gram hits

N-gram hits represents another method to evaluate how good can a language model predict a word. As it was discussed previously, sometimes, if a trigram does not seem trust worthy, it is better to back-off, and use a bigram. Moving on, a bigram could back off, due to insufficient data, to a unigram. In the case of a trigram model, this metric gives the percentage of how many times the model could use the full two-preceding words history over how many times had the model to back-off to find the probability for the current n-gram :

$$trigram\ hits[\%] = \frac{\#trigram\ hits}{\#words} \times 100 \quad (2.13)$$

This metric has proven to be very useful when comparing different domain-specific language models.

2.3 FSG grammar

For the systems that deal with recognition of simple commands and control, it is more convenient to describe the user language by a grammar model. A finite state grammar (FSG) is a graph model in which the nodes correspond to the vocabulary words, and the transitions between the words are represented through the links of the graph. If the task is relatively small (digits recognition, phone dial, etc.) than this type of language model can be successfully used. Moreover, finite state grammar can be successfully used in word spotting applications.

This model explicitly describes all possible word sequences allowed by the grammar of the recognition task. Moreover, a cost can be attached to each link to specify the probability of finding that word preceded by another word. A grammar is composed of a set of rules that together define what may be spoken. This type of grammar can be successfully used when the vocabulary is only a few thousands or hundreds words wide.

2.4 Phonetic modeling

In the context of state-of-the-art continuous speech recognition systems, the acoustic models do not model the words of the source language in a direct manner, but in an indirect one. For Large-Vocabulary Continuous Speech Recognition Systems(LVCSR), where large-vocabulary generally means that the systems have roughly 5,000 to 60,000 words, it is difficult to build whole-word models because :

- ❖ There are simply too many words, with different acoustic representations and it is unlikely to have sufficient occurrences of these words in the training set to build context-dependant models.
- ❖ Every new ASR task comes with new specific words, without any available training data, such as newly invented jargons and proper nouns.[7]

The term continuous refers to the fact that the words are run together naturally, and not isolated, where each word would be preceded and followed by a pause.

The purpose of the phonetic model is to link the acoustic model, which estimates the acoustic probabilities of the *phonemes*, to the language model, which estimates the probability of sequences of *words*. The phonetic analysis component converts the processed text into the corresponding phonetic sequence.[8] In other words, the phonetic dictionary is a linguistic instrument, which makes the correspondence between the written form and the phonetic form of the words in the source language. This is followed by a prosodic analysis to attach the corresponding pitch and duration information to the phonetic sequence. In linguistics, prosody is the stress, the rhythm and the intonation of speech. Prosody may indicate several features of the speaker or the utterance, like the emotional state of the talker, or the form of the utterance (statement, command or question) or the presence of irony or sarcasm and many other elements of language that cannot be encoded by grammar or by choice of vocabulary.[9]

The difficulty with which a phonetic dictionary is developed depends on the size of the vocabulary. Even though a manually created dictionary would guarantee a perfect phonetisation, this task might prove to be extremely time-consuming when designing a large-vocabulary speech recognition system. Moreover, it would require a good command of the respective language.[1]

2.5 Acoustic modeling

Acoustic models refer mainly to the representation of knowledge about phonetics, acoustics, different pronunciations, gender and dialect differences among the speakers, environment variability and so on. A speech recognition system which can be applied to a vast number of talkers, without the need to be trained individually on every one, is called a *speaker-independent* system. Such a system is based on some clustering algorithms with the final goal of creating word and sound reference patterns, which can be used across large range of speakers and accents. In the early stages, these patterns were characterized by a more intuitive template-based approach, but gradually evolved in more rigorous statistical models.[3]

The popularity and use of the Hidden Markov Model as the main foundation for automatic speech recognition has remained constant over the past two decades. HMM is today the preferred method for speech recognition mainly because of the steady stream of improvements of the technology. Another reason why HMMs are popular is because they can be trained automatically and are simple to use.

Hidden Markov model (HMM) can provide an efficient way to build trust worthy parametric models and also incorporate the dynamic programming principle in its core for unified pattern segmentation and pattern classification of time-varying data sequences. The underlying assumption of the HMM is that the data samples can be well characterized as a parametric random process, and the parameters of the stochastic process can be estimated in a precise and well-defined framework. The HMM has become one of the most powerful statistical methods for modeling speech signals. Its

principles have been successfully used in automatic speech recognition, formant and pitch tracking, spoken language understanding and machine translation.

2.5.1 Feature extraction

Since HMMs do not model directly the waveform of the acoustic signal, in this section it will be discussed a kind of acoustic processing commonly called feature extraction or signal analysis in speech recognition literature. The term features refers to the vector of numbers which represent one time-slice of speech signal. Most commonly used kinds of features are LPC features, PLP features and MFCC features. They are called spectral features because they represent the waveform in terms of the distribution of different frequencies which make up the waveform.[10]

Speech parameters are often processed by filters. The most common filtering occurs at the spectral level, where the power spectrum is processed through filter bank channels. The MFCC features use the Mel-scale filter bank while the PLP features utilize the Bark scale for its critical band analysis.

The first feature we use is the speech waveform itself. The fact that humans, and to some extent machines, are capable of transcribing and understanding speech just given the sound wave leads to the conclusion that the waveform contains enough information to make this task possible. Sometimes, this information is hard to unlock just by looking at the waveform, but even so a visual inspection is sufficient to retrieve some relevant characteristics. For instance, the difference between vowels and some consonants is relatively clear on a waveform. Vowels are characterized by an open configuration of the vocal tract so there is no build-up of air pressure above the glottis. This contrasts with consonants, which are characterized by a constriction or closure at one or more points along the vocal tract. This translates in a visible difference of energy. Researchers are able to look at the spectrogram and identify several vowels or consonants on account of their amplitude. In Figure 2.3 is a Matlab energy figure of vowels (“a”, “e”, “i”, “o”, “u”). Between them, the areas where the energy is almost zero, are the moments of silence, when the speaker pauses before moving on to the next vowel.

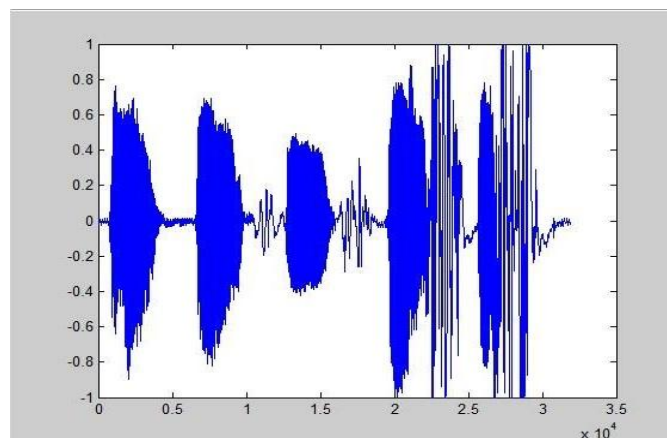


Figure 2.3 Matlab figure illustrating the amplitude of vowels

Moving on, since the speech signal is not a stationary one, the spectral analysis cannot be done on the entire signal, but on short frames(20-30ms), on which the signal is quasi-stationary. The original

signal is segmented in the time domain, using a Hamming window, and the feature extraction process is performed on every single window. In general, time-domain features are much less accurate than frequency-domain features such as the mel-frequency cepstral coefficient(MFCC). This is said because many features such as formants, useful in discriminating vowels, are better characterized in the frequency domain. When computing the MFCC coefficients, a non-linear frequency scale is used, since it better approximates the human hearing system. This analysis process takes into account that the seizing of different sound tones is done on a logarithmic scale inside the ear, proportional with the fundamental frequency of the sound. In this manner, the human ear response is non-linear with respect to frequency, since it is able to sense small frequency differences among the low frequency components easier than among the high ones.

In order to determine the cepstral coefficients, the spectrum, computed using FFT, is smoothened through some triangular filter banks, each centered on a frequency found on the Mel scale. The Mel scale is a perceptual scale of pitches built according to some listeners which are equal in distance from one another.[11] The purpose of this set of triangular filters is that of splitting the signal over the frequency bandwidths associated with the Mel scale. For a vocal signal with a bandwidth of 8kHz, a number of 24 filter sets is considered sufficient to compute the MFCC parameters. Nevertheless, in speech recognition systems this number is configurable and through experiments, one can find its optimum value for the respective application. By applying the logarithmic compression at the output of the set of filters, the distribution of the coefficients follow a Gaussian law. Then, over each band it is computed the mean energy. The MFCC coefficients are obtained after applying the Discrete Cosine Transform, which is a very convenient instrument. It deals only with real numbers, it has a strong „energy compaction” property : most of the signal information tends to be concentrated in a few low-frequency components and decorrelates these values.[12]

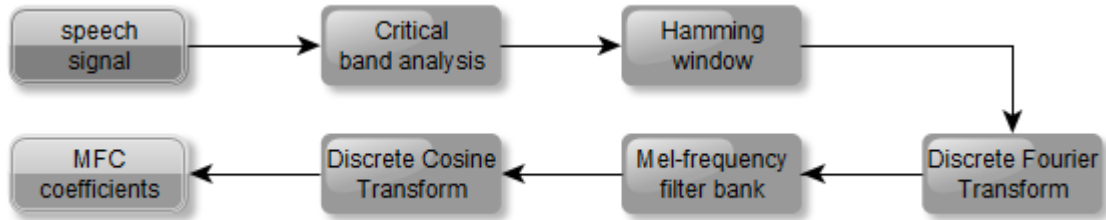


Figure 2.4 The stages for extracting the MFC coefficients

In other words, in order to obtain the MFCC coefficients:

- ❖ First the Fourier transform of (a windowed excerpt of) a signal is computed:

$$X_a[k] = \sum_{n=0}^{N-1} x[n] e^{\frac{-2j\pi n k}{N}}, 0 \leq k \leq N \quad (2.14)$$

- ❖ The set of M ($m = 1, 2, \dots, M$) triangular overlapping windows is defined, so that to map the powers of the spectrum obtained above onto the mel scale:

$$H_m[k] = \begin{cases} 0, & k < f[m-1] \text{ or } k > f[m+1] \\ \frac{2(k - f[m-1])}{(f[m] - f[m-1])(f[m+1] - f[m])}, & f[m-1] < k < f[m] \\ \frac{(f[m+1] - k)}{(f[m+1] - f[m])(f[m] - f[m-1])}, & f[m] < k < f[m+1] \end{cases} \quad (2.15)$$

The formula can be, also, expressed like this:

$$H'_m[k] = \begin{cases} 0, & k < f[m-1] \text{ or } k > f[m+1] \\ \frac{(k - f[m-1])}{(f[m] - f[m-1])}, & f[m-1] < k < f[m] \\ \frac{(f[m+1] - k)}{(f[m+1] - f[m])}, & f[m] < k < f[m+1] \end{cases} \quad (2.16)$$

In this case $\sum_{m=1}^M H'_m[k] = 1$. The only thing that differs between the two representations is a vector of constants for all the input signals, so as long as the same filter is used among everywhere, the choice of which one is applied is unimportant.[12]

This set of filters computes the spectrum around the central frequency of each band. Their band increases along the index m . [13]

The first filter bank will start at the first point, reach its peak at the second point, then return to zero at the third point. The second filter bank will start at the second point, reach its peak at the third point, then be zero at the fourth point and so on. The final plot of the M filters overlaid on each other looks like this:

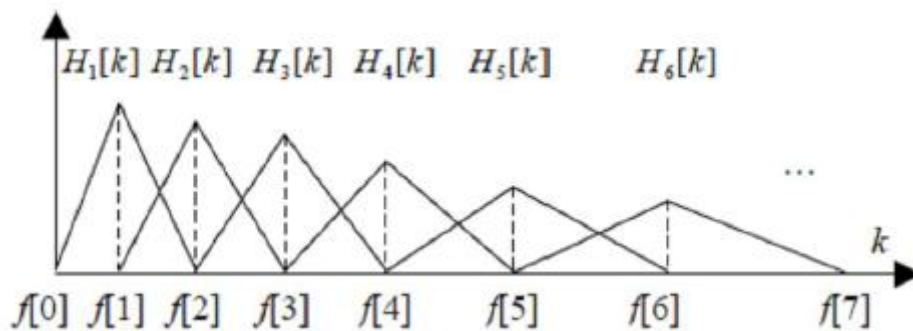


Figure 2.5 Frequency bands on Mel scale[12]

Let us consider that : f_l and f_h are the lowest, respectively the highest frequency in the filter bank, expressed in Hz, F_s is the sampling frequency, expressed as well in Hz, M the number of filters and N the size of the FFT window. The boundary point $f[m]$ are placed uniformly, along the Mel scale:

$$f[m] = \left(\frac{N}{F_s}\right) B^{-1}\left(B(f) + \frac{B(f_h) - B(f_l)}{M + 1}\right) \quad (2.17)$$

Where Mel scale B is :

$$B(f) = 1125 \ln \left(1 + \frac{f}{700}\right) \quad (2.18)$$

And its inverse, B^{-1} is :

$$f = 700 \left(\exp\left(\frac{b}{1125}\right) - 1\right) \quad (2.19)$$

- ❖ Next, the logs of the powers at each mel frequency are taken.

$$S[m] = \ln \left[\sum_{k=0}^{N-1} |X_a[k]|^2 H_m[k] \right], 0 \leq m \leq M \quad (2.20)$$

- ❖ The DCT is applied on the list of M Mel log powers, as if it were a signal

$$c[n] = \sum_{m=0}^{M-1} S[m] \cos\left(\frac{n\pi(m + 1/2)}{M}\right), 0 \leq n \leq M \quad (2.21)$$

- ❖ The MFCCs are the amplitudes of the resulting spectrum. Only the 2-13 DCT coefficients are taken, the rest being discarded.[13]

Temporal changes in the spectra play an important role in human perception. Even though each set of coefficients is computed over a short Hamming window, the information contained by the temporal dynamics of these parameters is very useful in automatic speech recognition. One way to capture this information is by using *delta coefficients*, that measure the change in coefficients over time. They are also known as differential and acceleration coefficients. It turns out that computing the MFCC trajectories and appending them to the original feature vector would significantly increase the performance of the automatic speech recognition system. Temporal information is particularly complementary to HMMs, since HMMs assume each frame is independent of the past, in contrast with these dynamic features that broaden the scope of the frame.[7]

When 16-kHz sampling rate is used, a typical state-of-art speech system can be build based on the following features:

- 13th order MFCC c_k
- 13th order 40-msec 1st - order delta MFCC computed from $\Delta c_k = c_{k+2} - c_{k-2}$
- 13th order 40-msec 2nd - order delta MFCC computed from $\Delta\Delta c_k = c_{k+1} - c_{k-1}$

The short-time analysis Hamming window of 256 ms is typically used to compute the MFCC .The $c_k[0]$ is included in the feature vector. In conclusion, the feature vector used for speech recognition is generally a combination of these features :

The short-time analysis Hamming window of 256 ms is typically used to compute the MFCC .The $c_k[0]$ is included in the feature vector. In conclusion, the feature vector used for speech recognition is generally a combination of these features :

$$x_k = \begin{pmatrix} c_k \\ \Delta c_k \\ \Delta\Delta c_k \end{pmatrix} \quad (2.22)$$

and have proven to give very good results. It is formed of 39 coefficients : 12 MFCC + energy, together with their first and second order temporal derivatives.

2.5.2 HMM framework

HMMs are very popular in speech recognition domain, mainly because of the advantages they offer. Compared to simple Markov models, in the case of HMMs there is no bijection between the state and the output. This offers a greater flexibility and it matches perfectly the speech signal, in which the same phoneme can have different durations depending on the case. A hidden Markov model is a stochastic process, which models the intrinsic variability of the speech signal and the structure of the spoken language in a consistent statistical modeling framework. HMMs are probabilistic finite state machines, which can be combined to obtain word sequence models out of smaller units. In the task of large-vocabulary speech recognition, sequences of words are built hierarchically from word models, which in turn are built from sub-word models with the help of a pronunciation dictionary. For good recognition results, these sub-word models have to be context-dependent phone models.[1]

Through its nature, a speech signal is significantly variable, due to variations of pronunciation or environmental factors. When the same word is said by several speakers, the acoustic signals may be amazingly different, even though the underlying linguistic structure may be the same. HMM uses a Markov chain to establish the linguistic structure and a set of probability distributions to score the variability in the acoustic realization of the sounds in the utterance. Given a sufficient collection of the variations of the words of interest, one can obtain the most „suited” set of parameters that define the corresponding model or models, through an efficient estimation method, known as Baum-Welch algorithm. This estimation of parameters can be translated through training and learning of the system. In the end, the resulted model should be able to indicate whether an unknown utterance is indeed a realization of the word represented by the model.[3]

The hidden Markov model introduces a non-deterministic process that generates output observation symbols in any give state. Thus, the observation is a probabilistic function of the state. It can be viewed as a double-embedded stochastic process with an underlying stochastic process (the state sequence) not directly observable. This underlying process can only be probabilistically associated with another observable stochastic process producing the sequence of features we can observe. A

hidden Markov model is basically a Markov chain where the output observation is a random variable X generated according to a output probabilistic function associated with each state.[14]

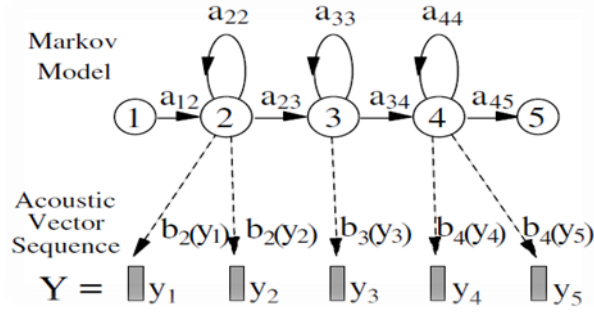


Figure 2.6 HMM-based phone model with 5 states[15]

The entry and the exit states are non-emitting. These are included to simplify the process of concatenating phone models to make words. Although the definition of an HMM allows the transition from any state to another state, in speech recognition the models are created in such a manner to disallow arbitrary transition. Due to the sequential nature of speech, there are placed strong constraints on transitions backward or on skipping transitions. Self-loops allow a sub-phonetic unit to repeat so as to cover a variable amount of the acoustic input.[1] Formally speaking, a hidden Markov model is defined by :

- ❖ $Y = \{y_1, y_2, \dots, y_M\}$ - an output observation alphabet. The observation symbols correspond to the physical output of the system being modeled.
- ❖ $\Omega = \{1, 2, \dots, N\}$ - a set of states representing the state space.
- ❖ $A = \{a_{ij}\}$ - a transition probability matrix, where a_{ij} is the probability of taking a transition from state i to state j :

$$a_{ij} = P(s_t = j | s_{t-1} = i) \quad (2.24)$$

- ❖ $B = \{b_i(k)\}$ - an output probability matrix, where $b_i(k)$ is the probability of emitting symbol y_k when state i is entered. Let $X = X_1, X_2, \dots, X_t, \dots$ be the observed output of the HMM. The state sequence $S = s_1, s_2, \dots, s_t, \dots$ is not observed (hidden), and $b_i(k)$ can be rewritten as follows :

$$b_i(k) = P(X_t = y_k | s_t = i) \quad (2.25)$$

- ❖ $\pi = \{\pi_i\}$ - a initial state distribution where :

$$\pi_i = P(s_0 = i), \quad 1 \leq i \leq N \quad (2.26)$$

Since a_{ij} , $b_i(k)$ and π_i are probabilities, they must satisfy the necessary properties :

$$a_{ij} \geq 0, b_i(k) \geq 0, \pi_i \geq 0 \quad \forall \text{ all } i, j, k \quad (2.27)$$

$$\sum_{j=1}^N a_{ij} = 1 \quad (2.28)$$

$$\sum_{k=1}^M b_i(k) = 1 \quad (2.29)$$

$$\sum_{i=1}^N \pi_i = 1 \quad (2.30)$$

The acoustic model parameters $\lambda = \{a_{ij}, b_i()\}$ are efficiently estimated from a corpus of training utterances using the forward-backward algorithm, which is an example of expectation-maximization.

In conclusion, the complete description of a HMM includes two-size parameters, N and M , representing the total number of states and the size of observation alphabets, observation alphabet Y , and three matrices of probability measures $\mathbf{A}, \mathbf{B}, \boldsymbol{\pi}$. The following notation :

$$\Phi = (\mathbf{A}, \mathbf{B}, \boldsymbol{\pi}) \quad (2.31)$$

is used to indicate the whole parameter set of an HMM.

Given the above definition of HMMs, the three basic problems can be formulated now before they can be applied to real-world applications:

- A. **The Evaluation Problem** : Given a model Φ and a sequence of observations $\mathbf{Y} = (Y_1, Y_2, \dots, Y_T)$ what is the probability $P(\mathbf{Y}|\Phi)$ that this sequence \mathbf{Y} to have been generated by the model Φ ?
- B. **The Decoding Problem** : Given a model Φ and a sequence of observations $\mathbf{Y} = (Y_1, Y_2, \dots, Y_T)$, what is the most likely state sequence $\mathbf{S} = (s_0, s_1, \dots, s_T)$ in the model that produces the observations?
- C. **The Learning Problem** – Given a model Φ and a set of observations, how can we adjust the Φ model parameters to maximize the joint probability(likelihood) $\prod_x P(\mathbf{X}|\Phi)$?

By solving the *evaluation problem*, we are able to evaluate how well a given HMM matches a given observation sequence. Therefore, HMM is used to do pattern recognition, since the likelihood $P(\mathbf{X}|\Phi)$ can be used to compute posterior probability $P(\Phi|\mathbf{X})$, and the HMM with the highest posterior probability is determined as the desired pattern for the observation sequence. By solving the *decoding problem*, we can find the best matching state sequence given an observation sequence, or in other words, we can uncover the “hidden” state sequences. And by solving the *learning problem*, we will have the means to automatically estimate the model parameter Φ from the training

set.[14] The hardest task is the learning one, since from the training data one must estimate the HMM's parameters such that they can characterize the chosen speech unit.

The vocal signal is split in elementary units, like : words, phonemes, tri-phonemes. To each unit, a HMM is associated, and to each state of a HMM a time window, with the voice parameters computed for this specific window. During speech, the vocal tract passes through a sequence of states (which are modeled with the HMM states) and in each state a segment from the vocal speech is emitted with a vector of parameters which constitute the output of the HMM's state.

This output vector of the HMM must take continuous values, since the voice parameters take values in a continuous space. For this reason, Gaussian mixtures are used to model the observations of the HMMs' states. Each parameter of the output vector can be modeled through a weight sum of functions with normal distributions :

$$b_j(y_t) = \sum_{g=1}^G w_{jg} N(y_t, \mu_{jg}, \Sigma_{jg}) \quad (2.32)$$

where :

$$N(Y, \mu_i, \Sigma_i) = \frac{1}{\sqrt{(2\pi)^n |\Sigma|}} e^{-\frac{1}{2}(Y-\mu_i)^T \Sigma^{-1} (Y-\mu_i)} \quad (2.33)$$

Where $Y=[y_1, y_2, \dots, y_n]$ is the n -dimensional observation vector, n is the number of the voice parameters which are extracted from the observations, $|\Sigma|$ is the covariance's diagonal matrix determinant, G is the number of components of the mixture and w_{jg} is the weight of the component g of the state j . [12]

Modeling speech using hidden Markov models makes two assumptions :

- ❖ 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, so that means the history of previous states is not necessary.
- ❖ Observation independence : observations are conditionally independent of all other observations given the state that generated it.[1]

These two assumptions may lead to an unrealistic model of speech, but they are needed due to the mathematical and computational simplifications they bring. The estimation and decoding problems would be very difficult to be addressed without these two assumptions. Nevertheless, the last two decades of HMMs success in speech signal modeling prove that these „limitations” are not significant.

2.5.3 Choosing the basic unit

In other words, the task of choosing an appropriate modeling unit is not as simple as it may appear at a first glance. In order to design a workable system, there are some important issues to be taken into consideration when selecting the most basic units :

- ❖ The unit should be *accurate*, to represent the acoustic realization that appears in different contexts.
- ❖ The unit should be *trainable*. To estimate the parameters of the unit, there should be enough available data. Here it is pointed out again why words are the least trainable choice in

building a recognition system, since, despite their accuracy, it is almost impossible to get several hundred repetitions for all the words. Words are a proper choice of basic units only in the cases when speech recognition is domain specific i.e. for digits only.

- ❖ The unit should be *generalizable*, so that any new word may be derived from a predefined unit archives for task-independent speech recognition. If this record would consist in a fixed set of word models, there would be no possible way to derive the new word model.[7]

A practical challenge is how to balance these three important criteria. Thus, instead of modeling words, large-vocabulary recognition systems use sub-words as basic speech units, such as phones, since words are neither trainable, nor generalizable.

Phonetic models provide no training problem, since sufficient occurrences for all phones can be found in just a couple of thousand phrases. They can be trained on one task and tested on another because they are vocabulary independent. These make phones trainable and generalizable. However, this phonetic model assumes that a phoneme is identical in any context and any word is obtained by concatenating independent phones. This is not the case, because phonemes are not produced independently and the realisation of a phoneme strongly depends by its immediately neighboring phonemes. To sum up, these phonetic models lead to less accurate models.

This drawback can be overcome if we consider context dependent units. If we have a large enough training set to estimate these context-dependent parameters, we could significantly improve the recognition accuracy. Here we introduce the notion of triphone model, a phonetic model that takes into consideration both the left and the right neighboring phones. If two phones have the same identity but different left or right context, they are considered different triphones. The different realizations of a phoneme are denoted with the term allophone.[1]

Triphone models are very powerful phonetic models and they are more consistent than context-independent units, but in this case the training becomes a challenging task. Since every triphone context is different, the main idea is to find instances of similar contexts and merge them, so that to obtain a manageable number of models that can be better trained.[1]

Moving one step further, with the purpose of balancing trainability and accuracy between phonetic and word models, the modeling of sub-phonetic events is observed. For sub-phonetic modeling, we can treat the state in phonetic HMMs as the basic sub-phonetic unit. In this context, the concept of clustering hidden Markov models has been proposed and generalized to the state-dependent output distributions across different phonetic models. Each cluster represents a set of similar Markov states and is called a senone. A sub-word is thus composed of a sequence of senones after the clustering process is finished. The optimal number of senones for a system is mainly determined by the available training set and can be tuned.

2.6 ASR Evaluation

As a sanity check, it is better to use a small sample from the training data to measure the performance of the training set. Training-set performance is useful in the development stage to identify potential implementation bugs. Eventually, the tests must be done on a development set that typically consists of data never used in training.

A way to evaluate the performance of the language model is to evaluate the word error rate(WER) yielded when placed in a recognition system. The WER is a very convenient tool used when comparing different language models, as well as for evaluating improvements within one system. The general difficulty when using this method lies in the fact that the recognized word sequence can

have a different length from the reference sequence, which is supposed to be correct. These two sequences of words are aligned through a algorithm which has as final goal minimizing the cost of editing the recognized sentence, so that to look alike the reference one. The WER can be computed as :[16]

$$WER[\%] = \frac{\#Insertions + \#Substitutions + \#Deletions}{\#Number\ of\ words\ in\ the\ ref.\ transcription} \times 100 \quad (2.34)$$

There are typically three types of word recognition errors in speech recognition :

- *Substitution* : an incorrect word was substituted for the correct word
- *Deletion* : a correct word was omitted in the recognized sentence
- *Insertion* : an extra word was added in the recognized sentence.

This kind of evaluating the performance provides, however, no specific details regarding the nature of the translation errors and further work is required in order to find the source of the error. Moreover, this kind of measurement does not keep count that a substitution error could be easily removed if the number of erroneous characters is small (like „look”, ”looks”), or difficult if the number of erroneous characters would be high(like „maintain”, ”sustain”). In order to overcome this drawback, sometimes it is used an evaluation done at the character level.

$$ChER[\%] = \frac{\#Ch.\ Insertions + \#Ch.\ Substitution + \#Ch.\ Deletion}{\#Number\ of\ characters\ in\ the\ ref.\ transcription} \times 100 \quad (2.35)$$

The last method of evaluation is done at the sentence level, and is useful only in the cases when the erroneous transcription of a single word in a word sequence makes the recognition useless.

$$SER[\%] = \frac{\#No.\ of\ erroneous\ sentences}{\#Sentances\ in\ the\ ref.\ transcription} \times 100 \quad (2.36)$$

From these three performance criteria used in evaluating a recognition system, the most commonly used is WER.

2.7 Speech Recognition Tools

For this project I have used CMU Sphinx. It is an open source toolkit and it is available online. CMU Sphinx system successfully integrated the statistical method of hidden Markov models and hence, it was able to train and embed context-dependent phone models in a sophisticated lexical decoding network. [3] It is a very popular and commonly used speech recognition tool because it offers the possibility of developing speaker-independent, large-vocabulary, continuous speech recognition systems with remarkable results.[1] This tool also presents summaries of the most inserted/deleted/substituted words and can compute the sentence/word error rates in a per speaker manner.

3 CHAPTER Automatic Speech Recognition Systems

3.1 Current stage for Automatic Recognition System for Albanian

One of my targeted languages was Albanian, a language with poor resources. These languages are spoken by a large number of people, but so far too few acoustic resources (speech data bases) and linguistic resources (text corpuses) were acquired in order to develop an unconstrained continuous speech recognition system.

The Baum-Welch training paradigm requires speech audio clips along with their textual transcriptions in order to estimate the models parameters. Thus, speech databases are critical resources along with their characteristics, such like the number of hours of speech, number of speakers, etc, in developing a speech recognition system.

As previously remarked, Albanese has no speech resources available, neither freely, nor commercially. Moreover, the speaker-independency desiderate implies resources from a large number of speakers. The inter-speaker speech variability is an important factor and can be overcome by completely and accurately modeling the various possible pronunciations of every phone. This can, in turn, be achieved by using recordings from a vast number of speakers.

3.2 Speech database acquisition for Albanian ASR

A complete speech database is formed from :

- ❖ a set of speech signal samples.
- ❖ a set of transcriptions, which must be perfectly synchronized with what is spoken in each speech sample.
- ❖ additional information regarding speech type (isolated words, continuous, spontaneous).

Since direct recording was not a possible solution, we have started to build speech databases by extracting fragments from website news. These audio clips also had correspondent transcriptions, which, in most of the cases, were related. We had access at 4 news databases : www.balkanweb.tv, www.topchannel.tv, www.topchannel2.tv and www.vizionPlus.tv. Speed gave us access to each database's .php files. By processing the .php files specific for each database, we searched for the URL in each file and created two lists : one containing the fileids of the files, and the other one the correspondent link.

3.2.1 Acquisition tools for audio clips for Albanian ASR

With the help of a Java Program, we processed all the files having the php extension. We looked after the pattern „www.youtube.com” and extracted the substring corresponding to the URL in a list. We also checked if the URLs were still available, by removing the URLs which returned the code „404” to the ”checksURL” method. After procesing all the files in the four databases, we had as output four lists in the format : ”ID : URL”.

The next step was to separate each list in two separate lists : one with just one column, containing all the fileids, and the other one with the corresponding youtube links. We chose to do this with a script, since it was faster. We used ”sed”[17], which is a stream editor.

Problems encountered at this step :

- For 50 files in www.topchannel2.tv database it worked perfectly, but when we tested it on 1000 files, there appeared more URLs than IDs. The problem was that some of the .php files contained the same links. We resolved this error with the help of a Linux command which sorted uniquely the elements after the second column, that of the URLs.

So far, for each database we created two lists : one with the IDs corresponding to the .php file names and one with the URLs found in those .php files.

The next step was to download the content of those audio clips, and then convert them in files with “wav” extension, as required by Sphinx. To do this, we used ffmpeg tool.[18] All the audio clips in the databases share the same sampling frequency (16kHz) and the same sample size (16 bits).

During acquisition of these databases, one significant issue gained our attention. Some audio files had to be split into smaller samples (5 seconds to 25 seconds, as CMU Sphinx suggests). For this we used diarisation. Speaker diarisation is the process of partitioning an input audio stream into homogeneous segments according to the speaker identity. It is a combination of speaker segmentation and speaker clustering. The first one aims to find speaker change points in an audio stream, while the second one aims at grouping together speech segments on the basis of speaker characteristics.[19]Through speaker diarisation process we have managed to transform the audio clips from the website news’ databases into audio data ready to be further used in speech recognition.

3.2.2 Acquisition tools for transcriptions for Albanian ASR

As stated before, Albanian is a low-resourced language. That is a language spoken by a large number of people, but so far no prior work has been done to collect and/or organize resources for developing an automatic speech recognition system. Given the lack of availability of Albanian text corpora and the need of large amount of text to create a language model suitable for an automatic speech recognition system, one of our goals was to acquire this type of language resources. Our only solution, given also the lack of time, was to gather the resources from the website news’ databases and organize them in the purpose of making a trust worthy text corpora. For every downloaded file with the extension “wav”, we had to find the correspondent transcription file. The first step was to convert the files from the “php” extension to “txt” conversion. We changed the extension from “php” to “html”, and then we converted the files from “html” to “txt” format using lynx tool. [22] Lynx is a Web browser that only reads text. We preferred lynx because it parses the raw HTML. The difference from *wget*, another Web Browser, is that lynx will render the HTML (it hides all the tags, arranges the text etc). After this, the files had to be encoded in UTF-8. This was done using “iconv” command. [23] Iconv converts string to requested character encoding. The .php files were initially encoded with UCS-2 little endian, and in order to move further we needed them encoded with UTF-8. The second step was to parse the files, in order to bring them in the format required by Sphinx. As can be seen from Figure 3.1, the useful information, that is the actual news, was surrounded by a Header and a Footer. Luckily for us, all the files belonging to a website news’ database had the same Header and Footer. Moreover, the text contained special characters (like “, #, \$, etc), punctuation signs, uppercases and numbers and also had an undesired format. In this shape, the text was useless for Sphinx speech recognition toolkit, that is why these files had to be processed and brought in the required shape.

```

18      * [14]Vendi
19      * [15]Kosova
20      * [16]Rajoni
21      * [17]Bota
22      * [18]Sport
23      * [19]Show Biz
24      * [20]Lifestyle
25      * [21]Video
26      *
27      _____
28      [22][kerko.png]
29
30      Obama/Osama dhe "katrahura" e TV amerikane
31      03/05/2011 02:10
32      Obama/Osama dhe "katrahura" e TV amerikane "Obama bin Laden vdiq!".
33      Ndryshimi qendron vetëm tek një germë. Ngatërrimi mes "S" dhe "B",
34      deformon totalisht lajmin.
35      Në fakt një gjë e tillë ka ndodhur në Shtetet e Bashkuara me
36      televizionet, pas njoftimit të presidentit Obama për vdekjen e Bin
37      Laden.
38      Ngjashmëria rastësore e mbiemrit të shefit të Shtëpisë së Bardhë me
39      emrin e kryeterroristit të vrarë nga forcat speciale amerikane,
40      shkaktoi një rrëmujë të vërtetë në titrat e mediave amerikane.
41      Në shumë media titrat janë korigjuar menjëherë, por [23]kryevepra ishte
42      gafa e folësit të televizionit "KTXL-TV":
43      "Presidenti Obama njoftoi kombin dhe mbarë botën se presidenti Obama
44      vdiq", tha folësi i KTXL-TV. / Top Channel
45      [24]Facebook
46      Lajmet kryesore
47      [25]Obama/Osama dhe "katrahura" e TV amerikane [26]Rama mbledh qeverinë

```

Figure 3.1 Example of Albanian .txt file in raw form

We created a tool for the purpose of cleaning the text corpora, and bring .txt files in the needed format. This cleaning application is, mainly, written in the Java programming and can run on any operating system which has a Java Virtual Machine (JVM) installed. Besides Java, we also used a couple of Linux scripts to correct things that passed the Java filtering. The cleaning application takes a corpus as an input and, after applying several processing operations, it returns a text without any digits, punctuation marks or special characters. All the programs that we used work as a pipeline, meaning the output of one program is the input of the following program.

The first thing was to look for a specific header and footer for each database. For example, for the www.topchannel2.tv database, we noticed that in every raw file, the useful information, that is the actual news, was included between the header : “ [22][kerko.png]” and the footer : “ [24]Facebook”. With the help of a Java Program, we removed the text above the Header and below the Footer.

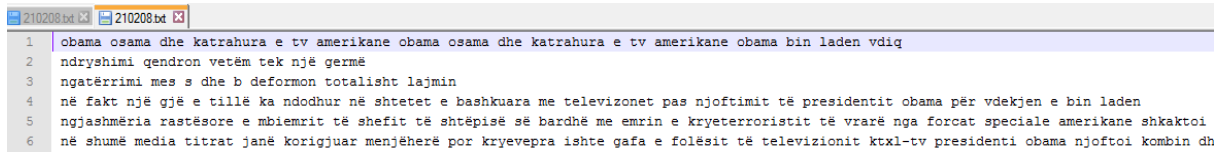
The next cleaning operation was to eliminate the new lines and also the lines which contained the word “IFRAME”, since it appeared after the Header, but it did not contain any useful information. One line represents, in fact, one sentence. I chose to do this with the help of a Linux script, since it was easier to implement.

The third cleaning operation deals with the punctuation marks and other special characters. In ASR we do not output punctuation marks, so we do not need to estimate their occurrence probability. Consequently, all punctuation marks have to be removed or properly replaced by a word sequence. For instance : a) dots, question marks and exclamation marks are replaced with a new line character (this way, we will have one sentence per line in the output file) , b) commas are removed, c)

brackets are deleted, d) characters like “\$” and “=” are replaced with their naming “dollars” and “i barabartë”.

The fourth cleaning operation was to remove the numbers. We chose this approach for this project, since we decided to focus all our energy in creating a wide Albanian database.

In the end, all letters were lowercased and the empty line were removed.



```

1 obama osama dhe katrahura e tv amerikane obama osama dhe katrahura e tv amerikane obama bin laden vdiq
2 ndryshimi qendron vetem tek nje germen
3 ngaterrimi mes s dhe b deformon totalisht lajmin
4 ne fakt nje gjë e tillë ka ndodhur ne shtetet e bashkuara me televizionet pas njoftimit te presidentit obama per vdekjen e bin laden
5 ngjashmeria rastësore e mbiemrit te shefit te shtepise se bardhe me emrin e kryeterroristit te vrare nga forcat speciale amerikane shkaktoi
6 ne shume media titrat jane korigjuar menjehere por kryevepra ishte gafa e folësit te televizionit ktxl-tv presidenti obama njoftoi kombin dh

```

Figure 3.2 Cleaned Albanian text

3.2.2.1 Diacritics

Albanese is a language that does not make intensive use of diacritics, but, nevertheless, it uses 2 (“ë” and “ç”). The occurrence frequency of “ë” is very high. Even though for a human reader the meaning of a text still makes sense in the absence of diacritics (given the paragraph context), the diacritics restoration task is not trivial for a computer. In order to simplify this operation, we chose to substitute “ë” with “ww” and “ç” with “cc” in the transcription files, since these combinations do not exist in Albanian language.

3.3 Building and acoustic, a language and a phonetic model for a small albanian database

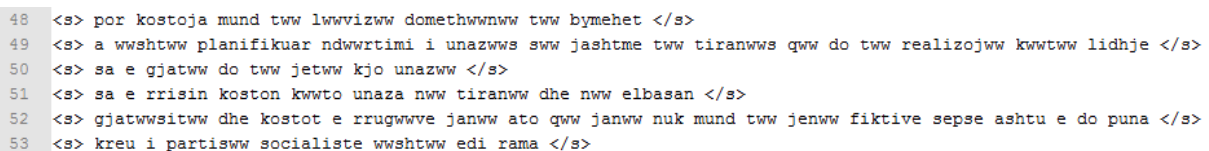
The next step consisted in creating the necessary models used in speech recognition from a small database (13 speakers, 2h), for which we had both the transcriptions and the corresponding ways.

database name	type	#no of phrases	#no of words	#no of unique
MediaEval2013	recordings	969	14063	1165

Table 3-1 MediaEval 2013 database

3.3.1 Building the language model

Language model is the representation of the grammar or syntax of the task. We used a Linux script which took as input a transcription file and returned as output the *counts file*, the *vocabulary file* and the sorted and sphinx format *language model file*. The language model toolkit expects its input to be in the form of normalized text files, with utterances delimited by <s> and </s> tags.



```

48 <s> por kostoja mund tww lwwvzww domethwnnw tww bymehet </s>
49 <s> a wwshtww planifikuar ndwvrtimi i unazww sww jashtme tww tiranwws qww do tww realizojww kwvww lidhje </s>
50 <s> sa e gjatww do tww jetww kjo unazww </s>
51 <s> sa e rrisin koston kwvto unaza nww tiranww dhe nww elbasan </s>
52 <s> gjatwwsitww dhe kostot e rrugwwve janww ato qww janww nuk mund tww jenww fiktive sepse ashtu e do puna </s>
53 <s> kreu i partisww socialiste wwshtww edi rama </s>

```

Figure 3.3 transcription file for language model

<s> : beginning-utterance silence

<sil> : within-utterance silence

</s> : end-utterance silence

Note that the words <s>, </s> and <sil> are treated as special words and are required to be present in the filler dictionary.

The *vocabulary file* contains a list of all the unigrams in the input file, while the *counts file* contains the number of occurrences of the unigrams, bigrams and trigrams.

More data will generate better language models. The *albanian1.transcription* contains 968 lines, but this is only the start.

3.3.2 Building the phonetic model

The phonetic model is a pronunciation dictionary that maps all the words in the vocabulary to a sequence of phonemes.. It is needed to link the acoustic model, which estimates phonemes acoustic likelihoods, to the language model, which estimates word sequence probabilities. The phonetic model works as an interface between the acoustic model which works with phonemes, and the language model, which works with words.

Developing a phonetic dictionary is a quite difficult task. Since a manually created dictionary would have required a good knowledge of the language and also a tedious work, we have preferred an automatically approach. Thus, the need for a graphemes-to-phonemes tool which could automatically create phonetic transcriptions for a given vocabulary is obvious. Our need for a graphemes-to-phonemes tool is not singular, since the task of automatically creating phonetic transcriptions for words in a vocabulary is very important in speech recognition and it has been approached by several researchers.

With the help of a Matlab program we obtained the phonetic dictionary.

3.3.2.1 Graphemes-to-phonemes method description

We adopted a SMT-based approach for the task of automatically creating the phonetic transcriptions. A Statistical machine translation (SMT) is a machine translation paradigm where translations are generated on the basis of statistical models whose parameters are derived from the analysis of bilingual test corpora.[20] A SMT system translates text in a source language into text in a target language. Two components are required for training :

- ❖ A parallel corpus consisting of sentences in the source language and their corresponding sentences in the target language
- ❖ A language model for the target language.[1]

First of all, a **grapheme** represents the smallest semantically distinguished unit in a written language, analogous to the phonemes of spoken languages. In this case, we consider graphemes (letters) as “words” in the source language and sequences of graphemes (words) as “sentences” in the source language. As for the target language, its “words” are actually phonemes and its “sentences” are actually sequences of phonemes.

3.3.2.2 Phones list

The total number of phones in the Albanian language is 37 : 7 vowels and 30 consonants. This list of phones was generated automatically with the help of a Linux script. In Table 3-2 is the list of all

the phones used in our Automatic Speech Recognition System, together with word samples with both they written and phonetic form.

Type	Phoneme		Words Samples	
	IPA	Used	Written form	Phonetic form
vowels	i	i	ali	ali
	ɛ	e	atyre	at y re
	a	a	artistik	art ist i k
	ə	ë	bëjmë	bel j mel
	ɔ	o	ciko	ci ko
	y	y	bymehet	by me het
	u	u	buxheti	bu x he ti
	p	p	publike	pu bli ke
	b	b	problem	pro ble m
	t	t	pritjen	pr it jen
	d	d	presidenti	pre si den ti
	c	q	paqena	pa qe na
	ɟ	gj	energji	en er gj i
	k	k	hipokrizi	hi po kri zi
	g	g	gyl	gy l
consonants	ts	c	cope	co pe
	dz	x	xhirua	xhi ru a
	tʃ	ç	siç	si çl
	dʒ	xh	xhirua	xhi ru a
	θ	th	rrethanave	rr e th a na ve
	ð	dh	radhe	ra dhe
	f	f	njoftoi	njo ft oi
	v	v	investime	in ve sti me
	s	s	fiskal	fi sk al
	ʃ	sh	ashtu	a sh tu
	z	z	muzika	mu zi ka
	ʒ	zh	zhvillim	zh vi ll i m
	h	h	ish	i sh
	m	m	fillimi	fi ll i mi
	n	n	barnat	ba rn at
	ɲ	nj	njohjes	njo h je s
	ŋ	ng	ngjallur	ng ja ll ur
	j	j	arsyeja	ar sye ja
	l	l	alarmit	al ar mi t
	ɫ	ll	abdullah	ab du ll ah
	r	rr	merrej	me rre j
	ʀ	r	adresuar	ad re su ar

Table 3-2 Albanian Phoneme set

3.3.3 Building the acoustic model

The acoustic model is the representation of the grammar or syntax of the task. As it was concluded in Section 2.5, the state-of-the-art large vocabulary speech recognition systems use Bakis-type Hidden Markov Models (HMMs) with Gaussian Mixture Models (GMMs) as output distributions to model sub-words speech units such as context-dependent phones (tri-phones) or senones.[1] The HMMs model these speech units using acoustic feature vectors (MFCC coefficients) extracted out of the original time-domain speech signal.

3.3.3.1 Speech unit selection

The selection of the speech units was the first issue to be approached. As discussed in Section 2.5.3, words cannot be used as basic speech units when designing a large-vocabulary continuous speech recognition system. They are neither trainable, meaning there are not enough occurrences for every word to robustly train a model, nor generalizable, for every new ASR task, with a new vocabulary, a new set of models needs to be constructed. Thus, sub-words speech units such as context-independent phones (simply called phones), context-dependent phones (called tri-phones) or syllables were taken into consideration. The trainable attribute narrowed our possibilities to phones and tri-phones, since we do not have a large amount of training data to train syllable models.

3.4 Data preparation for training and testing an ASR

3.4.1 Training

CMU Sphinx project offers the possibility to create acoustic models for a new language. The trainer learns the parameters of the models of the sound units using a set of sample speech signals. This is called a training database.

The trainer needs to be told which sound units he has to learn the parameters of, and at least the sequence in which they occur in every speech signal in your training database. This information is provided to the trainer through a file called the *transcript file*, in which the sequence of words and non-speech sounds are written exactly as they occurred in a speech signal, followed by a tag which can be used to associate this sequence with the corresponding speech signal.

The trainer then looks into a *dictionary* which maps every word to a sequence of sound units, to derive the sequence of sound units associated with each signal. There are two dictionaries, one in which legitimate words in the language are mapped sequences of sound units (or sub-word units), and another in which non-speech sounds are mapped to corresponding non speech or speech-like sound units. They are referred as the *language dictionary* and the latter as the *filler dictionary*. [21]

The file structure for the database is:

- ❖ etc
 - database.dic - *Phonetic dictionary*
 - database.phone - *Phoneset file*
 - database.lm.DMP - *Language model*
 - database.filler - *List of fillers*
 - database.fileids - *List of files for training*
 - database.transcription - *Transcription for training*

❖ wav

- fileID.wav – *Recording of speech utterances*

Let's go through the files and describe their format and the way to prepare them :

Fileids (*database.fileids*) file is a text file listing the names of the recordings (utterance ids), one by line, in the format speaker_1/file_1.

Fileids file contains the path in a filesystem relative to wav directory. Note that fileids file should have no extensions for audio files, just the names.

Transcription file (*database.transcription*) is a text file listing the transcription for each audio file.

It is important that each line starts with <s> and ends with </s> followed by id in parentheses. The parenthesis must contain only the fileid. It is critical to have the fileids and the transcription file perfectly synchronised. The number of line in both should be identical and the last part of the fileids file (speaker1/file_1) and the utterance id file_1 must be the same on each line.

Speech recordings (wav files) - Recording files must be in MS WAV format with specific sample rate - 16 kHz, 16 bit, mono for desktop application. Audio files should not be very long and should not be very short. Optimal length is between 5 seconds and 30 seconds. Amount of silence in the beginning of the utterance and in the end of the utterance should not exceed 0.2 seconds Audio format mismatch is the most common training problem.

Phonetic Dictionary (*database.dic*) , as discussed in Section 2.4, should have one line per word with word following the phonetic transcription.

It is important not to use case-sensitive variants like “a” and “A”. Also, Sphinxtrain does not support some special characters like ‘*’ or ‘/’ and support most of others like ‘+’ or ‘.’. To avoid potential errors, it is better to use alphanumeric-only phone-set.

Phoneset file (*database.phones*) should have one phone per line. The number of phones should match the phones used in the dictionary plus the special SIL phone for silence:

Language model file (*database.lm.DMP*) should be in CMU binary format, that is DMP format.

Filler dictionary (*database.filler*) contains filler phones (not-covered by language model non-linguistic sounds like breath, hmm or laugh).

The next step in the training process is to edit the configuration file, sphinx_train.cfg, found in the etc subfolder. Here must be specified the paths to the above mentioned files and also the model parameters, the number of senones and the number of Gaussian densities. The more senones model has, the more precisely it discriminates the sounds. But on the other hand if there are too many senones, model will not be generic enough to recognize unseen speech. That means that the WER will be higher on unseen data. The optimal numbers of these parameters depend on the database. To train the model properly, we tried different values and selected the ones which give the best WER for a development set.

After this, the training process is done in two steps : a) the feature vectors (MFCCs) are extracted from the wav files and b) the acoustic model is trained.

```
Baum welch starting for 2 Gaussian(s), iteration: 3 (1 of 1)
0% 10% 20% 30% 40% 50% 60% 70% 80% 90% 100%
Normalization for iteration: 3
Current Overall Likelihood Per Frame = 30.6558644286942
Convergence Ratio = 0.633864444461992
Baum welch starting for 2 Gaussian(s), iteration: 4 (1 of 1)
```

Figure 3.4 Example of output during decoding

Figure 3.4 represents the typical output during decoding .
The logdir sub-folder will have the following structure :

000.comp_feat
05.vector_quantize
20.ci_hmm
30.cd_hmm_untied
40.buildtrees
45.prunetree
50.cd_hmm_tied

Table 3-3Structure of decoding process

On the stage 000.comp_feat the feature files are extracted. The system does not directly work with the speech signals. The signals are first transformed into a sequence of feature vectors, which are used further.

The script slave_feat.pl will compute, for each training utterance, a sequence of 13-dimensional vectors (feature vectors) consisting of the Mel-frequency cepstral coefficients (MFCCs). The MFCCs will be placed automatically in a directory called 'feat'.

Once the jobs launched from 20.ci_hmm have run to completion, the Context-Independent (CI) models for the sub-words units in the dictionary have been trained.

When the jobs launched from the 30.cd_hmm_untied directory run to completion, the models for Context-Dependent sub-word units (triphones) with united states have been trained. These are called CD-untied models and are necessary for building decision trees in order to tie states.

The jobs in 40.buildtrees will build decision trees for each state of each sub-word unit.

The jobs in 45.prunetree will prune the decision trees and tie the states.

Following this, the jobs in 50.cd-hmm_tied will train the final model for the triphones in the training corpus. These are called CD-tied models and are trained in many begins. It begins with 1 Gaussian per state HMMs, followed by 2 Gaussian per state HMMs and so on till the desired number of Gaussians per State have been trained. The jobs in 50.cd-hmm_tied automatically train all the intermediate CD-tied models.

After the training is finished, a new sub-folder will appear, called model_parameters. This sub-folder contains :

- **mixture weights** : the weights given to every Gaussian in the Gaussian mixture corresponding to a state
- **transition matrices** : the matrix of state transition probabilities
- **means** : means of all Gaussians
- **variances** : variances of all Gaussians
- **noisedict** : contains SIL
- **mdef** : model definition file for context independent phones CI. The function of a model definition file is to define or provide a unique numerical identity to every state of every HMM that you are going to train, and to provide an order which will be followed in writing out the model parameters in the model parameter files. During the training, the states are referenced only by these numbers.

3.4.1.1 The hierarchical training strategy

The core acoustic models of a modern speech recogniser typically consist of a set of tied three-state HMMs with Gaussian output distributions. This core is commonly built in the following steps :

1. HMM system design. This step consists in choosing the speech units to be modeled, the voice features to be used as modeling parameters, the HMM topology and the number of Gaussian mixtures per state. The output of this stage is a set of „prototypes” – all the models are given default values.
2. System initialization. The set of „prototypes” resulted from the previous stage is initialized using the isolated speech unit training technique. The small isolated phones database (PHONES) is utilized for this purpose. The alternative to this type of system initialization would be „flat starting”, which involves computing the voice features for each frame of each speech utterance, and using the statistical results as initial model parameters. The output of this stage is a set of roughly initialized HMMs.
3. Embedded phone HMM training. Now, the initialized models are trained using the isolated words database (WORDS). Whereas isolated unit training is sufficient for building whole word models, the main HMM training procedures for building sub-word systems revolve around the concept of *embedded training*. This method is employed because the WORDS database provides only information on the order of the phones and not on their temporal borders. In the end, the output of this stage is a robustly trained HMM set. When performing embedded training, it is good practice to monitor the performance of the models on unseen test data and stop training when no further improvement is obtained.
4. Embedded triphone HMM training. This stage consists in two steps : the first one is a design adjustment that aims to create triphone models $x - q + y$ and the second one is another training session. Given the best set of phone HMMs trained at the previous step, we can build triphone HMMs by cloning the phone models (a triphone HMM is created by cloning the phone HMM for the central state). The newly created triphone HMM set is retrained (through embedded-training) by using the WORDS database. After this training session the performance of the system is reevaluated. ($x - q + y$ denotes the triphone corresponding to phone q spoken in the context of a preceding phone x and a following phone y)
5. Embedded triphone HMMs training. This step uses models obtained at step four and retrains them using the WORDS database. Finally, the continuous speech ASR system performance is evaluated.

3.4.2 Testing

It is critical to test the quality of the trained database in order to select best parameters, understand how application performs and optimize parameters. To do that, a test decoding step is needed. The decoding is now a last stage of the training process.

After testing the models built on the small database, the next step was to try to extend them, in order to obtain an accurate speech recognition system.

3.5 Final Results for Albanian ASR

Table 3-3 presents the speech databases that we built during our research period. The total amount of training speech data summed up to about 8 hours of speech from 23 different speakers and 6 hours of audio clips extracted from the websites' databases. SD2 contains audio clips which are not of god quality, being filtered at 5.5 kHz. SD4 contains audio clips extracted from www.topchannel2.tv in which several speakers are present. Even though we have the transcriptions for all the spoken text, the clips are too long. Consequently, Sphinx toolkit fails to perfectly align the audio to the transcription.

ID	database	Duration	filtered low-pass [kHz]	type	comments
SD1	MediaEval 2013	2	8	recordings	native speakers
SD2	Chunk1-6	3:20	5.5	broadcastnews	expert
SD3	Chunk 7-10	2	8	broadcastnews	expert
SD4	topchannel2	5:40	8	broadcastnews	loose

Table 3-4 Albanian Speech database

Table 3-4 summarizes the data regarding the text corpora that were collected and further used in the experiments. The numbers are computed on the clean corpora (after the processing operations described in the previous section).

ID	database	#total words	#unigrams	#phrases
TD1	MediaEval2013	14063	1165	969
TD2	topchannel+vizionPlus+balkanweb	48560551	377170	4841049
TD3	MediaEval2013+Chunk 1-10	74063	11342	3340
TD4	Chunk 7-10	22003	11342	845
TD5	topchannel2	57129	12310	187

Table 3-5 Albanian Text copora

The acoustic models were created using the CMU Sphinx toolkit and the default training strategy. We employed (–states) HMMs to model context-dependent phones (triphones) using Mel-Frequency Cepstral Coefficients (MFCCs). The total number of HMM states (called senones) was limited to 1000. Every senone was modelated with a Gaussian Mixture Model (GMM) with 8 Gaussian components.

ID	trained on
AM01	SD1
AM02	SD1+SD2
AM03	SD1+SD2+SD4
AM04	SD3
AM05	SD3+SD4

Table 3-6 Albanian Acoustic Models

Finally, the large-vocabulary attribute of our continuous speech recognition system was achieved by creating a general language model for Albanian using all the text corpora available, that we had processed from the website news databases. This language model is denoted LM02.

ID	trained on
LM01	TD2
LM02	TD3
LM03	TD4
LM04	TD1+TD5(90%)+TD2(10%)
LM05	TD4+TD5

Table 3-7 Albanian Language Models

Table 3-8 presents the various experiments made during a period of one year, consequently revealing the evolution of our ASR system.

Exp	ASR System	Used Models	Evaluation set	Accuracy [%]	Error rate [%]
1	ASRS-1	AM05+LM05	SD1	14.46	85.54
2	ASRS-2	AM02+LM02	SD1+SD2	45.63	54.37
3	ASRS-3	AM04+LM01	SD3	38.88	61.62
4	ASRS-4	AM02+LM01	SD3	25.21	74.79
5	ASRS-5	AM04+LM03	SD3	72.39	27.61
6	ASRS-6	AM02+LM03	SD3	66.14	33.86
7	ASRS-7	AM04+LM04	SD3	40.87	59.13

Table 3-8 Albanian experimental results

Several conclusions can be drawn from Table 3-7. First, ASRS-1 had a low accuracy percentage. From these results, we drawn the conclusion that topchannel2 was not a trust worthy speech database. The audio clips were too long, and the Sphinx toolkit failed to align the whole clip to the audio. Moreover, since the clips were raw news broadcasts, the environment in which they were recorded presented different disadvantages. These wavs contained high additive noise, multiple acoustic sources, like music or other people talking in the back, or reverberant environments.

Another issue worth mentioning is the difference between ASRS-4 and ASRS-6. Even though they are tested on the same database, that is SD3, and use the same acoustic model, that is AM02, their performance is very different. Obviously, the difference lies in the language model. ASRS-6 has a greater accuracy percentage because the language model is trained on the same database on which it is tested. Even though ASRS-6 would seem a good ASR at a first glance, these are called artificially improved results.

In the end, the best configuration seems to be ASRS-5. But this configuration uses models that are trained and tested on the same database, that is SD3. In order to properly evaluate an ASR, it must be tested on unseen data. Consequently, we trained an interpolated language model LM04 (MediaEval2013 + topchannel2 (90%) + all news corpora (10%)) in order to obtain some real results. ASRS-7 reflects the real performances of our continuous recognition system.

3.6 Automatic Speech Recognition System for English

Our next targeted language was English. Here, our job was simplified because we had access to a very large speech database. **TIMIT** is a corpus of phonemically and lexically transcribed speech of American English speakers of different sexes and dialects and is designed for the development

and evaluation of automatic speech recognition systems. TIMIT contains broadband recordings of 630 speakers of eight major dialects of American English, each reading ten phonetically rich sentences. The TIMIT corpus includes time-aligned orthographic, phonetic and word transcriptions as well as a 16-bit, 16kHz speech waveform file for each utterance. Our task consisted in creating the models necessary for speech recognition.

3.6.1 Building the language model

The main goal of our work regarding English language was to develop a language model suitable for English continuous speech. The n-gram paradigm and the theoretical details about the construction of n-gram language models were presented in Section 2.2.1. As already mentioned, a large amount of textual data is required to create a general language model, one that is suitable in various domains. We used all the textual transcriptions of the audio files in the TIMIT speech database to create a tri-gram language model. The total number of phrases was 6300. Table 4-1 summarizes the available text corpora for training a language model.

database	#total words	#unigrams	#phrases	Duration [h]
TIMIT	54375	6103	6300	5

Table 3-9 TIMIT Text corpora

Before using it for continuous speech recognition, we wanted to evaluate the language model in terms of perplexity (PPL) and out-of-vocabulary (OOV) rate. For this, we made a new language model with 90% of the text corpora and evaluated it on the remaining 10%. The concepts of perplexity and out-of-vocabulary rate were discussed in Section 2.2.3.

Exp	Language model built with corpus	Evaluation test set	OOV	PPL
1	90% of Timit database	10% of Timit database	1680	12.93

Table 3-10 Language model evaluation for TIMIT database

Perplexity is a measurement of how well a probability distribution predicts a sample. In other words, it determines how well can the model predict the next word in a sentence. It is the probability of the test set, normalized by the number of words. The out-of-vocabulary words are words that cannot be predicted by the language model, since they were not found in the training set.

3.6.2 Building the phonetic model

As it was discussed in Section 2.4, a phonetic model is used to link the acoustic model to the language model.

3.6.2.1 Phones list

The list of phones (24 consonants and 20 vowels) in English language is presented below, in Table 3-11

IPA	Phone	Example	Phonetic
ɑ:	<i>aa</i>	f A ther	f aa1 dh axr
æ	<i>ae</i>	f A t	fat f ae1 t
ʌ	<i>ah</i>	b U t	b ah1 t
ɔ:	<i>ao</i>	d O Or,	d ao1 r
aʊ	<i>aw</i>	h O W	hh aw1
ə	<i>ax</i>	A bout	ax b aw1 t
aɪ	<i>ay</i>	h I de	hh ay1 d
e	<i>eh</i>	g E t	g eh1 t
ɪ	<i>el</i>	tab L E	t ey1 b el
m	<i>em</i>	syst E M	s ih1 s t ax m
n	<i>en</i>	tak E N	t ey1 k ix n
ɜ:	<i>er</i>	s E ARch	s er1 ch
eɪ	<i>ey</i>	g A te	g ey1 t
ɪ	<i>ih</i>	b I t	b ih1 t
i	<i>iy</i>	happ Y	hh ae1 p iy
i:	<i>iy</i>	b E At	b iy1 t
əʊ	<i>ow</i>	n O se	n ow1 z
ɔɪ	<i>oy</i>	t O Y	t oy1
ʊə	<i>uh</i>	f U ll	f uh1 l
u	<i>uw</i>	f O od	f uw1 d
b	<i>b</i>	B ook	b uh1 k
tʃ	<i>ch</i>	CH art	ch aa1 r t
d	<i>d</i>	ba D	b ae1 d
ð	<i>dh</i>	fa TH er	f aa1 dh axr
f	<i>f</i>	lau GH	l ae1 f
g	<i>g</i>	G ood	g uh1 d
h	<i>hh</i>	H ello	hh eh2 l ow1
dʒ	<i>jh</i>	J acket	jh ae1 k ix t
k	<i>k</i>	K ill	k ih1 l
l	<i>l</i>	L ate	l ey1 t
m	<i>m</i>	ga M e	g ey1 m
n	<i>n</i>	ma N	m ae1 n
ŋ	<i>ng</i>	sitti NG	s ih1 t ix ng
p	<i>p</i>	P ath	p ae1 th
r	<i>r</i>	R eason	r iy1 z en
s	<i>s</i>	ma SS	m ae1 s
ʃ	<i>sh</i>	SH ip	sh ih1 p
t	<i>t</i>	ba T	b ae1 t
θ	<i>th</i>	TH eatre	th iy1 t axr
v	<i>v</i>	V arious	v ae1 r iy ax s
w	<i>w</i>	W ater	w ao1 t axr
j	<i>y</i>	Y ellow	y eh1 l ow2
z	<i>z</i>	boy S	b oy1 z
ʒ	<i>zh</i>	vi S ion	v ih1 zh ix n
	<i>pau</i>	S hort	

Table 3-11 Phoneme set in English

3.6.3 Building the acoustic model

In order to properly evaluate the ASR, the models need to be tested on unseen data. For that, we used 80% of the database for training the acoustic model, and the rest of 20% for testing the models, in order to be able to draw some conclusions.

After many experimental setups, we decided to set the number of senones to 1000 and the final number of Gaussian densities at 32.

Exp	#no of senones	Accuracy [%]	Error [%]
1	1000	39.22%	60.78%
2	4000	29.71%	70.29%

Table 3-12 Results for TIMIT database

From table 4-3 can be drawn the conclusion that the number 4000 set for senones resulted in a model not generic enough, that translated into a higher WER on unseen data. The difference between the experiment 2 with 4000 senones and experiment 1 with 1000 senones is reflected in an average of 9.51% performance drop (expressed as the error increases).

3.7 Automatic Speech Recognition System for Romanian

The last targeted language was romanian. As in the case for english, we had access to a very large speech database. Speed Research Laboratory staff provided us 1.5G of speech resources, which we further used in the Romanian ASR implementation.

For database4, several speakers denoted 01,02,03.. 20 recorded several sets of audio clips, as follows :

ID	Type of speech	Domain
00 - 10000	isolated words	one word per file
01 - 1019	continuous read speech	newspaper articles
02 - 244	continuous (dialogue) read speech	library related
03 - 150	spontaneous read speech	tourism related - media corpus
04 - 150	spontaneous read speech	tourism related - media corpus

Table 3-13 Romanian database

Gathering all these audio clips, we created a speech database of 65 hours, from which we used 97% for training and 3% for testing.

3.7.1 Building the language model

As can be seen in Table 4-6, we had access to a very large database. Due to this amount of data, we had the possibility to design a large-vocabulary continuous speech recognition system for Romanian.

database	#total no. of words	#unique words	Duration [h]
database4	254 347	10038	35

Table 3-14 number of words for Romanian database

3.7.2 Building the phonetic model

As it was discussed in Section 3.2.2., a phonetic model is used to link the acoustic model to the language model. Speed provided us with the automatic phonetisation, so we successfully built the phonetic dictionary.

3.7.2.1 Phones list

In our studies we have employed the set of 34 phones. The table lists the standard IPA symbols along with the used symbols and also gives some words examples.

Type	Phoneme		Words	
	IPA	Used	Written	Phonetic
vowels	a	a	sAt	s a t
	ə	a1	gurĂ	g u r a l
	e	e	marE	m a r e
	i	i	lIft	l i f t
	j	i1	tarI	t a r i l
	ı	i2	Între	i2 n t r e
	o	o	lOc	l o c
	u	u	şUt	s l u t
	y	y	ecrU	e c r y
	ø	o2	blEU	b l o2
semi-vowels	ɛ	e1	dEal	d e l a l
	j	i3	fIară	f i3 a r a l
	ɔ	o1	Oase	o l a s e
	w	w	saU	s a w
	c	k2	CHem	k2 e m
	b	b	Bar	b a r
	p	p	Par	p a r
	k	k	aCum	a k u m
	(tʃ)	k1	Cenuşă	k l e n u s l
	g	g	Galben	g a l b e n
consonants	dʒ	g1	Girafă	g l i r a f a l
	ʒ	g2	unGHI	u n g2
	d	d	Dar	d a r
	t	t	ToT	t o t
	f	f	Faţă	f a t l a
	v	v	Vapor	v a p o r
	h	h	Harta	h a r t a
	ʒ	j	aJutor	a j u t o r
	ʃ	s1	coŞ	k o s l
	l	l	Lac	l a c
	m	m	Măr	m a l r

n	n	Nas	n a s
s	s	Sare	s a r e
z	z	Zar	z a r
r	r	Risc	r i s k
ts	t1	Țăran	t1 a1 r a n

Table 3-15List of phones in Romanian

3.7.3 Building the acoustic model

As in the previous cases, in our attempt to create a continuous speech recognition system for Romanian we have decided to use the state-of-the-art mathematical tool : HMMs with GMMs. After several experimental setups, we decided to set the number of senones to 1000 and the final number of Gaussian densities at 16.

The selection of the speech units was the first issue that mst be approached. Obviously, for a large-vocabulary continuous speech recognition system we cannot use words as basic units. They are neither trainable, because there are not enough occurences for every word to robustly train a model, nor generalizable, because for every new ASR task, with a new vocabulary, a new set of models needs to be constructed. The trainable attribute (there should be enough data to estimate the parameters of the unit) of a properly chosen speech unit limited our possibilities to phones and triphones. The reason is that we do not have enough occurences to train syllable models in the our database.

After training the acoustic model, we tested the obtained models, to be able to evaluate them.

Problems encountered at this step :

- The ASR returned 2-3 words per decoded audio clip, even if the decoded audio clip contained a phrase (more than 3 words). Our first approach to this problem was to decode only wavs that contained phrases, thinking that the decoding of isolated words caused the bad recognition. After we eliminated them from fileids list, the fileids corresponding to wavs with isolated words, the recognition was still very bad. After several experimental setups, we saw that some wavs had 128 kbit/s ratebit, and they were the main cause for the bad recognition results.

Exp	Training set	Evaluation set	Accuracy [%]	Error [%]
1	97% from database4	3% from database4	36.32	63.68

Table 3-16 Evaluation for Romanian database

Table 3-16 summarizes the performance of our continuous speech recognition system for Romanian. Database4 consisted in 107,751 audio files (almost 6700 per speaker) which were split into a training part comprising 104868 (almsot 6500 per speaker) and an evaluation part comprising 2883 files (260 per speaker). The training set contained both isolated words and phrases, while the testing set contained only phrases.

3.8 Demo Application

All our experiments presented so far in this thesis were done in an offline, experimental setup. For demonstration purposes a demo application was created. This application was provided by Sphinx and is developed in the Java programming language. It allows the user to load specific acoustic models and specific language models or grammar at his own choice. Having as basis this Java application, I implemented a graphical user interface (GUI), in order to be more user-friendly. The demo application is very easy to use and is a convenient way to evaluate the recognition system. First, the user must specify the desired language. A screenshot illustrating the first frame of the application is presented in Figure 3.5 .



Figure 3.5 GUI for Demo application

The language is selected by direct speaking, after the *Record Speech* button is pressed. Obviously, a microphone has to be connected to the computer so the application can record the spoken word. I implemented this task using a JSGF grammar.

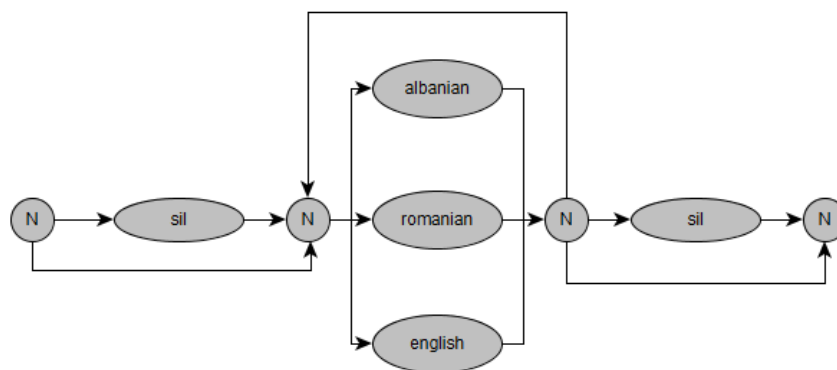
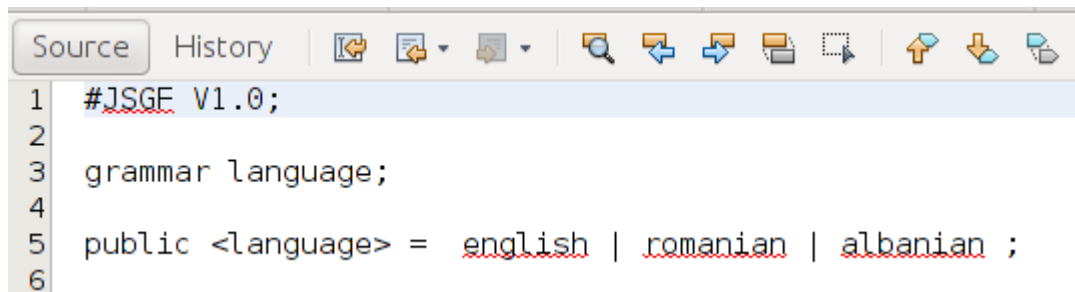


Figure 3.6 The language word graph

Figure 3.6 illustrates the word graph with all the words and the allowable transitions. The nodes named *N* are null nodes. A transition through one of these nodes does not output any word. These nodes serve as start and end points in the word graph. All three words have the same equally probability. The nodes marked as *sil* stand for silence zones in the audio data.



```

Source History
1 #JSFG V1.0;
2
3 grammar language;
4
5 public <language> = english | romanian | albanian ;
6

```

Figure 3.7 FSG Grammar for Demo application

After a language was selected, the frame specific to the language pops-up. From the Combo Box the user can select an ID, corresponding to a wav file and a transcription file specific to the chosen language. I have selected more ways for each of the targeted language, such that the user to be able to perform several tests, in order to evaluate the performance of the automatic recognition systems. Once the user presses the *Start* button, in the second TextArea appears the transcription file and in the first TextArea what our system recognized. The JFrame is illustrated in Figure 3.8.

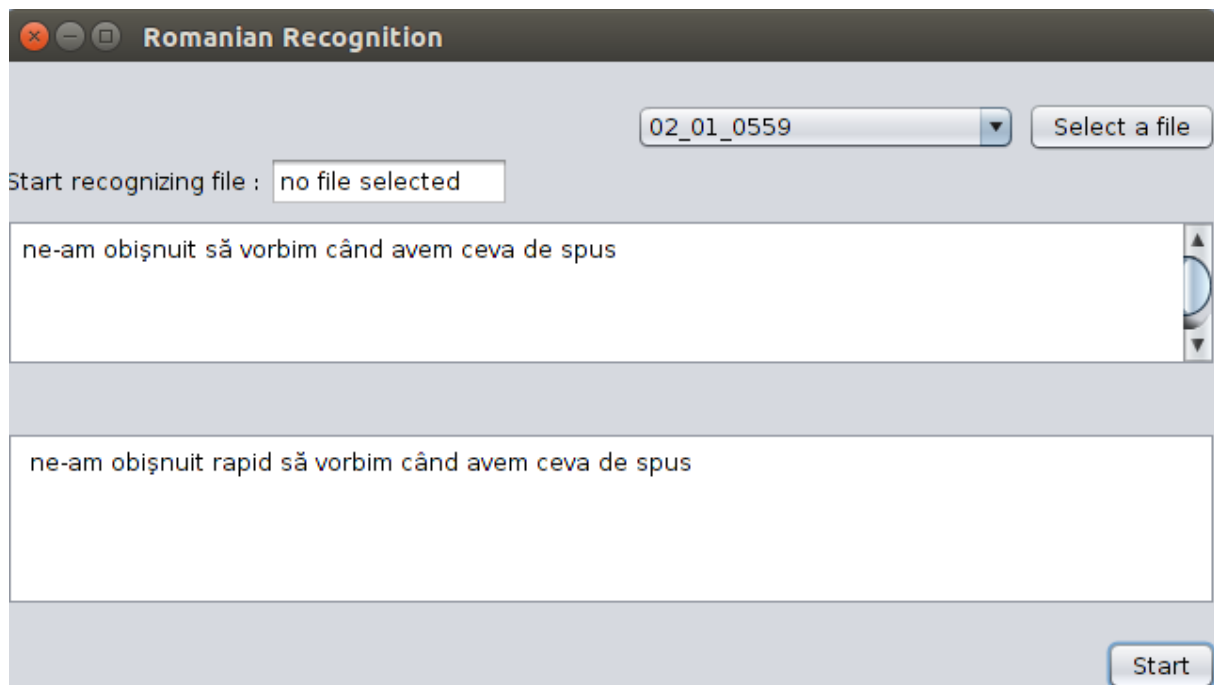


Figure 3.8 GUI for Romanian Recognition

Similar JFrames are made for Albanian and English recognition. The user can perform several tests and draw some conclusions regarding the performance of our Multilingual Automatic Speech Recognition System.

4 Conclusions

The main objective of this thesis was to create an automatic speech recognition system for three languages : Albanian, English and Romanian. Albanian is a so called low-resourced language, a language that is spoken by a large number of people, but no prior work of collecting and organizing speech and/or text resources has been done. At this part, our contribution was to acquire a speech database in order to train and evaluate the ASR. In the second part, we designed ASRs for English and Romanian, two languages with available resources.

This thesis presents the successive steps which were employed in order to create a multilingual speech recognition system. Chapter 1 and 2 describes the theoretical aspects regarding speech recognition. Chapter 3 presents the processing tools required to create an ASR and the stages of training the necessary models. Phonemes were chosen as basic speech units for the HMM-based recognition system, therefore a phonetic dictionary that maps words to their phonetic form is mandatory.

The first sections of Chapter 3 describe explicitly the steps in designing an ASR from zero. It presents the problems encountered when gathering resources for building a speech database. After the required resources are described, together with the cleaning tools for the text corpora, several experiments are presented. As a conclusion, we have managed to build a 14 hours speech database which can be further used to design a large vocabulary-speech recognition system.

Sections 3.6 and 3.7 present the available databases for Romanian and English. Each text corpora is described in terms of total number of words, in number of unique words and in number of phrases. For every speech database we have selected 90% for training purpose and 10% for testing one. The results presented in this thesis approach the reality, since all the tests are done on unseen data.

The last section in Chapter 3 presents a GUI demo application. It comes by default with the CMU Sphinx toolkit, my contribution being the GUI interface. It is a user-friendly application through which one can test our automatic speech recognition systems.

For all the three targeted languages, the inter-speaker variability could be approached by creating even more general acoustic models. The only way to accomplish this is by collecting a larger continuous speech database. The attribute “larger” refers to a more representative database, with at least 150 speakers of different ages, different social environments, different accents etc. This means a more difficult task than the acquisition of the Albanian speech database.

A second perspective consists in adding also number recognition to our ASRs. This is not a trivial task. Take, for example, the spoken representation of the number *1991*. The speech signal has nothing in common with the actual digits 1, 9, 9 and , but is, in fact, the spoken representation of the word sequence *one thousand nine hundred ninety one*. The ASR language model must predict the different words which compose this number and not the digits-written form. Additionally, the range of numbers is potentially infinite, while the range of words used to compose the numbers is limited. Another issue, similar with that of the numbers’ recognition, regards abbreviations. The fact that the speaker utters the *unabbreviated* form makes this task very challenging. Consequently, a new cleaning operation should be added, one that would replace abbreviated word forms with full word forms based on a list of abbreviations.

Bibliography

- [1] Cucu, H., „Towards a speaker-independent, large-vocabulary continuous speech recognition system for Romanian”
- [2] Huang, X., Acero, A., Wuen-Hon., Chapter 11 „Language Modeling” in *Spoken Language Processing- A Guide to Theory, Algorithm, and System Development*, Prentice Hall, 2001
- [3] [Juang], Juang, B.H., Rabiner, Lawrence R., „Automatic Speech Recognition – A brief History of the Technology Development”, Georgia Institute of Technology, Atlanta
- [4] Professor Dan Jurafsky, Lecture 4.6 Interpolation, Stanford NLP, <https://www.youtube.com/watch?v=-aMYz1tMfPg>, accessed at 01/06/2014
- [5] Professor Dan Jurafsky, Lecture 4.3 Evaluation and Perplexity, Stanford NLP, <https://class.coursera.org/nlp/lecture/129>, accessed at 17/06/2014
- [6] Wikipedia, Perplexity, <http://en.wikipedia.org/wiki/Perplexity>, accessed at 29/05/2014
- [7] Huang, X., Acero, A., Wuen-Hon., Chapter 9 „Acoustic Modeling” in *Spoken Language Processing- A Guide to Theory, Algorithm, and System Development*, Prentice Hall, 2001
- [8] Huang, X., Acero, A., Wuen-Hon., Chapter 1 „Introduction” in *Spoken Language Processing- A Guide to Theory, Algorithm, and System Development*, Prentice Hall, 2001
- [9] Wikipedia, Prosody, [http://en.wikipedia.org/wiki/Prosody_\(linguistics\)](http://en.wikipedia.org/wiki/Prosody_(linguistics)), accessed at 30/05/2014
- [10] Jurafsky, D., Martin, J., “Automatic Speech Recognition,” Chapter 7 “Speech Decoding” in *Speech and Language Processing: An introduction to natural language processing, computational linguistics, and speech recognition (2nd Ed.)*, Pearson Education, 2009.
- [11] Wikipedia, Mel scale, http://en.wikipedia.org/wiki/Mel_scale, accessed at 31/05/2014
- [12] Buzo, A., “Automatic Speech Recognition over Mobile Communication Networks,” Teză de doctorat, Universitatea Politehnica din București, România,
- [13] Wikipedia, MFCC, <http://en.wikipedia.org/wiki/MFCC>, accessed at 31/05/2014
- [14] Huang, X., Acero, A., Wuen-Hon., Chapter 8 „Hidden Markov Models” in *Spoken Language Processing- A Guide to Theory, Algorithm, and System Development*, Prentice Hall, 2001
- [15] Gales, M., Young, S., “The Application of Hidden Markov Models in Speech Recognition”
- [16] Wikipedia, WER, http://en.wikipedia.org/wiki/Word_error_rate accessed at 29/05/2014
- [17] <http://www.grymoire.com/unix/sed.html>, accessed at 30/06/2014
- [18] <https://www.ffmpeg.org/>, accessed at 30/06/2014
- [19] Wikipedia, http://en.wikipedia.org/wiki/Speaker_diarisation, accessed at 30/06/2014
- [20] Wikipedia, SMT, http://en.wikipedia.org/wiki/Statistical_machine_translation accessed at 10/06/2014, accessed at 03/07/2014
- [21] CMUSphinx Tutorial, Training Acoustic Model, <http://cmusphinx.sourceforge.net/wiki/tutorialam>, accessed at 10/06/2014
- [22] <https://kb.iu.edu/d/afik>, accessed at 02/06/2014
- [23] <http://www.fileformat.info/tip/linux/iconv.htm>, accessed at 03/06/2014

Annex 1

Source code of the main Java program for the demo application :

```
package edu.cmu.sphinx.demo.transcriber;

import edu.cmu.sphinx.api.Configuration;
import edu.cmu.sphinx.api.LiveSpeechRecognizer;
import edu.cmu.sphinx.api.SpeechResult;
import java.io.IOException;
import javax.sound.sampled.LineUnavailableException;

/**
 *
 * @author ioanacalangiu
 */
public class MainFrame extends javax.swing.JFrame {

    private boolean recEnabled;
    private String afisare;
    private Configuration configuration;
    private Configuration configuration1;
    private SpeechResult result;

    /**
     * Creates new form MainFrame
     */
    public MainFrame() {
        initComponents();
        setLocationRelativeTo(null);

        configuration = new Configuration();

        configuration.setAcousticModelPath("file:///home/ioanacalangiu/Documents/sphinx4-5prealpha-src/sphinx4-5prealpha/models/acoustic/wsjs_8kHz");

        configuration.setDictionaryPath("file:///home/ioanacalangiu/Documents/sphinx4-5prealpha-src/sphinx4-5prealpha/models/acoustic/wsjs_8kHz/dict/cmudict.0.6d");
```

```

configuration.setGrammarPath("file:///home/ioanacalanguiu/Documents/sphinx4-
5prealpha-src/sphinx4-5prealpha/src/apps/edu/cmu/sphinx/demo/dialog/");

    configuration.setGrammarName("language");

    configuration.setUseGrammar(true);

}

/**
 * This method is called from within the constructor to initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is always
 * regenerated by the Form Editor.
 */
@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated Code">
private void initComponents() {

    jButton1 = new javax.swing.JButton();
    jLabel1 = new javax.swing.JLabel();
    jLabel2 = new javax.swing.JLabel();
    jLabel3 = new javax.swing.JLabel();
    jLabel4 = new javax.swing.JLabel();

    setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
    setTitle("Multilingual Speech Recognition");

    jButton1.setText("Record Speech");
    jButton1.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            jButton1ActionPerformed(evt);
        }
    });

    jLabel1.setText("english");

```

```

jLabel2.setText("romanian");

jLabel3.setText("albanian");

jLabel4.setText("Please select your language:");

    javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());

    getContentPane().setLayout(layout);

    layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()

            .addGap(0, 0, Short.MAX_VALUE)

            .addComponent(jButton1))

        .addGroup(layout.createSequentialGroup()

            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

                .addGroup(layout.createSequentialGroup()

                    .addGap(53, 53, 53)

                    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

                        .addComponent(jLabel3)

                        .addComponent(jLabel2)

                        .addComponent(jLabel1)))

                .addGroup(layout.createSequentialGroup()

                    .addContainerGap()

                    .addComponent(jLabel4)))

                .addContainerGap(16, Short.MAX_VALUE))

            )

        );

    layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

        .addGroup(layout.createSequentialGroup()

            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()

```

```

        .addGap(37, 37, 37)
        .addComponent(jLabel4)
        .addGap(18, 18, 18)
        .addComponent(jLabel11)
        .addGap(30, 30, 30)
        .addComponent(jLabel2)
        .addGap(27, 27, 27)
        .addComponent(jLabel3)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 49,
Short.MAX_VALUE)

        .addComponent(jButton1))

    );

    pack();
} // </editor-fold>

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    if (jButton1.getText().equals("Stop recording")) {
        //ResponseText.setText(" ");
        recEnabled = false;
    } else {
        recEnabled = true;
        jButton1.setText("Stop recording");
        System.out.println("start");
        javax.swing.SwingWorker<String, String> _swingWorker;
        _swingWorker = new javax.swing.SwingWorker<String, String>() {
            @Override

            public String doInBackground() throws LineUnavailableException,
IOException, Exception {

                System.out.println("2");

                LiveSpeechRecognizer recognizer = new
LiveSpeechRecognizer(configuration);

```

```

System.out.println("3");
recognizer.startRecognition(true);
System.out.println("4");
while(recEnabled){
    System.out.println("Select your language:");
    String utterance =
recognizer.getResult().getHypothesis();
    System.out.println(utterance);

    if(utterance.equals("english")){
        recognizer.stopRecognition();
        System.out.println("english");
        EnglishFrame_v2 test = new EnglishFrame_v2();
        test.setVisible(true);
        break;
    }
    if(utterance.equals("romanian")){
        recognizer.stopRecognition();
        System.out.println("romanian");
        RomanianFrame test1 = new RomanianFrame();
        test1.setVisible(true);
        break;
    }
    if(utterance.equals("albanian")){
        recognizer.stopRecognition();
        System.out.println("albanian");
        AlbanianFrame test2 = new AlbanianFrame();
        test2.setVisible(true);
        break;
    }
    else {
        System.out.println("Please try again :)");
    }
}

```

```

        }

        System.out.println("stop");

        recognizer.stopRecognition();

        return afisare;
    }

    @Override
    protected void done() {

        jButton1.setText("Record speech_1");

    }

};

 SwingWorker<String, String> _swingWorker = new SwingWorker<String, String>() {

    }

}

/**
 * @param args the command line arguments
 */
public static void main(String args[]) {

    /* Set the Nimbus look and feel */

    //<editor-fold defaultstate="collapsed" desc=" Look and feel setting
code (optional) ">

    /* If Nimbus (introduced in Java SE 6) is not available, stay with the
default look and feel.

    * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
    */

    try {

        for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {

            if ("Nimbus".equals(info.getName())) {

                javax.swing.UIManager.setLookAndFeel(info.getClassName());

                break;

            }

        }

    }

}

```

```

        }

        } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(MainFrame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

        } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(MainFrame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

        } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(MainFrame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

        } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(MainFrame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

        }

//</editor-fold>

/* Create and display the form */
java.awt.EventQueue.invokeLater(new Runnable() {
    public void run() {
        new MainFrame().setVisible(true);
    }
});
}

// Variables declaration - do not modify
private javax.swing.JButton jButton1;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel3;
private javax.swing.JLabel jLabel4;

// End of variables declaration
}

```

Source code of the Romanian GUI : (The one for English and Albanian are exactly the same)

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

package edu.cmu.sphinx.demo.transcriber;

import edu.cmu.sphinx.api.Configuration;
import edu.cmu.sphinx.api.SpeechResult;
import edu.cmu.sphinx.api.StreamSpeechRecognizer;
import java.io.File;
import java.io.IOException;
import java.net.URL;
import javax.sound.sampled.LineUnavailableException;

/**
 *
 * @author ioanacalangiu
 */
public class RomanianFrame extends javax.swing.JFrame {

    private SpeechResult result;
    private boolean recEnabled;
    private String afisare;
    private Configuration configuration;

    /**
     * Creates new form RomanianFrame
     */
}
```

```

    */

    public RomanianFrame() {

        initComponents();

        setLocationRelativeTo(null);

        configuration=new Configuration();

        configuration.setAcousticModelPath("file:///home/ioanacalangu/Documents/sphinx4-5prealpha-src/sphinx4-5prealpha/myProject/models/acoustic/romanian.cd_cont_1000");

        configuration.setDictionaryPath("file:///home/ioanacalangu/Documents/sphinx4-5prealpha-src/sphinx4-5prealpha/myProject/models/phonetic/talkshowPlusTheRest.dic.full");

        configuration.setUseGrammar(false);

        configuration.setLanguageModelPath("file:///home/ioanacalangu/Documents/sphinx4-5prealpha-src/sphinx4-5prealpha/myProject/models/language/europarl9amHotnews.10k.augmented.3GramLM.sorted.dmp");

    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */

    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        jSlider1 = new javax.swing.JSlider();
        jButton1 = new javax.swing.JButton();
        jScrollPane1 = new javax.swing.JScrollPane();
        jTextArea1 = new javax.swing.JTextArea();
        jScrollPane2 = new javax.swing.JScrollPane();
        jTextArea2 = new javax.swing.JTextArea();
        jComboBox1 = new javax.swing.JComboBox();

```

```

jButton2 = new javax.swing.JButton();
jTextField1 = new javax.swing.JTextField();
jLabel1 = new javax.swing.JLabel();

setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
setTitle("Romanian Recognition");

jButton1.setText("Start");
jButton1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton1ActionPerformed(evt);
    }
});

jTextArea1.setColumns(20);
jTextArea1.setRows(5);
jScrollPane1.setViewportView(jTextArea1);

jTextArea2.setColumns(20);
jTextArea2.setRows(5);
jScrollPane2.setViewportView(jTextArea2);

jComboBox1.setModel(new javax.swing.DefaultComboBoxModel(new String[] {
"02_01_0532", "02_01_0549", "02_01_0559", "Item 4" }));

jButton2.setText("Select a file");
jButton2.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton2ActionPerformed(evt);
    }
});

jTextField1.setText("no file selected");

```

```

jLabel1.setText("Start recognizing file :");

javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());

getContentPane().setLayout(layout);

layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

    .addComponent(jScrollPane1)

    .addComponent(jButton1, javax.swing.GroupLayout.Alignment.TRAILING)

    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup())

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

    .addGroup(layout.createSequentialGroup())

        .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)

        .addComponent(jComboBox1,
javax.swing.GroupLayout.PREFERRED_SIZE, 202,
javax.swing.GroupLayout.PREFERRED_SIZE)

        .addGroup(layout.createSequentialGroup())

            .addComponent(jLabel1)

            .addGap(4, 4, 4)

            .addComponent(jTextField1,
javax.swing.GroupLayout.PREFERRED_SIZE, 128,
javax.swing.GroupLayout.PREFERRED_SIZE)

            .addGap(0, 0, Short.MAX_VALUE)))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

    .addComponent(jButton2))

    .addComponent(jScrollPane2,
javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.PREFERRED_SIZE, 644,
javax.swing.GroupLayout.PREFERRED_SIZE)

);

layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

```

```

        .addGroup(layout.createSequentialGroup())

    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addGap(21, 21, 21)

    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
        .addComponent(jComboBox1,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addComponent(jButton2)))
        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
            .addContainerGap(50, Short.MAX_VALUE)

    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
        .addComponent(jTextField1,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addComponent(jLabel1)))

    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(jScrollPane2,
javax.swing.GroupLayout.PREFERRED_SIZE, 79,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addGap(34, 34, 34)
        .addComponent(jScrollPane1,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addGap(18, 18, 18)
        .addComponent(jButton1))
    );

    pack();
} // </editor-fold>

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    if(jButton1.getText().equals("Stop")){

```

```

        recEnabled = false;
    } else {
        recEnabled = true;
        jButton1.setText("Stop");
        jTextArea2.setText(" ");
        System.out.println("1");
        javax.swing.SwingWorker<String, String> _swingWorker;
        _swingWorker = new javax.swing.SwingWorker<String, String>() {
            @Override

            public String doInBackground() throws LineUnavailableException,
IOException, Exception {

                System.out.println("2");

                File file = new File("/home/ioanacalangiu/Documents/sphinx4-
5prealpha-src/sphinx4-5prealpha/myProject/romana/02_01_0532.txt");

                String id=jTextField1.getText();

                File targetInput = new
File("/home/ioanacalangiu/Documents/sphinx4-5prealpha-src/sphinx4-
5prealpha/myProject/romana/"+id+".txt");

                ReadTranscript transcript = new ReadTranscript(targetInput);

                String text = transcript.readFile();

                jTextArea1.setText(text);

                StreamSpeechRecognizer recognizer = new
StreamSpeechRecognizer(configuration);

                System.out.println("3");

                recognizer.startRecognition(new
URL("file:///home/ioanacalangiu/Documents/sphinx4-5prealpha-src/sphinx4-
5prealpha/myProject/romana/"+id+".wav").openStream());

                System.out.println("4");

                while((result = recognizer.getResult()) != null){

                    System.out.println("5");

                    if(recEnabled==true){

                        System.out.println("6");

                        jTextArea2.setText(result.getHypothesis());

                        break;

```

```

        }

    }

    recognizer.stopRecognition();

    return afisare;

}

@Override

protected void done() {

    jButton1.setText("Start");

    jTextField1.setText("no file selected");

}

};

_swingWorker.execute();

}

}

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {

    // TODO add your handling code here:

    jTextField1.setText(jComboBox1.getSelectedItem().toString());

}

/**
 * @param args the command line arguments
 */

public static void main(String args[]) {

    /* Set the Nimbus look and feel */

    //<editor-fold defaultstate="collapsed" desc=" Look and feel setting
code (optional) ">

    /* If Nimbus (introduced in Java SE 6) is not available, stay with the
default look and feel.

    * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html

```

```

        */

        try {

            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {

                if ("Nimbus".equals(info.getName())) {

                    javax.swing.UIManager.setLookAndFeel(info.getClassName());

                    break;

                }

            }

        } catch (ClassNotFoundException ex) {

            java.util.logging.Logger.getLogger(RomanianFrame.class.getName()).log(java.util.
logging.Level.SEVERE, null, ex);

        } catch (InstantiationException ex) {

            java.util.logging.Logger.getLogger(RomanianFrame.class.getName()).log(java.util.
logging.Level.SEVERE, null, ex);

        } catch (IllegalAccessException ex) {

            java.util.logging.Logger.getLogger(RomanianFrame.class.getName()).log(java.util.
logging.Level.SEVERE, null, ex);

        } catch (javax.swing.UnsupportedLookAndFeelException ex) {

            java.util.logging.Logger.getLogger(RomanianFrame.class.getName()).log(java.util.
logging.Level.SEVERE, null, ex);

        }

        //</editor-fold>

        /* Create and display the form */

        java.awt.EventQueue.invokeLater(new Runnable() {

            public void run() {

                new RomanianFrame().setVisible(true);

            }

        });

    }

    // Variables declaration - do not modify

```

```
private javax.swing.JButton jButton1;  
private javax.swing.JButton jButton2;  
private javax.swing.JComboBox jComboBox1;  
private javax.swing.JLabel jLabel1;  
private javax.swing.JScrollPane jScrollPane1;  
private javax.swing.JScrollPane jScrollPane2;  
private javax.swing.JSlider jSlider1;  
private javax.swing.JTextArea jTextArea1;  
private javax.swing.JTextArea jTextArea2;  
private javax.swing.JTextField jTextField1;  
// End of variables declaration  
}
```