FONDUL SOCIAL EUROPEAN
Investeşte în oameni!
Programul Operaţional Sectorial pentru Dezvoltarea Resurselor Umane 2007 – 2013
Proiect POSDRU/6/1.5/S/16 – *Doctoranzi în sprijinul inovării şi competitivităţii*

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

Nr. Decizie Senat 112 din 30.09.2011

TEZĂ DE DOCTORAT

Contribuţii la un sistem de recunoaştere de vorbire continuă, cu vocabular extins, independent de vorbitor pentru limba română

Towards a speaker-independent, large-vocabulary continuous speech recognition system for Romanian

Autor: Ing. Horia CUCU

COMISIA DE DOCTORAT

Preşedinte	Prof.dr.ing. Teodor PETRESCU	de la	Univeristatea Politehnica Bucureşti
Conducător de doctorat	Prof.dr.Ing. Corneliu BURILEANU	de la	Univeristatea Politehnica Bucureşti
Referent	Prof.dr.ing. Laurent BESACIER	de la	Universitatea Joseph Fourier, Grenoble, Franţa
Referent	Prof.dr.ing. Corneliu RUSU	de la	Universitatea Tehnică Cluj-Napoca
Referent	Prof.dr.ing. Cristian NEGRESCU	de la	Univeristatea Politehnica Bucureşti

Bucureşti
2011

ACKNOWLEDGEMENTS

I am deeply grateful to my PhD coordinator, prof.dr.eng. Corneliu Burileanu, for being very persuasive into convincing me to do this PhD and for his continuous guidance and support. I am certain that without his help I would not be presenting today this thesis. I also want to thank him for creating and leading a very strong and large research team, within which I was able to evolve, and for providing us with all the high-tech equipment required and available in any high-performance research center.

I also want to thank prof.dr.eng. Laurent Besacier, my PhD advisor during the training stage in UJF Grenoble, for the very welcoming and fruitful research collaboration. My PhD research was boosted-up by this collaboration and his guidance in speech and language modeling contributed decisively to transforming the initial Romanian CSR system into a LV-CSR system.

I also want to thank the evaluation commission members: prof.dr.eng. Corneliu Rusu, prof.dr.eng. Cristian Negrescu and prof.dr.eng. Teodor Petrescu, for their helpful comments and remarks, which helped me to improve this thesis.

My thoughts and thanks also head towards the large research team in which I had the privilege to work for the past three years. This was in fact a team-effort and many other colleagues helped with their bright ideas and with creating speech databases, phonetic resources, etc. My special thanks go to my closest and most stable colleague, dr.eng. Andi Buzo, with whom I have shared a lot of ideas, discussed and designed many tools and algorithms and finally managed to assemble the LV-CSR system.

Finally, I would like to thank my family and friends who have supported me this whole time. I am deeply grateful to my wife Valentina who helped me and supported me all along. Thank you for being so patient and understanding and for all your help!

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 THESIS MOTIVATION

The world we live in has drastically changed over the past few decades. The development of the first computers and their availability and usage on large-scale has played a significant role in this change. The computational power was initially available only within research centers and army institutes and afterwards within large companies. Lately, the computational power became available to almost anyone in the form of personal computers, smart-phones, PDAs, etc. Nevertheless, the regular consumer is not familiar with the human-machine interfaces available today and the further spreading of high-tech is somehow constrained by this issue. The human being is used to speak, to gesticulate, to think, etc. as opposed to type or use a mouse. This is the main reason for which human-machine interfaces have lately become an important topic for the research community. Spoken dialogue systems are apparently the most natural communication systems between humans and machines because speech is in fact the most natural communication method used by humans to exchange information. The human user is not required to have any special skills to be able to use a spoken dialogue system. On the other hand, the computer requires sophisticated tools to be able to understand what the user is speaking (speech recognition and understanding systems) and to be able to speak (speech synthesis systems).

The field of Automatic Speech Recognition (ASR) has been a hot topic in the international scientific community for over twenty years now. This led to the development of resources and methods and eventually high-performance commercial systems for most of the internationally-spoken languages such as English, French, Mandarin, etc. One important advantage for these languages was the fact that speech resources needed to build robust acoustic models and text resources needed to build general or domain-specific language models were widely available on the Internet or were systematically provided by the evaluation programs organized by DARPA (Defense Advanced Research Projects Agency), NIST (National Institute of Standards and Technologies) and other organizations. On the opposite side, for many other languages the performance of ASR systems is directly dependent on the amount of available resources and is generally passable only in very particular cases: small vocabulary, or isolated words recognition or closed task grammar, etc. Moreover, the task of creating new language or acoustic resources for a given spoken language is typically a costly and tedious task. Given these facts, the amount of effort currently being invested into porting and adapting language and acoustic resources and even models from high-resourced languages to low-resourced languages is perfectly motivated.

For Romanian, the target language of this thesis, there are practically very few speech and text resources available. Most of the existing resources have been developed by research groups and are not widely available. Even though several Romanian research groups focus on speech recognition, the lack of resources is probably the main reason for which a large-vocabulary continuous speech recognition (LV-CSR) system for Romanian has not been developed yet.

This thesis is motivated by these facts and was started with the purpose of creating a LV-CSR system for Romanian and setting up a methodology for the development of domain-specific ASR systems for low-resourced languages.

1.2 The Field of Speech Recognition

The automatic speech recognition (ASR) process addresses the problem of mapping an acoustic signal to a sequence of words. Automatic speech understanding (ASU) extends this goal to producing some sort of understanding of the sentence, rather than just the words. When the input acoustic signal contains speech uttered by different speakers, the ASR task can be regarded as a two-step process: speaker diarization (who spoke when?) and speech-to-text transcription (what did he say?).

Automatic speech recognition has a wide range of applicability. The most important domain seems to be that of hands-free and eyes-free interfaces to computers or other devices. There are many applications in which the users need to use their hands and eyes for something else and speech remains their only alternative to being efficient. Moreover, as emphasized in the previous section, speech is the most natural mean of communication for human beings. Other major application areas are spoken dialogue systems for call centers and speech-to-speech translation systems. Speech-to-speech translation is at this moment a very hot topic in many academic and industrial research centers. Finally, ASR is applied to dictation: transcription of an extended monologue by a single specific speaker. Dictation is common in several fields, such as law, where many trials or official meetings need to be transcribed for further reference. Each of these applications is typically more restrictive than the general problem which requires the automatic transcription of naturally spoken continuous speech, by an unknown speaker, in any environment. The various sources of speech variability, which will be discussed further on, make the general task a very challenging one. Nevertheless, in many practical situations, the variability is restricted. For example, there may be a single, known speaker, or the speech to be recognized may be carefully dictated text rather than a spontaneous conversation, or the recording environment may be quiet and non-reverberant. In speech-to-text transcription, a distinction is made between parts addressing acoustic variability (acoustic modeling), and parts addressing linguistic uncertainty (language modeling).

One of the most important factors which influence the difficulty of the speech transcription process is the specific *speech recognition task*. This includes the language, the size of the vocabulary to be recognized and the linguistic uncertainty of the domain. Different spoken languages present different challenges for a speech recognizer. For a large number of languages there are very few speech and text resources available. These so-called *low-resourced languages* are spoken by a large number of people, but no prior work of collecting and organizing speech and/or text resources has been done. Consequently the task of designing an ASR system has to include resource collection also. There are other languages and dialects which are mostly spoken and have practically no written resources for language modeling. In this case the situation is even worse, because there is no way of acquiring the language resources and, in general, the linguistic rules are very loose.

Other languages "suffer" from a complex morphology. For example *rich-morphology languages* such as French and Romanian have larger vocabularies than poor-morphological languages such as English. In Romanian the present tense of the verb *to go* has five morphologically different forms: "merg", "mergi", "merge", "mergem", "mergeți", "merg", while in French it has six: "vais", "vas", "va", "allons", "allez", "vont". In English, the same verb has only two morphologically different forms: "go", "goes". German and Turkish are some of the so-called *agglutinative languages*. In these languages a large number of new words can be formed by concatenation of morphemes. This also leads to larger vocabularies and consequently makes automatic speech recognition a more challenging task.

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

The rough percentage of incorrect words on several standard speech recognition tasks is presented in Table 1.1. The data refers to state-of-the-art ASR systems designed for English [Jurafsky, 2009]. The word error rate (WER) is the standard performance figure used for ASR evaluation (see Section 2.5).

Table 1.1 WER results reported around 2005 for ASR on various tasks [Jurafsky, 2009]

ASR Task	Vocabulary size	WER [%]
TI Digits	11 words (zero-nine, oh)	0.55
Wall Street Journal read speech	5000 words	3.0
Wall Street Journal read speech	20000 words	<6.6
Broadcast News	64000+ words	9.9
Conversational Telephone Speech	64000+ words	20.7

Another important factor which influences the difficulty of the speech process is the speaking style. The speaking style refers to how fluent, natural or conversational the speech is. Obviously, *isolated words speech* recognition, in which each word is surrounded by some sort of pause, is much easier than recognizing *continuous speech* in which words run into each other and have to be segmented. In fact, in the early days of automatic speech recognition, systems solved the problem of where to locate word boundaries by requiring the speaker to leave pauses between words: the pioneering dictation product Dragon Dictate [Baker, 1989] is a good example of a large-vocabulary isolated words recognition system.

Continuous speech tasks themselves vary greatly in difficulty. For example, the task of recognizing *read speech* is much easier than the task of recognizing more natural styles of speech such as *conversational* or *spontaneous speech*. The greater acoustic variability makes the latter task more challenging. This difference in difficulty between continuous speech tasks is reflected in the increased word error rates for spontaneous speech recognition compared with the recognition of read speech (see Table 1.1).

The difficulty and consequently the accuracy of the speech recognition process is also influenced by the *acoustic environment* in which the speech is recorded, along with any *transmission channel*. Outside of quiet offices and laboratories, there are usually multiple acoustic sources including other talkers, environmental noise and electrical or mechanical devices. In many cases, it is a significant problem to separate the different acoustic signals found in an environment. The microphone used for recording also has a significant impact on the speech recognition accuracy. Commercial dictation systems and most of the laboratory research in speech recognition are done with high-quality, head-mounted microphones. Other types of microphones come with different disadvantages which contribute to the quality of the ASR system. Variations in transmission channel occur due to movements of the talker's head relative to the microphone and transmission across a telephone network or the internet. Probably the largest disparity between the accuracy of automatic speech recognition compared with human speech recognition occurs in situations with high *additive noise*, *multiple acoustic sources*, or *reverberant environments*. Noisy speech with a low signal-to-noise ratio can cause the word error rates to go up by 2 to 4 times compared to clean speech.

Finally, the speaker characteristics have also a significant impact on the accuracy of a speech recognizer. The variability in speaker characteristics resides in the speaker accent, the language/dialect he uses, whether he is a native or a non-native speaker, the speech rate, the speaker age and of course the differences in the speech production anatomy and physiology. Moreover, different speakers exhibit different degrees of intrinsic variability based on the emotional state, temporary health problems, etc. The inter-speaker variability can be dealt with by designing *speaker-dependent* ASR systems. The drawback here is that a new acoustic model has to be created for every new speaker. This leads to a more complex system, but also raises several trainability issues (insufficient training data for every speaker and others). On the other hand, *speaker-independent* ASR systems are simpler and more flexible (they can be used to recognize the speech of any speaker). Nevertheless, a speaker-independent system is less accurate for a given speaker when compared to a speaker-dependent system for that particular speaker (if sufficient training data is available for the speaker). Although speaker adaptation algorithms have made great progress over the past 15 years, it is still the case that the adaptability and robustness to different speakers exhibited by automatic speech recognition systems is very limited compared with human performance.

The speaker characteristics variability is evident and very annoying in native versus non-native speech. Although human beings can understand quite well non-native speech, the automatic speech recognition systems exhibit very limited robustness when they are required to recognize this type of speech. Several studies reported huge differences in performance for native versus non-native speech on the same ASR task. For example, the word error rate on Vietnamese-accented French and Chinese-accented French has been reported to be about 5 times higher than for native speakers on the same task [Tan, 2008]. Similarly, the word error rate on Korean-accented English has been reported to be about 9 times higher than for native speakers [Oh, 2007]. Obviously, the differences also depend on the speaking level for the non-natives and on the relationship between the two languages. For example, [Wang, 2003] reports that the word error rate on German-accented English is only 3 times higher than for native speakers on the same task. Nevertheless, non-native speech recognition is still an open issue and a high number of studies (among which we mention [Tan, 2007; Oh, 2007; Tan, 2008; Sam, 2010]) have been published in the past few years on this subject.

Although the field of speech recognition is very broad and different applications and possible solutions can be imagined, our focus in this thesis is on *large-vocabulary continuous speech recognition* (LV-CSR) systems. Large-vocabulary generally means that the systems have a vocabulary of roughly 20k to 60k words. As described above, continuous speech means that the words run into each other naturally. Moreover, our focus is on *speaker-independent* systems (systems that are able to recognize speakers to which the system has never been exposed before).

The state-of-the-art paradigm for large vocabulary continuous speech recognition is the hidden Markov model (HMM). The HMM framework has been introduced as a viable candidate for the acoustic modeling part of speech recognition back in 1975 [Baker, 1975]. For LV-CSR in particular, the HMM-based acoustic model is used in conjunction with an n-gram model which is responsible for the language modeling part. Statistical language models (n-grams) have become the state-of-the-art solution for language modeling since the tremendous expansion of the Internet, which provided enough data to suitably train these systems. Large-vocabulary continuous speech recognition systems and the specific problems they pose are explored in-depth in Chapter 2.

1.3 AUTOMATIC SPEECH RECOGNITION SYSTEMS FOR ROMANIAN

The field of speech recognition for the Romanian language has been approached by several research groups in Romania since the 1980's [Burileanu, 1983; Drăgănescu, 1986]. The first studies focused on simple tasks such as vowels recognition [Grigore, 1998], isolated words recognition [Burileanu, 1998; Valsan, 1998a; Sabac, 1998; Burileanu, 2004] and word spotting algorithms [Valsan, 1998b; Burileanu, 2003].

The main problem which inhibited the development of high-performance continuous speech recognition systems was the absence of standard speech and text resources for Romanian. Specific speech databases have been created over the years by ASR research groups, but these resources have not been standardized and are not publicly available. The authors of [Munteanu, 2006; Dumitru, 2008; Kabir, 2011] explicitly assert that there are no speech resources available for Romanian and that they were required to create speech databases before starting any research in continuous speech recognition. Due to this fact, large-vocabulary continuous speech recognition systems for Romanian are still a future plan. The latest work in speech recognition is still limited to small-vocabulary tasks, basic word-loop grammars or basic n-gram language models and pseudo speaker-independency. For example, in [Oancea, 2004] the authors report small-vocabulary (approximately 3000 words) continuous speech recognition results for a general ASR task modeled with a basic word-loop language model. The number of speakers is limited to 10, so speaker-independency is out of the question. Further development and research on speech recognition algorithms and techniques is reported in [Dumitru, 2008]. This work is still limited to a small-vocabulary task (approximately 4000 words) and presents recognition results for only 11 speakers. The first study which uses more complex language models (bi-gram LMs) for Romanian is [Militaru, 2009]. This work is also closer to speaker-independency, as it uses speech data from 30 speakers. Nevertheless this paper does not approach a general, large-vocabulary task, but a small-vocabulary (approximately 500 words), domain-specific ASR task (broadcasted forecast news).

In order to overcome the small speech database problem, researchers have tried to come up with innovative speech recognition methodologies. Several types of voice features were evaluated in [Gavăt, 2007; Dumitru, 2008] and lots of parameter-tuning experiments were performed in [Munteanu, 2006]. Moreover, several other recognition frameworks (different from the state-of-the-art HMM framework) were tried out. Neural network based approaches are presented in [Dumitru, 2008; Domokoş, 2009], a vector-quantization algorithm is illustrated in [Burileanu, 2004] and more complex, hybrid recognition techniques (fuzzy-HMMs and neural networks plus HMMs) are proposed and evaluated in [Dumitru, 2008].

Some Romanian speech recognition studies discuss a more important issue for the *international* scientific community: speech recognition robustness to various factors. The robustness to noise is dealt with by [Munteanu, 2008]. In this paper a multi-style training methodology is proposed and evaluated on a Romanian speech database. The results show that the proposed methodology is very effective even for signal-to-noise values as low as 10 dB. In [Giurgiu, 2011] the authors propose a methodology (vocal tract normalization) to increase ASR robustness to inter-speaker variations. The methodology is evaluated on clean and noisy speech. The paper concludes that vocal tract normalization is only able to improve the ASR performance on clean speech.

In conclusion, up until now several steps have been made towards developing a large-vocabulary continuous speech recognition system for Romanian, but the final goal has not been reached yet. Moreover, speaker-independency which is directly dependent on the speaker-variability of the training speech database has not been obtained. These two attributes: large-vocabulary and speaker-independent are indispensable for a general-purpose speech recognition system.

1.4 THESIS OBJECTIVES AND OUTLINE

Given the review of the speech recognition systems for Romanian illustrated above, the main objective of this thesis was the development of a speaker-independent large-vocabulary continuous speech recognition system for Romanian. This system should be able to recognize general Romanian continuous speech produced by any speaker with a decent performance. In order to achieve the main goal, several specific objectives were addressed:

a) The acquisition of phonetic, speech and text resources. These resources are all required to create a speech recognition system. A phonetic dictionary is needed to link the words to their phonetic form, a speech database is needed to create and evaluate the acoustic model, while text corpora are required to create general and/or domain-specific language models.

b) The development of specific tools needed to create and process the above mentioned resources and required to create and optimize the acoustic and language models.

c) The design, implementation and evaluation of a Romanian LV-CSR system using state-of-the-art techniques: the HMM framework for acoustic modeling and the n-gram paradigm for language modeling.

d) The design, implementation and evaluation of an ASR domain-adaptation methodology for under-resourced languages.

The thesis is organized in eight chapters, as follows:

Chapter 1 introduces the reader to the field of speech recognition and makes a brief summary of the main issues in this field. The first chapter continues with a review of the various Romanian speech recognition studies performed in the past several years. The solved and unsolved problems are underlined. Based on this review, chapter 1 concludes with the objectives and the outline of this thesis.

Chapter 2 presents the theoretical basis for large-vocabulary speech recognition. The speech recognition formalism is briefly explained and the general architecture of a state-of-the-art speech recognition system is detailed. The core of the second chapter comprises theoretical aspects regarding the development of n-gram language models and the development of HMM acoustic models. The state-of-the-art tools for language modeling and acoustic modeling are also summarized in this second chapter.

In *Chapter 3* we focus on presenting the main theoretical principles of statistical machine translation (SMT). As the thesis will show, SMT can be successfully used to help in the development phase of automatic speech recognition systems. The following chapters use SMT to create phonetic resources and domain-specific language models.

Chapter 4 is the first chapter that illustrates specific contributions of the author of this thesis. The phonetic, speech and text resources required for the development of ASR systems are listed and detailed in this chapter. We review the existing resources and discuss their pluses and minuses in the context of ASR. The core of the chapter comprises the description of the acquisition and processing tasks for these various resources. The analysis of the collected resources is also very important as this leads to various conclusions regarding the Romanian language. This chapter also presents several NLP tools and speech acquisition tools developed by the author of this thesis.

Chapter 5 deals with the development and optimization of the acoustic models. The extensive experiments that were performed in order to find the best setup for the HMM acoustic model are presented in this chapter. Besides these, chapter 5 proposes several methodologies for improving the recognition speed for isolated words. Isolated words recognition and continuous speech recognition results are given in this chapter.

Chapter 6 describes our efforts towards creating a general language model for the Romanian language. We have experimented with two types of language models: finite state grammars and n-gram language models. The second approach was successful and was adopted to develop a large-vocabulary speech recognition system. This chapter presents the first LV-CSR results for Romanian. The second part of chapter 6 proposes an SMT-based domain adaptation methodology for ASR systems (the second main contribution of the author).

Chapter 7 briefly deals with the speaker-independency issue. The best LV-CSR system presented in the previous chapter is evaluated in a more general setup, including speakers which were not part of the training database. The chapter concludes with some remarks regarding the speaker-independency of the LV-CSR system.

Chapter 8 summarizes the main conclusions of this thesis and underlines the author’s contributions. Moreover, this last chapter briefly discusses the envisioned future developments.

CHAPTER 2

LARGE-VOCABULARY CONTINUOUS SPEECH RECOGNITION

2.1 THE AUTOMATIC SPEECH RECOGNITION FORMALISM

The automatic speech recognition (ASR) process addresses the problem of mapping an acoustic signal to a sequence of words. This task is also called speech-to-text transcription. ASR is one of the first fields in which data-driven, machine learning, statistical modeling approach became standard. The basic statistical framework was created and developed during almost two decades by Baker [Baker, 1975], a team at IBM [Jelinek, 1976; Bahl, 1983] and a team at AT&T [Levinson, 1983; Rabiner, 1989]. The speech-to-text task can be formulated in a probabilistic manner as follows:

What is the most likely sequence of words W in the language L, given the speech utterance X?*

The formal representation uses the argmax function, which selects the argument that maximizes the word sequence probability:

$$W^* = \arg\max_{W} p(W \mid X) \tag{2.1}$$

Equation 2.1 specifies the most probable word sequence as the one with the highest posterior probability, given the speech utterance. Bayes rule is used to compute this posterior probability and the most probable word sequence becomes:

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

p(X), the probability of the speech utterance is independent of the sequence of words *W*, thus it can be ignored. Consequently, Equation 2.2 becomes:

$$W^* = \arg\max_W p(X \mid W)p(W) \tag{2.3}$$

Equation 2.3 exhibits two interesting factors which can be directly estimated. The initial problem (of estimating the word sequence given the speech utterance) has now been split into two simpler problems: a) the estimation of the prior probability of the word sequence *p(W)* and b) the estimation of the likelihood of the acoustic data given the word sequence *p(X|W)*. The probability of the word sequence can be estimated using solely a *language model*, while the likelihood of the acoustic data given the words sequence can be computed based on an *acoustic model*. The two models can be constructed independently as shown in Figure 2.1, but will be used together to decode a speech utterance as specified in Equation 2.3. Figure 2.1 presents the architecture of an ASR system and also shows the methods and type of data required in the development phase.

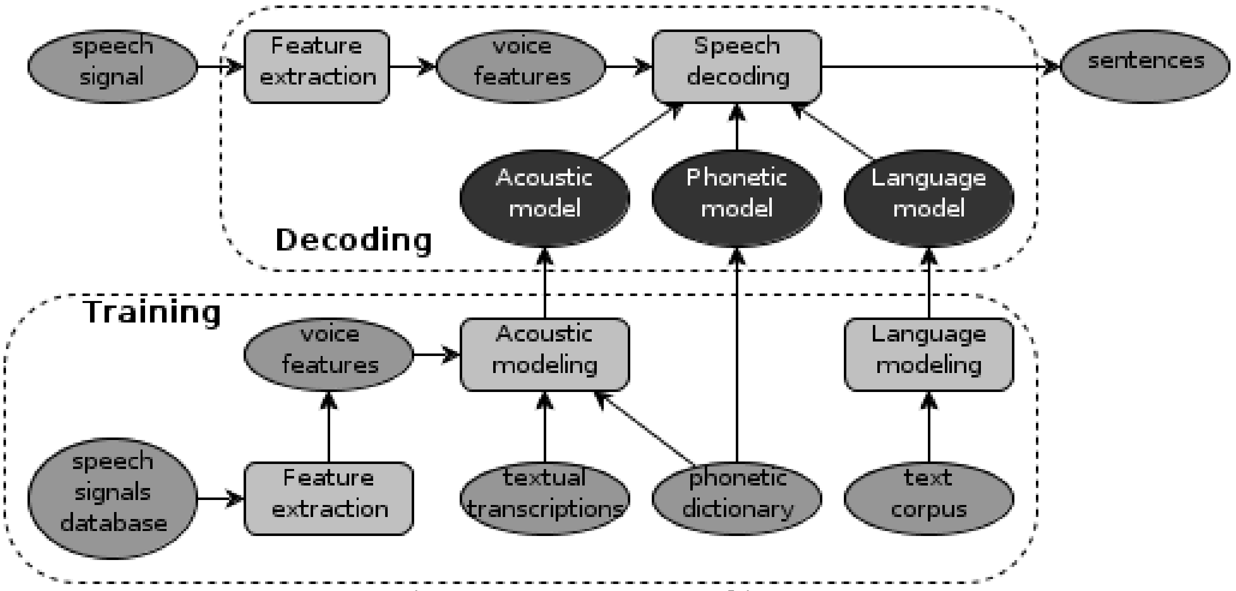


Figure 2.1 ASR system architecture

Besides the acoustic model and the language model which have been mentioned in the above formalism, the general ASR architecture also includes a phonetic model. This is due to the fact that, for large vocabulary systems, the acoustic model does not model all the words in the vocabulary (due to their high number – tens of thousands), but sub-words units such as phonemes. The phonetic model is most of the times a pronunciation dictionary which maps the words in the vocabulary to their phonetic representation.

Figure 2.1 also shows that the ASR system does not model speech directly (at the waveform level). A feature extraction block is employed to extract specific acoustic features which are further used to create the acoustic model. Consequently, the same feature extraction block is also needed and used in the recognition (or decoding) process.

This section continues with the analysis and description of the several blocks in Figure 2.1. Section 2.2 describes various language modeling issues and evaluation metrics, Section 2.3 focuses on phonetic modeling and Section 2.4 discusses several acoustic modeling issues.

2.2 Language Modeling

The language model (LM) block in Figure 2.1 is utilized during decoding to estimate the probabilities of all word sequences in the search space. In general, the purpose of a language model is to estimate how likely is a sequence of words $W = w_1, w_2, ..., w_n$, to be a sentence in the source language. The probability for such a word sequence helps the acoustic decoding in the decision process. For example, in the Romanian language these two phrases: *ceapa roşie este sănătoasă* (red onion is healthy) and *ce apar oşti ied este sănătoasă* (what appear armies kid is healthy) are acoustically very similar, but the second one does not make any sense. The role of the language model is to assign a significantly larger probability to the first word sequence and consequently help the ASR system to decide in favor of the first phrase.

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

$$p(W) = p(w_1, w_2, ..., w_n) = p(w_1)p(w_2 \mid w_1)....p(w_n \mid w_1, w_2, ...w_{n-1}) \qquad (2.4)$$

This means that the task of estimating the probability of the word sequence W is split into several tasks of estimating the probability of one word given a history of preceding words. Due to computational reasons, the history of preceding words cannot extend to include an indefinite number of words and has to be limited to m words. To put it another way, we make the assumption that only a limited number of previous words affects the probability of the next word. This leads to the conventional *n-gram* language model, which has represented the state of the art for large-vocabulary speech recognition for the past 25 years [Renalds, 2010]. Typically, m is chosen based on the amount of training data available (more training data is needed to accurately create longer history n-gram language models). Most commonly, trigram language models are used. They consider a two-word history to predict the third word. This requires the collection of statistics over sequences of three words, so-called 3-grams (trigrams). Language models may also be estimated over 2-grams (bigrams), single words (unigrams), or any other order of n-grams.

2.2.1 N-gram models construction

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

$$p(w_j \mid w_i) = \frac{count(w_i, w_j)}{\sum_w count(w_i, w)} \qquad (2.5)$$

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

$$p(w_k \mid w_i, w_j) = \frac{count(w_i, w_j, w_k)}{\sum_w count(w_i, w_j, w)} \qquad (2.6)$$

A large amount of training data (typically hundreds of millions or even billions of words) is needed to accurately estimate these probabilities. Also, higher order n-gram language models require larger amounts of training data. The problem of data sparseness, which is a typical problem for any statistical system, has to be taken into account and is addressed in the next subsections.

2.2.2 Addressing the data sparseness problem

2.2.2.1 Count Smoothing Methods

One of the key problems in n-gram modeling is the inherent data sparseness of real training corpora. Regardless how large the training corpus is, there will be n-grams which will not be seen within it, but may appear in the evaluation or test corpus. In this extreme case, the probability assigned to the unseen n-gram, given the maximum likelihood estimation equation 2.5 (or 2.6) is 0. Besides this case, there are other n-grams which occur only very few times (less than ten times) in the training corpus. Moreover, this problem becomes more severe when higher order n-grams are employed. In all these cases the probabilities which were estimated based on the empirical counts that are observed in the training corpus, are very rough estimates and need to be adjusted.

The methods involved in the adjustment process are called smoothing methods. They basically subtract probability mass from seen n-grams and redistribute it to unseen n-grams. There are several smoothing methods which tend to particularize the redistribution of probability mass given some specific reasons.

The most basic smoothing method is called *add-one smoothing*. It simply adds a fixed number (for example 1) to every n-gram count. This means that even n-grams which do not appear in the training corpus, but are made up of words in the vocabulary, will be assigned non-null probabilities. Analyzing the newly assigned probabilities, we quickly notice that add-one smoothing gives undue credence to n-grams that do not appear in the training corpus [Koehn, 2010]. One simple remedy would be to add a smaller number α, instead of 1 (a method called *add-α smoothing*). This number, α, will have to be empirically estimated on a held-out corpus.

Deleted interpolation smoothing tries to adjust the actual n-gram counts by answering the question: “If we observe an n-gram c times in the training corpus, how often do we expect to see it in a real application (in the evaluation corpus, for example)?”. This method splits the training corpus into two parts and uses one part to estimate n-gram counts and the second part to answer to the above question. Secondly, by switching the roles of the two parts and interpolating the results, this method comes up with better expected counts than the add-α smoothing method.

Another smoothing method, *Good-Turing*, uses the actual counts (c) and the count-of-counts statistics (N_c is the number of n-grams which occur c times in the training corpus) to adjust the counts (c^*) for all seen and unseen n-grams:

$$c^* = (c+1)\frac{N_{c+1}}{N_c} \tag{2.7}$$

The Good-Turing method provides a principled way to adjust counts, but is not very reliable for large c, for which N_c is typically 0. This drawback can be solved by simply not adjusting the counts for frequent n-grams. [Koehn, 2010] compares these smoothing methods and concludes that, on a particular analyzed corpus, the Good-Turing method obtains the best results, with the deleted interpolation method following closely.

2.2.2.2 Back-off Methods

A second approach to solve data sparseness is to use several language models, which have particular advantages, to create an interpolated language model that may benefit from all its constitutive parts. For example, higher order n-grams may provide valuable additional context, but lower order n-grams are more robust. If several orders (1, 2 and 3) n-gram language models p_n have been already built, an interpolated language model p_I can be constructed by linearly combining them:

$$p_I(w_3 \mid w_1, w_2) = \lambda_1 p_1(w_3) \times \lambda_2 p_2(w_3 \mid w_2) \times \lambda_3 p_3(w_3 \mid w_1, w_2) \qquad (2.8)$$

The λ coefficients in Equation 2.10 have to be positive, sub-unitary numbers and should sum up to 1. Depending on their ratio, the lower order LMs, or the higher order LMs can be given more credit. The coefficients tuning can be done empirically on a held-out set.

The back-off mechanism uses multi-order interpolated n-gram LMs to deal with unseen n-grams in a slightly different way than the smoothing methods. If we need to estimate the probability for an n-gram which was not seen or was rarely seen in the training corpus, a good idea would be to also take into consideration the probability assigned to this n-gram by the lower order n-gram model. The optimization problem here is to choose the right balance between the highest order n-gram models and all the lower order models (if these are to be used at all). Several back-off methods were proposed starting with the *Witten-Bell smoothing* method [Witten, 1991], which focuses on the diversity of words that follow a history. The most commonly method used today is the *Kneser-Ney smoothing* method introduced in [Kneser, 1995], which takes into account the diversity of histories for a particular n-gram. An extension to this method is the *modified Kneser-Ney smoothing* [Chen, 1998], which uses a method called absolute discounting to reduce the probability mass for seen events.

[Koehn, 2010] compares these back-off methods and concludes that, on a particular analyzed corpus, the modified Kneser-Ney method leads to a 5-10% lower perplexity (see Section 2.2.3) than all the other methods.

2.2.3 Language models evaluation metrics

The role of a language model is to predict the next word given its predecessors by taking advantage of the language redundancy. The capability of prediction should be objectively measured if we want to be able to compare different language models and eventually improve them.

2.2.3.1 Perplexity

The most common evaluation metric for a language model, when a speech recognition system is available, is the *word recognition error rate* (Section 2.5). Alternatively, without involving speech recognition systems, we can asses the prediction power of a language model by measuring the probability that the language model assigns to test word sequences. A good language model should assign a high probability to a good text and a low probability to a bad text. In this case the most common evaluation metric is the *perplexity*. Perplexity is derived from cross-entropy, a measure which can be computed given a particular language model *LM* and a particular word sequence $W = w_1, w_2, ..., w_n$, as follows:

$$H(p_{LM}) = -\frac{1}{n}\log p_{LM}(w_1, w_2, ..., w_n) = -\frac{1}{n}\sum_{i=1}^{n}\log p_{LM}(w_i \mid w_1, w_2, ..., w_{i-1}) \qquad (2.9)$$

The perplexity is derived from cross-entropy using a simple transformation:

$$PPL(p_{LM}) = 2^{H(p_{LM})} \qquad (2.10)$$

A higher perplexity on a particular word sequence means a lower capability of prediction for the language model, given the particular word sequence. In fact, the perplexity can be computed on both a test-set text and also on the training-set text and, obviously, it has slightly different meanings in these two cases. The test-set perplexity evaluates the generalization and prediction capability of the language model, while the training-set perplexity measures how the language model fits the training data, like the likelihood. It is generally true that, in the context of ASR, lower perplexity correlates with better recognition performance [Huang, 2001].

2.2.3.2 Out of vocabulary words

All the smoothing methods described above deal with n-grams which are not part of the training corpus, but are made up of words which appear in the training corpus. They cannot be used to adjust the language model to assign a non-null probability to a word which is not part of the initial vocabulary. These words are called *out of vocabulary (OOV) words* and, consequently, cannot be predicted by the language model.

These OOV words make it harder to evaluate a language model. Because their perplexity is infinite, it cannot be summed up to the other n-grams perplexities to obtain the word sequence perplexity. In this case, besides perplexity, the percentage of OOV words (among the total number of words) needs to be specified and both these metrics have to be taken into account for comparison:

$$OOV[\%] = \frac{\#OOVs}{\#words} \times 100 \tag{2.11}$$

2.2.3.3 N-gram hits

The *n-gram hits* is another metric which can be used to draw some important conclusions regarding the prediction capability of an n-gram language model. As shown in the previous sections, back-off models use more n-gram language models to address the data sparseness problem. For example, a trigram language model tries to predict the current word based on a two-preceding-words history (trigram model), but may back-off (due to insufficient data) to a one-preceding-word history (bigram model) or even to a null history (unigram model). For a trigram model, the trigram hits percentage gives a measure of how many times the model could use the full two-preceding-words history as compared to how many times the model needed to back-off to find the probability for the current n-gram:

$$trigram\ hits[\%] = \frac{\#trigram\ hits}{\#words} \times 100 \tag{2.12}$$

The n-gram hits metric is an auxiliary evaluation metric, which may be very useful to compare domain-specific language models. A higher n-gram hits correlates with better domain-adapted language models.

2.2.4 Other language model types

Besides n-gram language models which represent current state-of-the-art in language modeling, the experiments presented in this thesis also employ simpler language models such as word-loop grammars and finite state grammars.

A word-loop grammar is a model which assigns equal probabilities to all the words in the vocabulary and, implicitly, to all word sequences. This type of language model is, in fact, a unigram language model with equal (not corpus-estimated) unigram probabilities. Of course, a word-loop grammar can be successfully utilized for isolated words recognition, but is expected to have poor results in a continuous speech recognition setup.

A finite state grammar or word network grammar is a graph-based model in which the nodes are words and the directed links represent allowable word transitions. A finite state grammar explicitly specifies all the allowable sequences of words for a given task. Particular costs can be assigned to the existing links, thus specifying different probabilities for the allowable word sequences. If the task is relatively small (digits recognition, phone dial, menu browsing, etc.) than this type of language model can be successfully used. Moreover, finite state grammars can be successfully used in word spotting applications. However, this type of language models is not suitable for large-vocabulary continuous speech recognition.

2.3 Phonetic Modeling

General-purpose large-vocabulary speech recognition systems do not use words as basic speech units because a) every new ASR task comes with specific, new words for which there isn't any available training data and b) the number of different words in a language is too large. Instead of using words as basic speech units, these systems model sub-words speech units, such as phonemes. Consequently, a phonetic model is needed to link the acoustic model (which estimates phonemes acoustic likelihoods) to the language model (which estimates word sequence probabilities).

A phonetic model is usually a pronunciation dictionary that maps all the words in the vocabulary to a sequence of phones. The phonetic dictionary can be regarded as an interface between the acoustic model, which works with phones and the language model which works with words.

The development of a phonetic dictionary is an important, but difficult task. Although a manually created dictionary could be very useful and would assure a perfect phonetisation, the task is extremely time-consuming and tedious and also requires a very good knowledge of the language. Therefore, several approaches of automatically building phonetic dictionaries have been proposed and successfully used (see Section 4.1.3).

2.4 Acoustic Modeling

The previous section has argued that state-of-the-art continuous speech recognition systems do not estimate directly the likelihood of the acoustic data for a given word sequence ($p(X|W)$ – Section 2.1). Instead they estimate the likelihood of smaller speech units, most commonly phones. Consequently, the acoustic model consists of a set of phones models which are linked, during the decoding process, to form words models and eventually word sequences models (which are finally used to estimate $p(X|W)$). This generative approach has been proven to be very well served by the Hidden Markov Model (HMM) mathematical apparatus [Baker, 1975; Poritz, 1988; Rabiner, 1989; Jelinek, 1998].

HMMs are probabilistic finite state machines, which may be combined hierarchically to construct word sequence models out of smaller units. In large-vocabulary speech recognition systems, word sequence models are constructed from word models, which in turn are constructed from sub-word models (typically context-dependent phone models) using a pronunciation dictionary.

2.4.1 Acoustic features

HMMs do not use directly the time-domain waveform to model the speech signal. As Figure 2.1 has shown, a feature extraction block is employed to compute some feature vectors which will be eventually modeled by the acoustic model.

The speech signal is a rather un-stationary signal, therefore a spectral analysis cannot be done on the whole time-domain waveform, but only on short (20ms to 30ms), quasi-stationary frames. These frames are typically generated every 10ms (thus consecutive 25ms frames would overlap by 15ms). Each frame is multiplied by a window function. The window function is needed to smooth the effect of using a finite-sized segment for the subsequent feature extraction by tapering each frame at the beginning and end edges. As most features are spectral in nature, the Fourier Transform is employed and the multiplicative effect of the window function in the time domain is convolutive in the spectral domain. A tapered window function creates a smoother and less distorted spectrum. Without a specified window function the default arising from the framing operation is that of a rectangular window effect which will generate undesirable spectral artifacts. For the windowing process, the Hamming window is the most popular.

The initial time-domain speech waveform is transformed by the framing and the windowing processes into a time-domain sequence of quasi-stationary frames. Several types of speech features, which can be extracted out of these frames with the purpose to model speech, have been proposed over the past three decades. The most commonly used acoustic features are perceptual cesptral features such as the Mel-Frequency Cepstrum Coefficients (MFCCs) introduced by [Davis, 1980] and the Perceptual Linear Prediction (PLP) coefficients introduced by [Hermasky, 1990]. A particular advantage of cepstral representations compared with spectral representations is the de-correlation of cepstral coefficients, compared with the high correlations observed between neighboring spectral coefficients.

MFCCs are based on the log spectral envelope of the speech signal, transformed to a non-linear frequency scale that roughly corresponds to that observed in the human auditory system. This representation is smoothed and orthogonalized by applying a discrete cosine transform, resulting in a cepstral representation. The MFCCs success arises from the use of perceptually based Mel-spaced filter bank processing of the Fourier Transform and the particular robustness and the flexibility that can be achieved using the general cepstral analysis. The MFCCs are computed as presented is Figure 2.2.

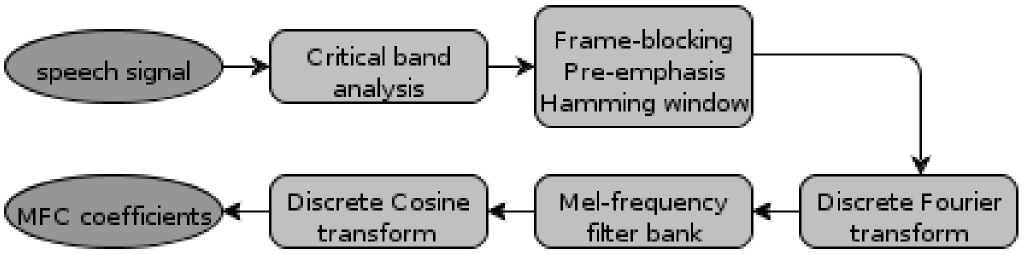

speech signal
Critical band analysis
Frame-blocking Pre-emphasis Hamming window
Discrete Fourier transform
Mel-frequency filter bank
Discrete Cosine transform
MFC coefficients

Figure 2.2 Analysis block diagram for MFCC feature vectors

Perceptual linear prediction (PLP) includes an auditory-inspired cube-root compression and uses an all-pole model to smooth the spectrum before the cepstral coefficients are computed. The PLP analysis is an extension of the Linear Prediction Coding (LPC) technique, but it is more effective because it takes advantage of some characteristics derived from the psycho-acoustic properties of the human ear [Hermansky, 1990]. These characteristics are modeled by a filter-bank. The PLP coefficients are obtained as presented is Figure 2.3.

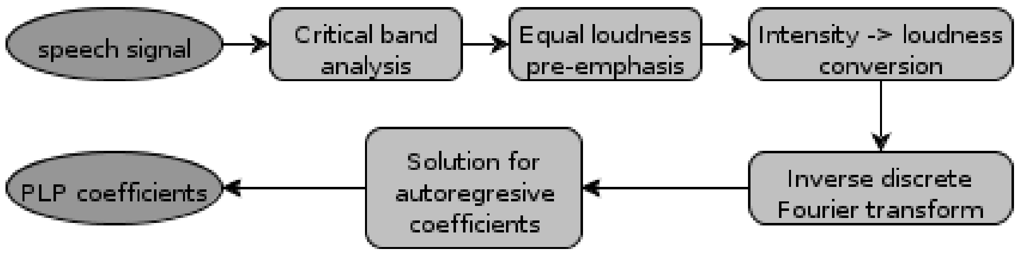

speech signal
Critical band analysis
Equal loudness pre-emphasis
Intensity -> loudness conversion
Inverse discrete Fourier transform
Solution for autoregresive coefficients
PLP coefficients

Figure 2.3 Analysis block diagram for PLP feature vectors [Hermansky, 1990]

Even though each feature set (MFCC or PLP) is computed on a short frame of speech signal, it is well known that information embedded in the temporal dynamics of the features is also useful for recognition. Typically two kinds of dynamics have been found useful for speech recognition: a) velocity of the features (known as delta features), which is determined by its average first-order temporal derivative and b) acceleration of the features (also known as delta-delta features), which is determined by its average second-order temporal derivative. Moreover, the total log energy of the feature and its derivatives have been proven to be useful for speech recognition.

Consequently, speech recognition accuracy is substantially improved if the feature vectors are augmented with the first and second temporal derivatives of the acoustic features, thus adding some information about the local temporal dynamics of the speech signal to the feature representation [Furui, 1986]. Most commonly, ASR systems use a 39-dimensional feature vector, corresponding to twelve MFCCs plus energy, along with their first and second temporal derivatives.

2.4.2 The HMM/GMM framework

The previous section has detailed the features which are generally extracted out of the speech signal for further modeling (in the training phase) or for speech recognition (in the decoding phase). The state-of-the-art approach for modeling basic speech units (typically sub-word units such as phones) makes use of the HMM/GMM framework.

2.4.2.1 HMM definition

An HMM is a probabilistic finite state automaton, consisting of a set of states connected by transitions, in which the state sequence is hidden. Instead of observing the state sequence, a sequence of acoustic feature vectors is observed, generated from a probability density function (pdf) attached to each state. This is why the Markov process is considered to be "hidden" – the state sequence is not directly available to the observer. This type of Markov process has been proven to be a very good model of speech.

A more detailed representation of an HMM is presented in Figure 2.4. As the figure shows, an HMM is characterized by these parameters:

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

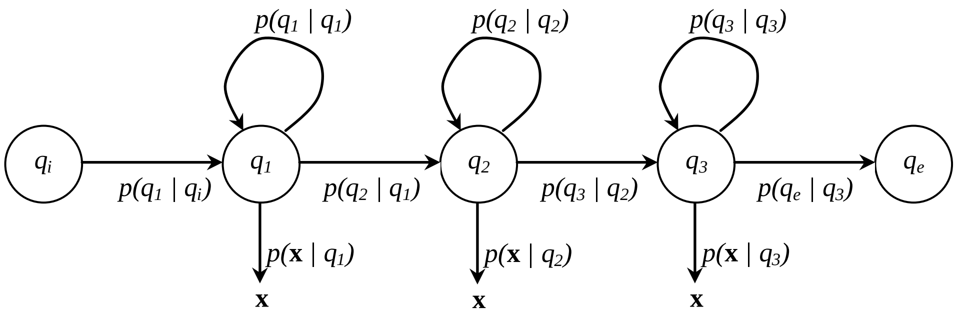


Figure 2.4 HMM representation as a parameterized stochastic finite state automaton

Although the definition of an HMM allows the transition from any state to any other state, in speech recognition the models are created to disallow arbitrary transitions, just as Figure 2.4 shows. This is because it is important and useful to model the sequential nature of speech, placing strong constrains on transitions backward or skipping transitions. Except in unusual cases, HMMs for speech disallow transitions to earlier states in the model. This kind of feed-forward HMM structure is called Bakis network. The most common model used for speech is even more constrained, allowing a state to transition only to itself (self-loop) or to a single succeeding state. The use of self-loops allows a sub-phonetic unit to repeat so as to cover a variable amount of the acoustic input.

The observation likelihood of a state q_i can be regarded as a discrete function if there is only a finite number of possible observations x_t. In the general case, the acoustic features, which are, in fact, the output of the HMM, may have a wide range of real values. Therefore, the observation likelihoods are discrete functions only in a simplifying approach, but in the general case they are probability density functions (pdfs). One of the most popular forms of output pdf for a state q_i is a d-dimensional Gaussian, parameterized by a mean vector μ_i and a covariance matrix Σ_i:

$$b_i(x) = p(x \mid q_i) = \mathrm{N}(x; \mu_i, \Sigma_i) = \frac{1}{(2\pi)^{d/2} \left|\Sigma_i\right|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu_i)^T \Sigma_i^{-1} (x-\mu_i) \right) \quad (2.13)$$

For a typical acoustic vector comprising 12th-order MFCCs plus energy, with first and second temporal derivatives, *d* equals 39.

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

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

These two assumptions may lead to an unrealistic model of speech, but they are needed due to the mathematically and computationally simplifications they bring. The estimation and decoding problems cannot be addressed, or can be addressed in a very complicated way without these assumptions. Nevertheless, the last two decades of HMMs success in speech signal modeling prove that these "limitations" are not so important.

2.4.2.2 Evaluation, Decoding and Estimation

Acoustic modeling using HMMs has become the dominant approach thanks to the development of various algorithms which enable some key computations to be carried out efficiently. These algorithms are based on the Markov and observation independence assumptions. To determine the overall likelihood of an observation sequence $X = (x_1, x_2, ..., x_t, ..., x_T)$ being generated by an HMM, it is necessary to sum over all possible state sequences $q_1q_2...q_T$ that could result in the observation sequence *X*. Rather than enumerating each sequence, it is possible to compute the likelihood recursively, using the *Forward algorithm*. The key to this algorithm is the computation of the forward probability $\alpha_t(q_j) = p(x_1, ..., x_t, q_t = q_j \mid \lambda)$, the probability of observing the observation sequence $x_1, ..., x_t$ and being in state q_j at time *t*. The Markov assumption allows this to be computed recursively using a recursion of the form:

$$\alpha_t(q_j) = \sum_{i=1}^{N} \alpha_{t-1}(q_i) a_{ij} b_j(x_t) \quad (2.14)$$

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

The decoding and evaluation problems can be solved using the Forward and Viterbi algorithms, given that a set of HMMs is available. In order to obtain a set of trained models we need to estimate the parameters of an HMM: the transition probabilities and the parameters of the output pdf (mean vector and covariance matrix in the case of a Gaussian pdf). The most straightforward criterion to use for parameter estimation is maximum likelihood (ML), in which the parameters are set so as to maximize the likelihood of the model generating the observed training data. Other training criteria may be used, such as maximum a posteriori (MAP) or Bayesian estimation of the posterior distribution, and discriminative training. Maximum likelihood training can be approximated by considering the most probable state–time alignment, which may be obtained using the Viterbi algorithm. Given such an alignment, maximum likelihood parameter estimation is straightforward: the transition probabilities are estimated from relative frequencies, and the mean and covariance parameters from the sample estimates. However, this approach to parameter estimation considers only the most probable path, whereas the probability mass is in fact factored across all possible paths. Exact maximum likelihood estimation can be achieved using the *Forward–Backward* or *Baum–Welch algorithm* [Baum, 1972], a specialization of the *expectation-maximization* (EM) *algorithm* [Dempster, 1977]. Each step of this iterative algorithm consists of two parts. In the first part (the E-step) a probabilistic state–time alignment is computed, assigning a state occupation probability to each state at each time, given the observed data. Then the M-step estimates the parameters by an average weighted by the state occupation probabilities. The EM algorithm has been shown to converge in a local maximum of the likelihood function. The key to the E-step lies in the estimation of the state occupation probability, $\gamma_t(q_j) = P(q_t = q_j \mid X, \lambda)$, the probability of occupying state q_j at time t given the sequence of observations. The state occupation probabilities can also be computed recursively:

$$\nu_t(q_j) = \frac{1}{\alpha_T(q_E)}\alpha_t(q_j)\beta_t(q_j) \tag{2.15}$$

where $\alpha_t(q_j)$ is the forward probability for state q_j at time t, $\beta_t(q_j) = p(x_{t+1}, x_{t+2}, x_T \mid q_t = q_j, \lambda)$ is called the backward probability, and $\alpha_T(q_E)$ is a normalization factor (the forward probability for the end state q_E at the end of the observation sequence, time T). The backward probabilities are called so because they may be computed by a recursion that goes backwards in time.

The output pdfs are the most important part of this model, and restricting them to single Gaussians results in a significant limitation on modeling capability. In practice, Gaussian mixture model (GMMs) are used as output pdfs. A GMM is a weighted sum of Gaussians:

$$b_i(x) = p(x \mid q_i) = \sum_{k=1}^{K} c_{ik}\mathrm{N}(x; \mu_{ik}, \Sigma_{ik}) \tag{2.16}$$

where we have a mixture of K Gaussian components, with mixture weights c_{ik}, for every HMM state. Training a GMM is analogous to HMM training: for HMMs the state is a hidden variable, for GMMs the mixture component is a hidden variable. Again the EM algorithm may be employed, with the E-step estimating the component occupation probabilities, and the M-step updating the means and covariances using a weighted average.

2.4.3 Speech units selection, context-dependency and clustering

We have already argued that, generally, large-vocabulary continuous speech recognition systems model sub-word units using HMMs. Still, there are several issues to be discussed when dealing with selecting the type of sub-word speech units. The authors of [Huang, 2001] see three high-level features that a proper speech unit must have:

- The unit should be *accurate*, to represent the acoustic realization that appears in different contexts.
- The unit should be *trainable* (there should be enough data to estimate the parameters of the unit).
- The unit should be *generalizable*, so that any new word can be derived from a predefined unit inventory for task-independent speech recognition.

Given this, it is even clearer why words cannot be used as basic speech units in large-vocabulary systems: they are neither trainable, nor generalizable.

Alternatively, phones can be chosen as basic speech units. Unlike word models, phonetic models provide no training problem because sufficient occurrences for all phones can be found in just a couple thousand phrases. Moreover, they are also vocabulary independent by nature and can be trained on one task and tested on another. Thus, phones are more trainable and generalizable. However, the phonetic model is inadequate because it assumes that a phoneme is identical in any context. Although we may try to say each word as a concatenated sequence of independent phonemes, these phonemes are not produced independently, because our articulators cannot move instantaneously from one position to another. Thus, the realization of a phoneme is strongly affected by its immediately neighboring phonemes. While word models are not generalizable, phonetic models over-generalize and, thus, lead to less accurate models.

If we make units context dependent, we can significantly improve the recognition accuracy, provided there is enough training data to estimate these context-dependent parameters. Context-dependent phonemes have been widely used for large-vocabulary speech recognition, thanks to its significantly improved accuracy and trainability. A context usually refers to the immediately left and/or right neighboring phones.

A triphone model is 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 contexts, they are considered different triphones. We denote the different realizations of a phoneme with the term allophones. Triphones are an example of allophones.

Triphone models are powerful because they capture the most important co-articulation effects. They are generally much more consistent than context-independent phone models. However, as context-dependent models generally have increased parameters, trainability becomes a challenging issue. We need to balance the trainability and accuracy with a number of parameter-sharing techniques.

Triphone modeling assumes that every triphone context is different. Actually, many phones have similar effects on the neighboring phones. The position of our articulators has an important effect on how we pronounce neighboring vowels. It is desirable to find instances of similar contexts and merge them. This would lead to a much more manageable number of models that can be better trained.

The trainability and accuracy balance between phonetic and word models can be generalized further to model sub-phonetic events. In fact, both phonetic and sub-phonetic units have the same benefits, as they share parameters at unit level. This is the key benefit in comparison to the word units. Papers by [Bahl, 1991; Hon, 1991; Hwang, 1991; Lee, 1988; Young, 1993] provide examples of the application of this concept to cluster hidden Markov models. For sub-phonetic modeling, we can treat the state in phonetic HMMs as the basic sub-phonetic unit. Hwang and Huang further generalized clustering to the state-dependent output distributions across different phonetic models [Hwang, 1991]. Each cluster thus represents a set of similar Markov states and is called a senone [Hwang, 1993]. A sub-word model is thus composed of a sequence of senones after the clustering is finished. The optimal number of senones for a system is mainly determined by the available training corpus and can be tuned on a development set.

Each allophone model is an HMM made of states, transitions, and probability distributions. To improve the reliability of the statistical parameters of these models, some distributions can be tied. For example, distributions for the central portion of an allophone may be tied together to reflect the fact that they represent the stable (context-independent) physical realization of the central part of the phoneme, uttered with a stationary configuration of the vocal tract. Clustering at the granularity of the state rather than the entire model can keep the dissimilar states of two models apart while the other corresponding states are merged, thus leading to better parameter sharing. This is one of the key solutions to create trainable context-dependent phonetic or sub-phonetic units.

In practice, senone models significantly reduce the word recognition error rate in comparison with the model-based clustered triphone models [Huang, 2001]. It is the senonic model's significant reduction of the overall system parameters that enables the continuous mixture HMMs to perform well for large-vocabulary speech recognition [Hwang, 1993].

2.4.4 Conclusion

To summarize the section on acoustic modeling we need to say that state-of-the-art large-vocabulary speech recognition systems use Bakis-type HMMs with GMMs as output pdfs to model speech units such as context-dependent phones (triphones) or senones. The HMMs model these speech units using perceptual acoustic features (MFCCs or PLP coefficients) extracted out of the original time-domain speech signal.

The Baum-Welch algorithm is used to estimate the HMM parameters. The Viterbi algorithm is used to decode the speech data: to find the most probable sequence of states given the acoustic features observations.

There is typically one HMM per speech unit and all these basic HMMs can be concatenated to form words HMMs, which can be further concatenated to form word sequences HMMs. This is the mechanism which allows us to eventually estimate the probability of a sequence of words given the initial speech data.

2.5 ASR Evaluation

The task of evaluating a speech recognition system involves comparing a reference (or correct) word sequence with the hypothesis word sequence returned by the system. The standard evaluation metric for comparing the two word sequences is the word error rate (WER). Given the correct word sequence, the first step in computing the word error rate is to compute the minimum edit distance in words between the reference and the hypothesized sequences. This is usually done using the dynamic programming algorithm called Dynamic Time Warping (DTW) given some standard weights for the three types of errors which can occur: insertions, deletions and substitutions. After the alignment and based on these three types of errors, the word error rate is computed using the following formula:

$$WER[\%] = \frac{\#\,Insertions + \#\,Substitutions + \#\,Deletions}{\#\,Words\ in\ reference\ transcription} \times 100 \tag{2.17}$$

Note that sometimes the word error rate can be greater than 100% because the above equation also includes the number of insertions.

In some applications a second evaluation metric, the sentence error rate (SER), might also be important. The sentence error rate is based on the word error rate and can be computed as follows:

$$SER[\%] = \frac{\#\,Sentences\ with\ at\ least\ one\ word\ error}{\#\,Sentences\ in\ the\ reference\ transcription} \times 100 \tag{2.18}$$

2.6 Speech Recognition Tools

The most popular and commonly used development and speech recognition tools are the Hidden Markov Model Toolkit (HTK) [Young, 1994] and the CMU Sphinx [Lee, 2002]. Both of them are open source toolkits and are available online. They offer the possibility of developing speaker-independent, large-vocabulary, continuous speech recognition systems in any language. HTK was intensively used during the last ten years, but lately Sphinx became more popular, both in the scientific community and also in the industry thanks to its free license for commercial applications. The speech recognition performance of the two toolkits has been compared by some studies. They generally conclude that similar systems developed with the two toolkits have a similar performance, but the acoustic modeling performed by Sphinx is slightly better [Samudravijaya, 2003].

The evaluation of speech recognition systems can be done using the HTK/Sphinx built-in evaluation tools, but the most commonly used evaluation toolkit is the NIST SCTK (National Institute of Standards and Technologies - Scoring Toolkit) [NIST, 2005]. The scoring tool performs the hypothesized-reference alignment, computes the word error rate and provides a series of other useful statistics. Among these, the most important ones are the words confusion matrices, showing which words are mostly misrecognized for others. 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.

The language modeling experiments presented in this thesis have been performed using the SRI-LM Toolkit [Stolcke, 2002]. There are also several other open-source toolkits which provide language modeling facilities. Among these the CMU-SLM (Carnegie Mellon University – Statistical Language Modeling) toolkit is the second most commonly used.

CHAPTER 3

STATISTICAL MACHINE TRANSLATION MAIN PRINCIPLES

The field of machine translation can be traced back to the early 1950s, in the era of code-breaking, when the translation process was firstly regarded as decoding an *encrypted message*. In fact, this principle is still valid today: we are still talking about *decoding a foreign language* and we are still using modeling techniques such as *the noisy-channel model*. Moreover, it appears that machine translation funding is basically driven by the same motivation: governments invest large amounts of money into translating the languages of countries which are considered to be a threat to national security.

The first approaches used basic rule-based methods to translate words from the source language to the target language. These methods evolved into more complex techniques that utilized morphological and syntactic information. During the 1980s the *interlingua* concept, that aimed to represent meaning in a language-independent manner, was introduced and explored. These systems required more sophisticated grammars, which could be used to analyze and to conceptualize the text in the source language and in the end to generate the text in the target language.

Given that one of the main applications of machine translation is helping human translators, a new idea emerged: the usage of *translation memories* in so-called *example-based translation* systems. These systems exploit the already available and growing parallel corpora created by human translators. This was the first data-driven approach to be proposed as a viable solution to machine translation.

The success of statistical methods in the field of speech recognition triggered their application in machine translation. The first *statistical translation models* were proposed in the labs of IBM Research in the late 1980s. Although the statistical approach was very intriguing, the scientific community continued to focus on syntax-based and interlingua systems throughout the 1990s. The growth of the Internet and consequently the increasing availability of text resources along with the development and openness of several standard tools implementing the IBM translation models led to the adoption of statistical machine translation (SMT) as the de facto approach around the year 2000. Since then a large number of academic and commercial research labs have developed statistical machine translation systems, while several large companies are on the market with such competitive systems.

3.1 The Statistical Machine Translation Formalism

The state-of-the-art systems in statistical machine translation are based on phrase translation models. The notion of phrase does not refer to a group of sentences as in the strictly linguistic sense, but to an expression, to a sequence of words. The phrase-based translation model was introduced in [Koehn, 2003]. These systems are the successors of the systems based on word translation models, developed by IBM.

The machine translation task can be formulated in a probabilistic manner as follows:

What is the most likely translation sentence e in the target language E,
given the input sentence f, in the source language F?

The formal representation uses the argmax function, which selects the argument that maximizes the translation probability:

$$e^* = \arg\max_{e} p(e \mid f) \tag{3.1}$$

Equation 3.1 specifies the most probable translation as the one with the highest posterior probability, given the input sentence in the foreign language. In order to factor in a language model we can use the Bayes rule similarly as in the speech recognition case:

$$e^* = \arg\max_{e} p(e \mid f) = \arg\max_{e} \frac{p(f \mid e)\,p(e)}{p(f)} = \arg\max_{e} p(f \mid e)\,p(e) \tag{3.2}$$

Although this derivation does not simplify the problem (it just changes the translation direction from *p(e|f)* to *p(f|e)*), the language model can help a lot in obtaining a good translation, by assuring the fluency of the output sentence.

In the case of a phrase-based translation model, the foreign sentence *f* is further split into *I* phrases f_i. The segmentation of the foreign sentence is not explicitly modeled: any segmentation is equally likely. Using this segmentation, the probability of the foreign sentence given the target sentence *p(f|e)* can be further decomposed into:

$$p(f \mid e) = \prod_{i=1}^{I} \phi(f_i \mid e_i)\; d(skippedWords_i) \tag{3.3}$$

Equation 3.3 highlights the two components which are used to estimate the likelihood of a phrase translation: a) the phrase translation table ($\phi(f_i|e_i)$) and b) the reordering model (*d*). Reordering is handled by a distance-based reordering model. The reordering distance is the number of words skipped (either forward or backward) when taking foreign words out of sequence. If two phrases are translated in sequence, then the lowest reordering cost (*d(0)*) is applied.

The reordering model assigns a lower cost if the phrases are translated in sequence and a higher cost, dependent on the number of skipped words, if the phrases need to be reordered. In other words the reordering model assures that movements of phrases over large distances are mode expensive than shorter movements or no movements at all.

The following equation integrates all the components that we have discussed so far (the translation table, the reordering model and the language model):

$$e^* = \arg\max_e \prod_{i=1}^{I} \phi(f_i \mid e_i)\ d(skippedWords_i)\ \prod_{i=1}^{|e|} p_{LM}(e_i \mid e_1 \ldots e_{i-1}) \qquad (3.4)$$

The three components contribute to producing the best possible translation by assuring that:

- the foreign phrases match the words in the target language (ϕ);
- the phrases are reordered appropriately (d);
- the translated phrase is fluent (p_{LM}).

The three contributions might be disproportionate. For example, the output could be very fluent, but it might not be translated very well. Or, the reordering could be too strict. To compensate all these things, the contribution of the models could be scaled using some weighting factors: λ_ϕ, λ_d, λ_{LM}. The weights can be taken into account as shown in the next equation:

$$e^* = \arg\max_e \prod_{i=1}^{I} \phi(f_i \mid e_i)^{\lambda_\phi}\ d(skippedWords_i)^{\lambda_d}\ \prod_{i=1}^{|e|} p_{LM}(e_i \mid e_1 \ldots e_{i-1})^{\lambda_{LM}} \qquad (3.5)$$

Figure 3.1 presents the architecture of a statistical machine translation system and also includes the operations which have to be performed to construct the three components of the system.

As the figure shows and as previously discussed, the text in the source language is translated in the foreign language based on a phrase translation model, a phrase reordering model and a language model for the target language. The resources needed to train these components are: a sentence-aligned parallel corpus (text paired with its translation) and a plain text corpus for the target language.

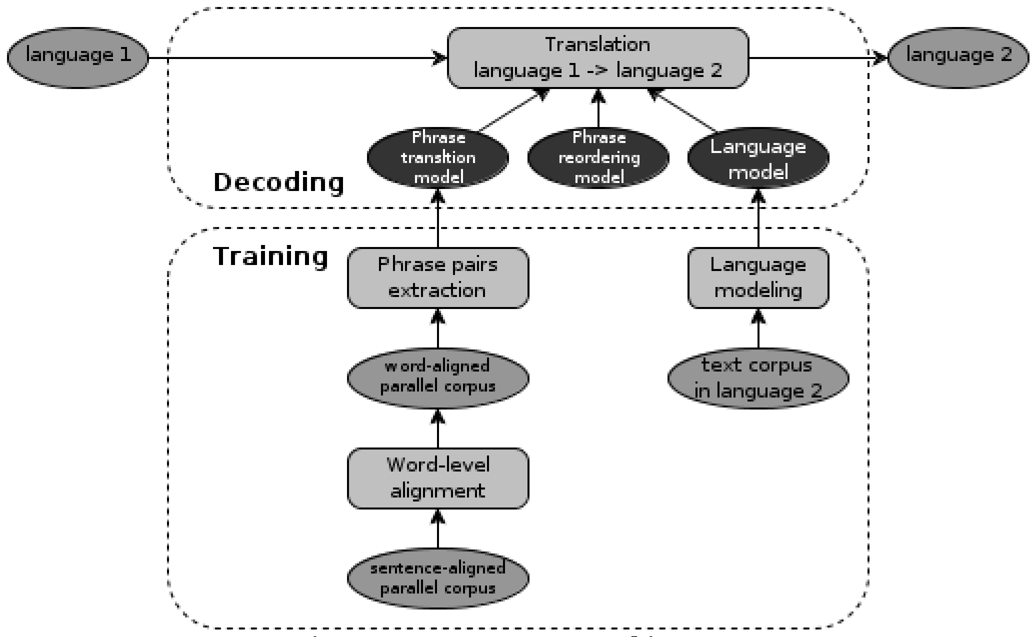


Figure 3.1 SMT system architecture

The first step in building the phrase translation table is to word-align the initially sentence-aligned parallel corpus. Generally, this word alignment process is performed using word-based translation models. Word-alignment will be detailed in Section 3.2. Second, using the word-aligned parallel corpus, phrase pairs that are consistent with this alignment are extracted and used to estimate the probabilities within the phrase translation table. This process is detailed in Section 3.3.

As already discussed, the reordering model is not estimated from data. Even though reordering probabilities could also be learn from the parallel corpus, this is not typically done for phrase-based models. Instead, reordering is handled by a predefined model. Some specific issues regarding the distance-based reordering model will be presented in Section 3.4.

The construction of a general n-gram language model has been already discussed in the context of automatic speech recognition in Section 2.2. All these things are still valid in the case of machine translation: typically, trigram language models are used to assure the fluency of the translated texts.

3.2 WORD-LEVEL ALIGNMENT FOR SENTENCE-ALIGNED CORPORA

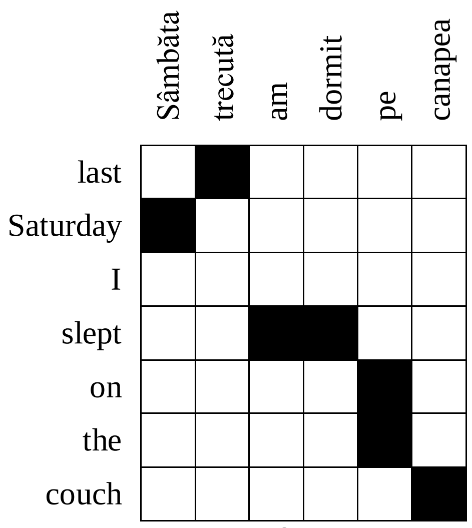
3.2.1 The task of word alignment

Word alignment is the operation which transforms a sentence-aligned parallel corpus into a word-aligned parallel corpus. It is the first operation that has to be employed in order to build a phrase translation table. Let us suppose we have the following pair of sentences in English, respectively Romanian:

English sentence: *Last Saturday I slept on the couch.*

Romanian sentence: *Sâmbăta trecută am dormit pe canapea.*

Given this pair of phrases, the task of word alignment can be seen as finding a set of alignment points between the English words and the Romanian words. Figure 3.2 presents an alignment provided by a human translator. The English words (displayed on the lines of the matrix) are aligned to the Romanian words (listed in the columns) as indicated by the filled cells in the matrix.



	Sâmbăta	trecută	am	dormit	pe	canapea
last		■				
Saturday	■					
I						
slept			■	■		
on					■	
the					■	
couch						■

Figure 3.2 An example of word alignment

We can observe that there are words which align to multiple other words (for example, the English word *slept* aligns to the Romanian words *am dormit*), words which align to a single word and some other words which are not aligned at all (for example the English word *I* is implied in the Romanian sentence).

Although in most of the cases the word alignment can be easily accomplished, there are cases in which even a human translator can be confused. For example, in our sentence pair the pronoun *I* is implied in Romanian because the verb (*am dormit*) is properly inflected (first person). Consequently, we could argue that the word *I* could be aligned to the word *am* (which contains the information about inflection).

There are other, more problematic cases in which the word alignment is not obvious at all: the idiomatic expressions. For example consider the case in Figure 3.3. The English idiomatic expression *kicked the bucket* and the Romanian expression *a dat ortul popii* have the same meaning: *died*. The two expressions can only be aligned at the phrase-level, because outside this context the English verb *kicked* is not a good translation for the Romanian *a dat*. It's the same case with the English noun *bucket*, which is obviously not a good translation for the Romanian *ortul popii*. In the particular case of idiomatic expressions we can only speak about *phrasal alignment*, an alignment that cannon be decomposed any further because the meaning is not preserved.

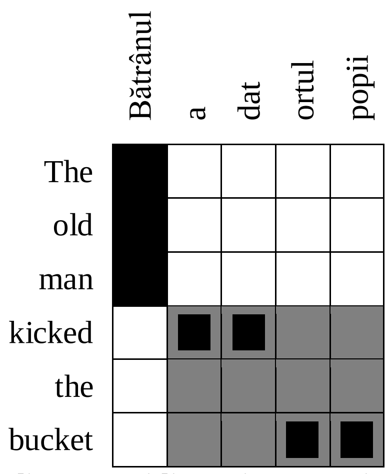


Figure 3.3 Problematic word alignment: idiomatic expressions cannot be word-aligned

Word alignment is a task that can be accomplished by the simpler and ex-state-of-the-art translation models: the word-based models developed by IBM around the 1990s.

3.2.2 IBM word-based translation models

The first statistical translation systems were based on lexical translation: the translation of isolated words, independent of their context. This approach is very similar to a common bilingual dictionary with probabilities for every translation option.

The translation model estimates a lexical translation probability distribution. The model is created using counts for every word-translation pair obtained from a word-aligned parallel corpus, by means of maximum likelihood estimation (MLE).

Let us consider an example where the source language is English and the target language is Romanian. In order to construct the probabilistic translation table for a given word (for example *car*) we need to count how many times this word is translated by different Romanian words. If we suppose that the word *car* appears 10000 times in the corpus and it is translated 7000 times by *maşină*, 1700 times by *automobil*, 1100 times by *autoturism* and 200 times by *vagon*, then the lexical translation probability distribution for this word would be the one presented in Table 3.1.

Table 3.1 Translation table for the English word car

Translation of *car*	**Count**	**Probability**
maşină	7000	0.70
automobil	1700	0.17
autoturism	1100	0.11
vagon	200	0.02

Translation tables for all the English words found in the training corpus can be created in a similar manner. Given these translation tables, an English sentence can be translated to Romanian in a word-by-word manner, using every possible translation for every word. This model assumes that the word-level translations are statistically independent and thus the translation probability of the whole sentence can be estimated only given the translation probabilities of its composing words. In order to find the best translation, all possible word alignments between the English words in the input sentence and the Romanian words in the hypotheses sentences are explored. Consequently, translating an English sentence to Romanian would implicitly generate a word alignment.

The alignment model used in IBM model 1 is somehow more complex, because it also allows for other alignments than simple word-to-word alignments. This is something natural because we already saw that a word in the source-language can be translated to more than one word in the target-language and vice-versa. Unfortunately, IBM model 1 only allows this one-to-multiple alignment in one direction: one source-language word translated into multiple target-language words. It does not allow a target-language word to stand as a translation for multiple source-language words.

IBM alignment model 1 also permits dropping words in the source-language sentence and adding words in the target-language sentence (these will be aligned to an imaginary NULL token inserted in the source sentence).

The construction of this type of translation models requires word-aligned corpora. Usually this is not a widely available resource: only sentence-aligned corpora are generally available. The training problem can be solved by the expectation maximization (EM) algorithm which works as follows:

- we first initialize the translation model with uniform probability distributions,
- second, we apply the model to the source language data (expectation step),
- third, we train the model using the reference target language data (maximization step),
- finally, we repeat steps 2 and 3 until convergence.

Generally, the EM algorithm is guaranteed to converge to a local minimum, but in the case of IBM model 1 it was mathematically demonstrated that the convergence always reaches the global minimum.

IBM model 2 uses IBM model 1 as an initialization step and further introduces an explicit word alignment model. This alignment model is also regarded as a probability distribution which is estimated using counts in the training data, in the same way (expectation maximization algorithm) as for the lexical translation probability distribution.

IBM model 3 explicitly introduces the notion of fertility. This is another probability distribution that statistically models how many target-language words are usually produced by a source-language word. Fertility explicitly deals with dropping source-language words, but cannot cope with inserting target-language words. This last issue is also dealt with in IBM model 3 by creating a special context (defined by another probability function) for inserting the NULL token in the source-language sentence.

In IBM model 4, the absolute alignment model introduced by IBM model 2 is replaced with a relative alignment model. In this model, the placement of the translation of a source-language word is typically based on the placement of the translation of the proceeding source-language word. This is motivated by the fact that large phrases tend to move together within long sentences. Words that are adjacent in the source sentence tend to be next to each other in the target sentence also. For instance, whether the 16th source-language word is translated into the 16th target-language word depends to a large degree on what happened to the 15th word.

The higher IBM models are more complex and use the lower models as initialization. Regardless of the used model, what it is clear is that these translation models can be trained solely on sentence-aligned parallel corpora. A by-product of the training process is the word-level alignment of the training corpus.

3.2.3 Word alignment based on the IBM models

As discussed in the previous section, a by-product of training the IBM translation models is that it establishes a word alignment for each sentence pair within the initially sentence-aligned corpus. However, there is one fundamental problem with the word alignment performed by the IBM translation models: the one-to-multiple alignment. Unfortunately, IBM models only allow this one-to-multiple alignment in one direction: one source-language word aligned to multiple target-language words. It is impossible to end up with an alignment of one target-language word to multiple source-language words.

However, in practice both one-to-multiple alignments are possible. In the example shown in Figure 3.2 the English words *on the* are aligned to a single Romanian word: *pe*, while the Romanian words *am dormit* are aligned to a single English word: *slept*.

To overcome this problem a simple trick is usually employed: running the IBM training in both directions. The two resulting word alignments can then be merged by taking the intersection or the union of alignment points of each alignment. This process is called *symmetrization of word alignments*. Generally, the intersection will contain reliably good alignment points (a high precision of the alignment points), but not all of them. The union will contain most of the desired alignment points (a high recall of the alignment points), but also additional faulty points. An example, using the previously used pair of English-Romanian sentences, is given in Figure 3.4. In this figure the intersection of alignment points is in grey and their union is in black.

Several methods have been developed to explore the space between the intersection of the alignments and their union. The most commonly used method [Och, 2003] explores the space between intersection and union with expansion heuristics that start with the intersection and add additional alignment points. The decision about which points to add may depend on various criteria which I will not describe any further.

The conclusion of this subsection is that the IBM word-based translation models can be successfully used to word-align an initially sentence-aligned parallel corpus.

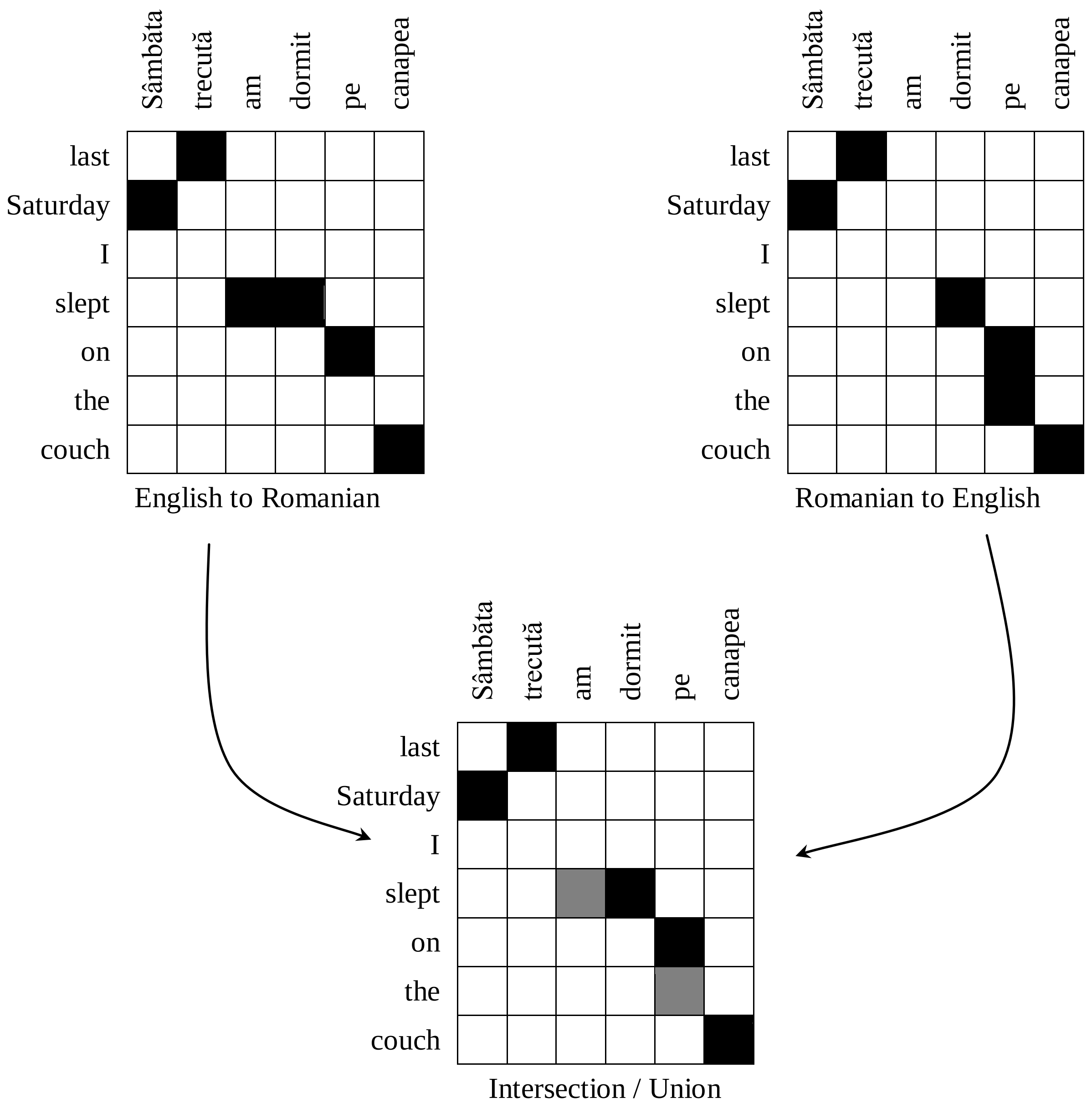


Figure 3.4 Symmetrization of IBM model alignments

3.3 The Phrase-Based Translation Model

3.3.1 The translation table

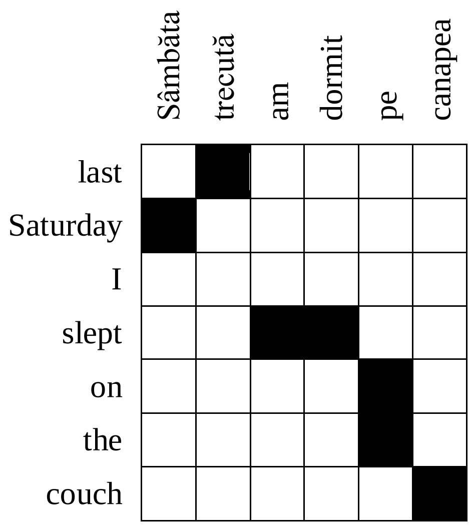
The phrase translation table is the core of the SMT system. Its translation performance is to a large degree dependent on the quality of the translation table. Several methods were proposed to extract phrase pairs from a word-aligned text corpus. The technique which is currently considered as state-of-the-art was presented in [Zens, 2002]. This method proposes to build the phrase table using only the phrases which are consistent to the word alignment: the words in a legal phrase pair are only aligned to each other, and not to outside words.

Let us consider again the example in Figure 3.2. Given this word alignment we would like to extract only the phrase pairs that are consistent with it, for example matching the Romanian phrase *pe canapea* with the English phrase *on the couch*. If we have to translate an English sentence that contains the phrase *on the couch* then we can use the evidence of this phrasal alignment to translate the phrase as *pe canapea*. Useful phrases for translation may be shorter or longer than this example. Shorter phrases occur more frequently, so they will more often be applicable to previously unseen sentences. Longer phrases capture more local context and help translate larger chunks of text at once, occasionally even entire sentences. Hence, when extracting phrase pairs, both short and long phrases should be collected, since all of them are useful.

So, the first step in creating the translation table is to extract phrase pairs based on their consistency with the word alignment. To be more specific, a phrase pair that is consistent with the word alignment is a phrase pair whose words (both source-language words and target-language words) align only to each other and not to words outside the phrase pair. Consequently, the extraction method loops over all possible phrase pairs and verifies the above constraint. Figure 3.5 reminds the word alignment in our previous example and displays the complete list of consistent phrase pairs that can be extracted from this sentence pair.

The first thing to be observed is that it is possible that for some English phrases, we are not able to extract matching Romanian phrases. This happens, for instance, when multiple English words are aligned to one Romanian word: *on the* are both aligned to the Romanian *pe*, so that no individual match for either *on* or *the* can be extracted.

This also happens when the English words align with Romanian words that enclose other Romanian words that align back to English words that are not in the original phrase. See the example of *Saturday I slept*, which aligns to *Sâmbăta ... am dormit*, words that enclose *trecută*, which aligns back to *last*. Here, it is not possible to match *Saturday I slept* to any Romanian phrase, since the only matching Romanian phrase has a gap.



consistent phrase pairs:

last – trecută

last Saturday – Sâmbăta trecută

last Saturday I – Sâmbăta trecută

last Saturday I slept – Sâmbăta trecută am dormit

last Saturday I slept on the – Sâmbăta trecută am dormit pe

last Saturday I slept on the couch - Sâmbăta trecută am dormit pe canapea

Saturday – Sâmbăta

Saturday I – Sâmbăta

I slept – am dormit

I slept on the – am dormit pe

I slept on the couch – am dormit pe canapea

slept – am dormit

slept on the – am dormit pe

slept on the couch – am dormit pe canapea

on the – pe

on the couch – pe canapea

couch – canapea

Figure 3.5 Extracted phrase pairs given the word alignment

Examples can be easily found for Romanian phrases that cannot be matched to any English phrases, due to similar reasons as mentioned above: *trecută am dormit, am, dormit*.

A second observation regards unaligned words. Unaligned English words may lead to multiple matches for Romanian phrases: for instance, *am dormit* matches with two English phrases: *I slept* and *slept*. Vice-versa if there were any unaligned Romanian words they would have led to multiple matches for English phrases.

The estimation technique for the conditional probability distributions of the phrase translation table is different than the technique which was used in the case of words translation models. The estimated probability that the source-language phrase *f* is the translation of the target-language phrase *e* is:

$$\phi(f \mid e) = \frac{count(e, f)}{\sum_{f_i} count(e, f_i)} \tag{3.6}$$

In other words, we estimate this probability by dividing the number of times the phrase pair *(e, f)* is collected to the number of times the phrase *e* is collected with any pair f_i.

3.3.2 Some basic translation model extensions

Although the translation model based on the phrase translation table appears to be very strong, there are some other aspects which have to be taken into account in order to avoid some translation errors.

First there has to be some protection mechanism against overestimated probabilities for infrequent phrases, especially if they are collected from noisy data. If, for example, a phrase pair *(e, f)* occurs only once in the training corpus then its conditional probability would be $\phi(f|e)=1$. This value is clearly an overestimate of how reliable the phrase pair is. To avoid this problem, the conditional probabilities for rare phrase pairs are decomposed into a product of the conditional probabilities for their composing word pairs. It is the same idea used in the word-based translation models: a phrase pair is decomposed into its constituent word pairs and its probability is computed based on the probabilities of the word pairs. This is basically a smoothing technique, which is also similar to the ones used in the case of n-gram language models: when the statistics for a word sequence are not reliable we back-off to shorter word sequences for which we have richer statistics and hence more reliable probability estimates. In the context of machine translation this technique is called *lexical weighting*.

Another interesting aspect which has not been considered in the basic translation model regards the number of words in the target-language sentence, or the output sentence length. One of the components of the SMT system, namely the language model, would always prefer shorter output sentences simply because fewer trigrams have to be scored. To guard against output that is too short or too long, a *word penalty*, which adds a factor *w* to each produced word, is usually introduced. A word penalty smaller than 1 favors shorter output sentences and vice-versa, if *w* is higher than 1, then longer output sentences are preferred.

One last thing that might need some tuning is the length of the segmented source-language phrases. Before actually decoding an input sentence it first has to be split into several phrases. This segmentation into more phrases is not explicitly modeled and initially all segmentations are equally likely. Of course, in the end the best input sentence segmentation will be indirectly determined based on the scores provided by the translation table, the reordering model and the language model for a given translation sentence. Nevertheless, the segmentation process can be biased towards shorter or longer phrases by introducing a *phrase penalty* tuning factor *p*. This tuning factor is called phrase penalty because the number of segmented phrases is directly linked to their length: more phrases implies shorter phrases, while fewer phrases implies longer phrases.

3.4 THE PHRASE REORDERING MODEL: SPECIFIC ISSUES

Reordering is one of the most difficult problems in machine translation. What makes it even harder is the fact that for different language pairs reordering manifests differently. There are language pairs for which restricting reordering to short local movements is sufficient for the translation of most sentences, while there are other language pairs for which reordering cannot be restricted at all. For example, the translation of the language pairs Chinese-English, French-English and Arabic-English is characterized by short local movements, while in the case of German-English and Japanese-English the movements are more ample [Koehn, 2010].

The reordering model is constructed in such a manner that it penalizes movements. In-sequence phrase translation has the smallest cost, while out-of-order phrase translation receives a smaller or a larger penalty depending on the movement amplitude. There is one model which counter-balances this behavior: the language model. If the reordering (the movements within the output sentence) produces better target-language sentences than the language model has the role of assigning a better score to the reordered sequence and hopefully to proclaim it as the winner. For example, the improvement in language model score for *Sâmbăta trecută* over *trecută Sâmbăta* (when translating the English *last Saturday*) should be much higher than the penalty involved in the movement.

For language pairs in which a good translation implies only short, local movements, a typical trigram language model works very well. However, for language pairs which have a different syntactic structure (Romanian-German, for example) the typical 3-word window used by the language model is just too small for making adequate judgments about overall grammaticalness of the sentence. In these cases other reordering mechanisms should be used.

Given the weaknesses of the reordering model, it may not come as a surprise that limiting reordering to monotone translation is not very harmful. Allowing no reordering at all has other benefits: the search problem for finding the optimal translation according to the model is reduced in complexity from exponential to polynomial, making search algorithms much faster [Koehn, 2010].

Allowing limited reordering, however, yields better translation results than allowing no reordering at all. If we permit moves within a window of a few words, we allow the local reordering required when translating Arabic–English (subject-verb, adjective-noun) or French-English (adjective-noun) [Koehn, 2010]. Since this is also something that the language model can handle, it often represents the best we can do with reordering. Larger reordering windows or completely unrestricted reordering often leads to worse translations.

3.5 DECODING A FOREIGN SENTENCE

Given the machine translation model presented in the previous sections, the task of decoding is the process of finding the best scoring translation according to this model. This is a hard problem, since there is an exponential number of choices, given a specific foreign sentence. In fact, it has been shown that the decoding problem for the presented machine translation models is NP-complete [Knight, 1999]. Consequently, examining all translation options for an input sentence and scoring them with the final goal of choosing the best translation is out of the question: this is computationally too expensive even for short sentences.

As exhaustive search is not an option, the decoding task has been tackled with various heuristic search techniques. These methods do not guarantee to find the best translation option, but try to find one as close as possible to the best. Provided that the decoding algorithm is error-prone, we distinguish two types of errors that can lead to bad translations: a) search errors – failures in finding the most probable translation according to the model and b) model errors – errors caused by a lousy phrase table, reordering model, etc.

The most popular decoding method, which is also state-of-the-art in machine translation, is the beam search algorithm, similar to the one introduced in [Jelinek, 1998] for speech recognition. This algorithm generates translation hypotheses from left to right, in sequence, marking off all the words in the source sentence. When all the words in the source-language sentence are exhausted, the various hypotheses are analyzed and the best translation, according to the translation model, is picked out.

3.5.1 The translation process

A human translator would generally translate a foreign sentence in the same manner as mentioned for the beam search algorithm. He would begin by translating the first word in the output sentence (this is not necessarily the first word in the input sentence) and would map this output word to a word in the input sentence. Afterwards, he would try to find the next best word in the translation and map it to a word in the input sentence. He would continue with translating small chunks of data until there are no more words to be translated in the input sentence. The notion of reordering is accommodated into this translation process by the possibility of picking input words out-of-sequence, while building the translation in-sequence.

Recalling the translation model described in the previous sections, we can make two important remarks: a) given an input sentence and its translation we can compute the probability of the translation using the phrase translation table ϕ, the reordering model d and the language model p_{LM} and b) the translation probability can be computed incrementally, as the translation is created, by adding in a partial cost every time a phrase is added to the translation.

To conclude: if the translation is constructed in-sequence, from left to right, then its probability can be computed incrementally every time a new phrase is added to the output sentence.

3.5.2 The Beam Search algorithm

The first step taken by the search algorithm, when it is given the task of translating a sequence of words in the source-language, is to create a complete list of translation options. This list contains all possible translations for the given sentence. Creating this list allows a quicker lookup than consulting the whole phrase translation table during decoding.

The algorithm continues by building partial translations using the various phrases within the translation options list. These partial translations are called hypotheses and store information about the translated words, the input words to which they map to, the partial score, etc. The algorithm starts with an empty hypothesis and expands it (creating a new hypothesis) by picking a phrase in the translation options list. Let us follow the example in Figure 3.6 which shows the construction of the hypotheses search graph for the English sentence *I have a green book*. We see that in this example the empty hypothesis is expanded into multiple one word hypotheses (*eu*, *mie* and *am*). These hypotheses are further expanded covering more words in the input phrase (the covered words are marked with an *x*). For example the hypothesis *eu* is expanded into *eu are* and *eu am*. This process continues until all the words in the foreign sentence are translated. In the example presented in Figure 3.6 we have three such cases: *eu am o verde carte*, *eu am o carte verde* and *am o carte verde*. These hypotheses cannot be expanded any further and represent endpoints in this search graph.

The final step in the algorithm is responsible for scoring all the endpoints in the search graph and sort them based on their score (or probability). In the end the translation option with the highest probability is proclaimed the winner.

There is one big issue regarding this search algorithm: the computational complexity. The hypotheses expansion process would end up by creating and scoring all possible translations and this makes the search heuristic computational prohibitive for any large sentence, because the search space grows exponentially with the sentence length.

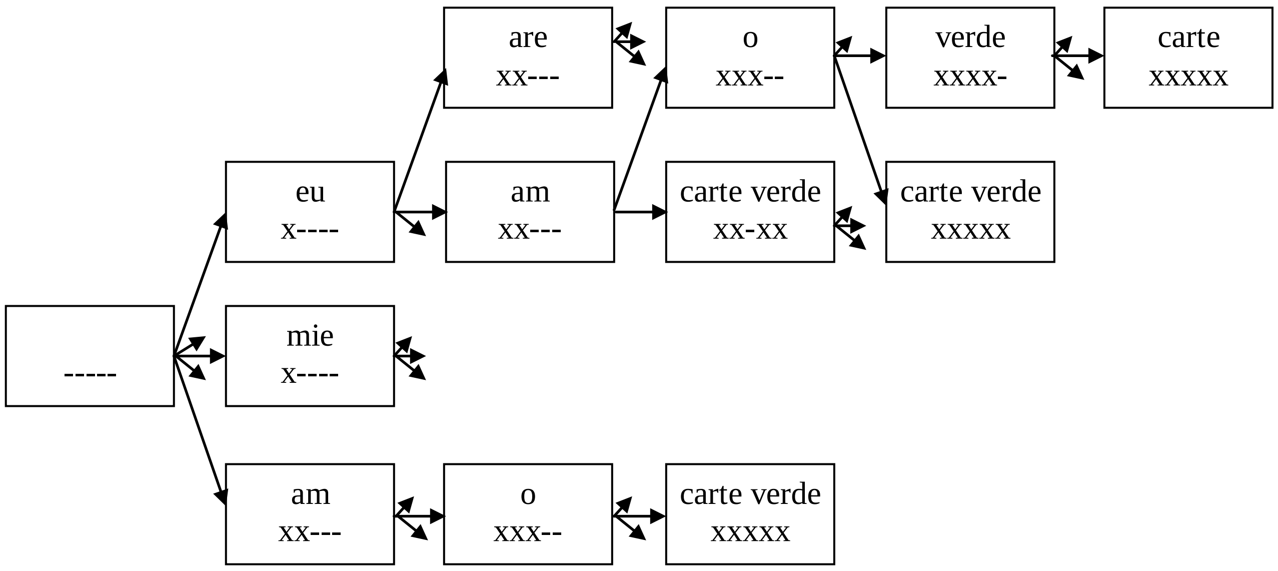


Figure 3.6 The decoding process: expanding the translation hypotheses

This computational problem is addressed in two ways: by reducing the number of hypotheses using hypothesis recombination and by pruning: deciding which low-probability hypotheses should be dropped early on, at the risk of failing to find the best translation.

The beam search algorithm organizes the hypotheses into comparable clusters called hypothesis stacks. The hypotheses are clustered based on the number of input words translated. From time to time these stacks are pruned and the worst hypotheses are discarded. Of course, a hypothesis which was pruned out at some moment could have turned out to be the best translation, but this risk has to be taken. The most popular pruning method employs a threshold by which a hypothesis is allowed to be worse than the best one in the stack. All other hypotheses are pruned out. This threshold is also denoted beam width, thus the name of the algorithm. The beam search algorithm assures that the computational complexity becomes manageable and an input sentence of any length can be decoded in a decent amount of time. In fact, the beam width or the threshold can be tuned for better performance or speed.

One last problem which must be approached when discussing comparable translation hypotheses is that of future cost estimation. A hypothesis that covers 3 input words is not necessarily comparable with a second hypothesis that covers other 3 input words, because in most of the cases there are some parts of the foreign sentence which are harder to translate and some other parts which are easier to translate. Consequently, when pruning out a hypothesis stack the score of the hypotheses has to include the partial score for the already translated words and an estimate of the future score (for translating the other parts of the sentence).

Event though the beam search algorithm is also used in speech recognition and part-of-speech tagging, all these issues that have been discussed for the machine translation version make the decoding process a lot more harder. The cause for this increased difficulty is of course machine translation’s particularity: reordering; the fact that the input can be processed out-of-order.

3.6 SMT Evaluation

When it comes to evaluating machine translation systems we find there is a huge debate regarding the best evaluation method. Human subjective evaluation is clearly the most relevant because, in the end, humans will benefit from the resulted translations. However, a system which evolves several times a day cannot be assessed fast enough and cheap enough in a subjective way. Consequently, it is very important to design automatic evaluation procedures and performance metrics that can be computed on-the-fly by a computer program. The debate regarding SMT evaluation is focused upon the correlation between automatic evaluation (and performance metrics) and human evaluation.

All the evaluation metrics which are currently used to automatically assess the performance of SMT systems compare one or more reference translations with the hypothesis translation.

One basic assessment of a hypothesis translation can be made by means of *precision* and *recall*. Precision compares the number of correctly translated words with the total number of words in the hypothesis translation, while recall compares the number of correctly translated words with the total number of words in the reference translation. The two evaluation metrics are equally important for SMT systems, because a 100% precision translation can have a poor recall and vice-versa. The standard way to combine the two evaluation metrics is by computing the *F-measure*, defined as the harmonic average of the two. However, there is one more problem with these evaluation metrics: they do not take into account the word order. Consequently we could end up with an incomprehensible translation with 100% precision and 100% recall (the meaning can be completely lost if the right words are wrongly reordered).

Another evaluation metric that was used in the early times to assess SMT systems is the *word error rate*. This evaluation metric was borrowed from speech recognition, but turned out to be too harsh for machine translation. For example, the hypothesis translation: *Cei trei băieți au construit o casă* has the same meaning as the reference: *O casă a fost construită de cei trei băieți*, but it will receive a very low word error rate due to inconsistent word order. This situation can be solved in some degree by the usage of the *position-independent word error rate*.

The standard automatic evaluation metric for statistical machine translation systems is the BLEU score [Papineni, 2002]. BLEU has an elegant solution to the role of word order: it works similarly to position-independent word error rate, but considers matches of larger n-grams with the reference translation. Base on the n-gram matches it computes the n-gram precision (the ratio of correct n-grams in relation to the total number of possible n-grams) for n-gram orders 1, 2, 3 and 4. BLEU is defined as:

$$BLEU-4 = \min\left(1, \frac{hypothesislength}{referencelength}\right)\prod_{i=1}^{4}\frac{correct\,i-grams}{possiblereference\,i-grams} \qquad (3.7)$$

The problem of precision-based metrics (no penalty for dropping words) is addressed by the first factor in this equation. This factor penalizes hypotheses that are too short. BLEU is usually computed on a larger text, because a single sentence may not have any 3-gram or 4-gram matches and, with zero matches for one of the n-grams, the BLEU score drops to zero.

The BLEU score has many critique points and intensive efforts are currently made to create variations and extensions that would eventually turn out to be more correlated with subjective evaluation. Nevertheless, state-of-the-art SMT systems are still evaluated and compared on large-scale using the BLEU score.

3.7 SMT Tools

The most popular translation toolkit used today is the Moses Toolkit [Koehn, 2007]. Moses is a statistical machine translation system that enables the development of translation models for any language pair. Besides the phrase-based translation models, which were described and used in this thesis, Moses can also be used to create tree-based models or factored translation models (models which enable the integration of linguistic information at the word level).

The Moses Toolkit makes use of GIZA++ [Och, 2003] for creating the word alignments. Giza++ is a statistical machine translation toolkit that can be used to train IBM Models 1-5 and an HMM word alignment model.

CHAPTER 4

THE ACQUISITION AND ANALYSIS OF SPEECH, PHONETIC AND LANGUAGE RESOURCES

One of the main contributions of this thesis regards the acquisition and processing of the main resources needed in creating a speaker-independent, large-vocabulary continuous speech recognition system (LV-CSR). Although it is one of the European Union languages, Romanian is still considered a low-resourced language from the point of view of speech and natural language processing resources. For example, the Linguistic Data Consortium (LDC) distributes speech resources for languages such as Czech, Brazilian Portuguese, Vietnamese, Tamil, Egyptian Arabic, etc., but does not provide any resources for the Romanian language. The situation is similar in the case of ELRA (European Language Resources Association), which also distributes language resources and speech resources. ELRA provides some basic linguistic resources for Romanian, but does not have any speech resources for this language. Moreover, recent work on Romanian speech recognition [Munteanu 2006; Dumitru, 2008; Petrea, 2010; Kabir, 2011] complain about the absence of a Romanian standard speech database and report the usage of self-created resources. Of course, this is not the best solution, because, without standard evaluation resources, the efforts of different Romanian speech-research groups cannot be directly compared. The only research-open Romanian "speech" database is "Sounds of the Romanian Language Corpus" [Teodorescu, 2009]. This corpus contains only basic recordings such as vowels, consonants and other Romanian specific sounds and a few emotionally charged phrases and, consequently, cannot be used for speech recognition. In conclusion, any effort towards the development of an automatic speech recognition system must start with the acquisition of a speech database. This is one of the main contributions of this thesis.

Regarding Romanian text corpora, which are needed for statistical language modeling, the situation is slightly better. LDC does not provide any standard text databases for Romanian, but ELRA distributes a few small Romanian text corpora. Regardless, [Macoveiciuc, 2010] states that prior to their work in 2010, there were no large, accessible, general-language corpora for Romanian. This is probably why recent work on Romanian NLP report the usage of different self-created corpora, obtained out of literature books [Vlad, 2007; Ungurean, 2008; Ciucă, 2010], legal documents [Domokoş, 2009], online newspapers [Bick, 2010] and the web as corpus [Macoveiciuc, 2010]. In [Cristea, 2006], the authors make an extensive review of all the language resources and tools created for Romanian as of 2006. As most of these resources are not generally available outside their research groups, we were also required to collect a large Romanian text corpus in order to create a robust, general language model for Romanian. The acquisition and processing of this corpus is another resource-creation contribution of this thesis.

As a conclusion, for the sake of future research on Romanian, it will probably be a very good idea to create standard speech and text resources and make them freely or commercially available to other research groups as well.

4.1 THE PHONETIC DICTIONARY AND THE GRAPHEMES-TO-PHONEMES TOOL

4.1.1 Romanian phonetics

The total number of phones in the Romanian language is somehow vague due to several reasons, among which the most important is the adoption of foreign words. In [Munteanu, 2006] the author uses 34 phones: 8 vowels, 4 semivowels and 22 consonants. In [Ordean, 2009] the authors consider the expert-knowledge provided by [Iordan, 2005] and use a set of 36 phones. The same number of phones, but with a slightly different approach is employed in [Domokoş, 2011].

In other languages the number of phonemes is quite different and also implementation dependent. For example, [Huang, 2001] uses 41 phones for English ASR, while [Mareuil, 1999] uses 44. In [Mareuil, 1999] an automatic multi-lingual phoneme classification is described and the number of phonemes for 6 European languages is asserted to be: 34 for French, 44 for English, 46 for German, 24 for Spanish, 49 for Italian and 38 for Portuguese.

In our studies we have employed the set of 34 phones used in [Munteanu, 2006] supplemented with two more vowels which are mostly used in pronouncing foreign words. The set of phones is presented in Table 4.1. Due to technical reasons we have used a different phoneme coding than the standard IPA coding. The table lists the standard IPA symbols along with our in-house symbols and also gives some words examples.

Table 4.1 Romanian phoneme set

Phoneme			**Words examples**	
Type	**IPA symbol**	**Used symbol**	**Written form**	**Phonetic form**
vowels	a	a	sat (village)	s a t
	ə	a1	gură (mouth)	g u r a1
	e	e	mare (sea/large)	m a r e
	i	i	lift (elevator)	l i f t
	ʲ	i1	tari (strong)	t a r i1
	ɨ	i2	între (between)	i2 n t r e
	o	o	loc (place)	l o c
	u	u	şut (shot)	s1 u t
	y	y	ecru (ecru)	e c r y
	ø	o2	bleu (light blue)	b l o2
semi-vowels	e̯	e1	deal (hill)	d e1 a l
	j	i3	fiară (wild animal)	f i3 a r a1
	o̯	o1	oase (bones)	o1 a s e
	w	w	sau (or)	s a w
consonants	c	k2	chem (call)	k2 e m
	b	b	bar (bar)	b a r
	p	p	par (pole)	p a r
	k	k	acum (now)	a k u m
	ʧ	k1	cenuşă (ash)	k1 e n u s1 a1
	g	g	galben (yellow)	g a l b e n
	ʤ	g1	girafă (giraffe)	g1 i r a f a1
	ɟ	g2	unghi (angle)	u n g2
	d	d	dar (gift)	d a r
	t	t	tot (all)	t o t
	f	f	fața (the face)	f a t1 a
	v	v	vapor (ship)	v a p o r
	h	h	harta (the map)	h a r t a
	ʒ	j	ajutor (help)	a j u t o r
	ʃ	s1	coş (basket)	k o s1
	l	l	lac (lake)	l a c
	m	m	măr (apple)	m a1 r
	n	n	nas (nose)	n a s
	s	s	sare (salt)	s a r e
	z	z	zar (dice)	z a r
	r	r	risc (risk)	r i s k
	ʦ	t1	țăran (peasant)	t1 a1 r a n

Table 4.1 shows the set of Romanian we used in our experiments, but it does not give any information regarding the frequency of occurrence for the phonemes, although in designing a speech recognition system this is a key factor. Speech databases need to be acquired and, during the acquisition process, we would like to take into account the relative frequency of occurrence for these phonemes. The reason is obvious: if the speech database is phonetically balanced (according to the usual phonemes occurrence distribution) then the individual acoustic models will be trained adequately to the phonemes they will be required to recognize. Our goal is, of course, to robustly train the acoustic models for the phonemes which occur very often. If, for example, the speech database does not provide sufficient training examples for a phoneme which occurs very often, then the ASR performance will surely decrease more comparing to the case when the speech database does not provide sufficient training examples for a phoneme which rarely occurs.

The occurrence distribution of the phonemes has been computed using three large text corpora we have collected (see Section 4.3.2): europarl, 9am and hotnews. All the words in these corpora have been phonetized using an existing phonetic dictionary and a graphemes-to-phonemes conversion tool (see Section 4.1.2 and Section 4.1.3). In the end, the phonemes frequency of occurrence has been computed. Table 4.2 and Table 4.3 show the most/least frequently used phonemes in the Romanian language.

Table 4.2 The most frequently used Romanian phonemes

Phoneme	Occurence in corpus [%]			
	europarl	9am	hotnews	all corpora
e	11.83%	10.91%	10.85%	10.91%
a	9.39%	9.55%	9.51%	9.52%
i	7.87%	7.86%	7.75%	7.79%
r	7.77%	7.35%	7.15%	7.25%
t	6.55%	6.61%	6.35%	6.46%
s	5.62%	6.24%	6.34%	6.26%
n	6.43%	6.08%	6.31%	6.25%
u	5.41%	5.44%	5.44%	5.44%
l	4.48%	4.69%	4.57%	4.61%
o	4.40%	4.31%	4.43%	4.38%

The three text corpora employed for these statistics contain 29 million phonemes (europarl), 323 million phonemes (9am) and respectively 513 million phonemes (hotnews), and thus we assert that the overall statistics (all corpora) are very close to the real phonemes usage frequency for the Romanian language. In fact the data comes to support our assertion: the most frequently used phonemes are listed in Table 4.2 in their frequency descending order as given by the overall statistics on all corpora, but we can note that the same order is maintained for the individual corpora also (one single exception: in europarl the “s” is less frequent than the “n”). The phonemes occurrence percentages within the three corpora are also very close, except for the “e” and “s” in europarl. The fact that occurrence exceptions are noticed in the europarl corpus is due to the fact that this first corpus is slightly small. The fact that the phonemes occurrence distribution is roughly the same for the other two larger corpora is another argument supporting the assertion that these statistic estimations are very close to the real ones.

Table 4.3 The least frequently used Romanian phonemes

Phoneme	% in corpus			
	europarl	9am	hotnews	all corpora
o2	0.0003%	0.0005%	0.0006%	0.0005%
y	0.0011%	0.0015%	0.0013%	0.0014%
g2	0.0093%	0.0261%	0.0291%	0.0273%
k2	0.1352%	0.2018%	0.2183%	0.2094%
j	0.1838%	0.2012%	0.2254%	0.2150%
h	0.0635%	0.1860%	0.2451%	0.2170%
o1	0.3293%	0.2200%	0.2278%	0.2283%
g1	0.2647%	0.2562%	0.2738%	0.2669%
w	0.3020%	0.5984%	0.6162%	0.5990%
g	0.5745%	0.6042%	0.6274%	0.6170%

The least frequent phonemes distribution is not that similar over the three corpora. This is probably due to the fact that for some of these phonemes the number of occurrences is not sufficient to issue proper statistics. Large differences between the phonemes percentages within the three corpora can be noticed for “g2”, “h” and “w”.

Figure 4.1 displays the entire phonemes distribution in the Romanian language. The figure uses the data within all corpora. Please note the huge difference in occurrence frequency between the most used vowels (“e”, “a” and “i”) and a large number of other phonemes which have an occurrence frequency of less than 1%.

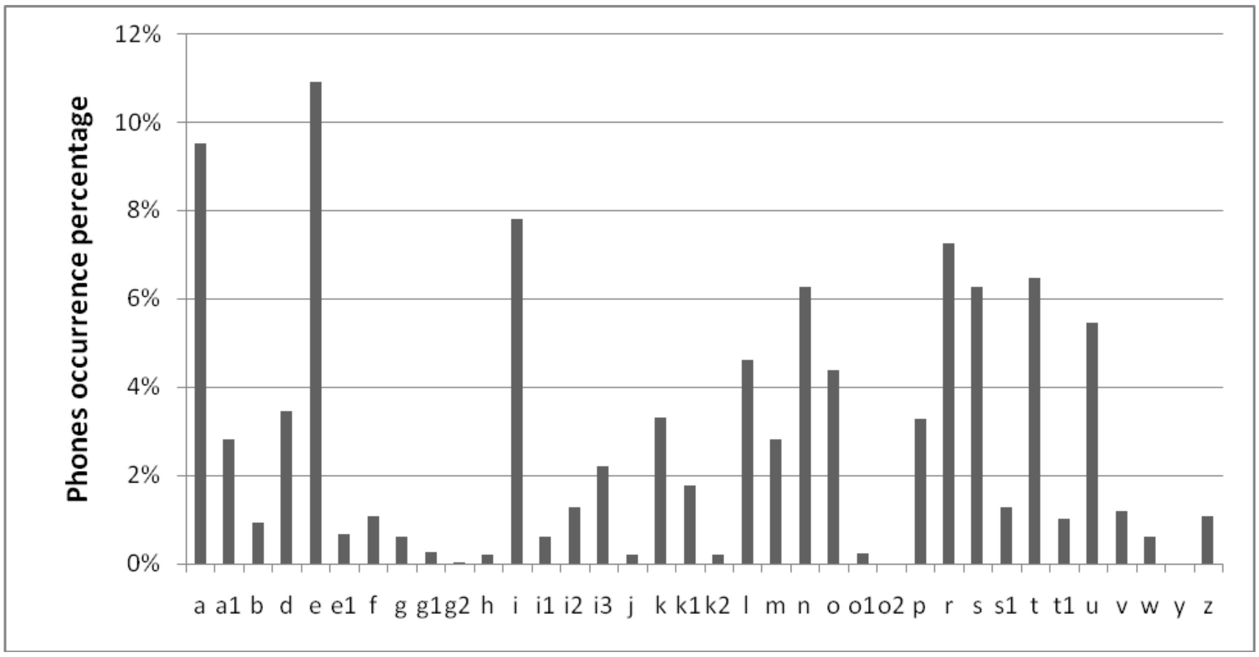


Figure 4.1 The phonemes occurrence distribution in Romanian

Figure 4.2 emphasizes better the differences in occurrence frequency for the various phonemes in Romanian. Note that the percentage of occurrence for the most frequent phoneme (“e”) is similar to the summed percentages for the least frequent 18 phonemes, while the most frequent 6 phonemes account for almost 50% of all phonemes.

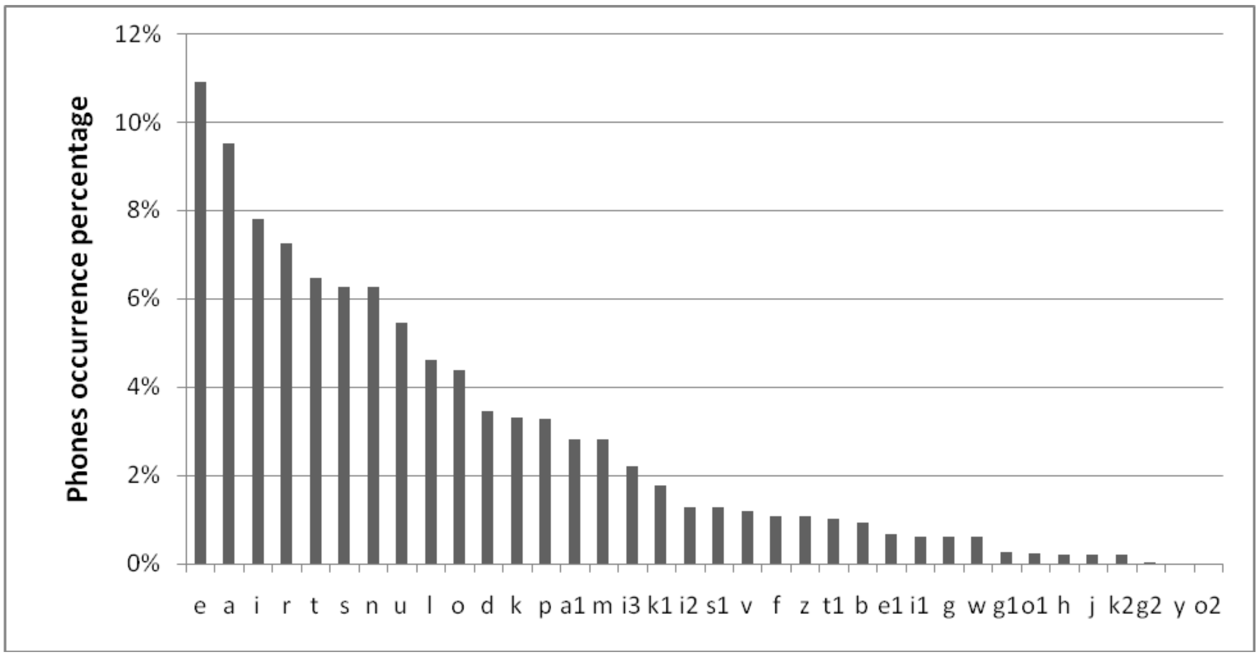


Figure 4.2 The phonemes sorted occurrence distribution in Romanian

The two figures clearly show a highly unbalanced occurrence distribution for the phonemes used in the Romanian language. This is an important piece of information needed in order to successfully develop a *low-cost large-vocabulary* automatic speech recognition system. If the cost (expressed in time and money) is not an issue, then one would create a huge speech database with enough occurrences for all the phonemes in the language, regardless of their occurrence frequency. On the other hand, if the cost is important and only a small speech database can be acquired, then the Romanian phonemes occurrence distribution needs to be taken into account. If a large-vocabulary speech recognition system is the target, then the training database should be designed to exhibit a similar phonemes occurrence distribution. The acoustic models trained on such a database will statistically recognize better Romanian phrases than other models trained on, for example, a speech database with a flat phonemes distribution. In the case of a large-vocabulary ASR is more desirable to better train the acoustic models which are needed more often during recognition (the models for the frequent phonemes) and invest less efforts in training the less frequent phones acoustic models.

4.1.2 The phonetic dictionary

A phonetic dictionary is mandatory for a large-vocabulary speech recognition system. Command and control systems or isolated words recognition systems can be implemented with acoustic models that use words as basic speech units, but for large-vocabulary speech recognition systems modeling sub-words speech units is mandatory. Most commonly used sub-words units are context-independent phones, context-dependent phones (usually triphones) or senones (parts of phones). In either case the decomposition of the word’s written form into a sequence of phonemes is mandatory. This is exactly the role of a phonetic or pronunciation dictionary: to map the word’s written form to its single (or multiple) pronunciation(s) (a sequence of phones). An excerpt of a phonetic dictionary is presented in Table 4.4.

An extensive phonetic dictionary of about 600 thousands word pronunciations was available before this work was started. The dictionary contains many, but not all the words in Romanian. A more important deficit is that it does not contain any proper names. Due to these facts the phonetic dictionary had to be updated several times during the process of speech databases acquisition (several hundreds words within the recorded phrases were not available in the pronunciation dictionary). This first problem was solved by manually creating phonetic transcriptions for all these missing word forms.

Table 4.4 Phonetic dictionary excerpt

…	
auxiliari	a u k s i l i a r i1
auz	a u z
auzea	a u z e1 a
auzeam	a u z e1 a m
auzeați	a u z e1 a t1 i1
auzeau	a u z e1 a w
auzi	a u z i
auzi(2)	a u z i1
auzim	a u z i m
auzind	a u z i n d
…	

When the large-vocabulary desiderate was approached more seriously, a critical problem occurred: several thousands words within the text corpora used for language modeling had no pronunciation in the phonetic dictionary. In order to take full advantage of the language model, an ASR system has to benefit from a full phonetic dictionary (pronunciations for all the unigrams in the language model). Otherwise, the words which do not have a phonetic transcription will not have any chance of appearing at output, because they are not part of the speech decoding search graph. Most of these missing words were, of course, proper names (country names, city names, people names, etc.).

This second issue could not be addressed as the first one, by manually creating phonetic transcriptions for all these missing word forms. The amount of work would have been enormous and would also require phonetic and linguistic knowledge. Moreover, this problem is expected to appear for every new speech recognition task (which generally comes with a new vocabulary). Thus, the need for a graphemes-to-phonemes tool which could automatically create phonetic transcriptions for a given vocabulary is obvious.

4.1.3 The graphemes-to-phonemes tool

Our need for a graphemes-to-phonemes tool is not singular. The task of automatically creating phonetic transcriptions for the words in a vocabulary is very important in speech recognition, but also in speech synthesis and it has been approached by several researchers.

4.1.3.1 Related work

Several approaches to the problem of automatic grapheme-to-phoneme conversion were proposed in the literature. Among these, the most popular are rule-based approaches, machine-learning-based systems and statistical systems.

The rule-based approach considers designing and applying a set of linguistic graphemes-to-phonemes conversion rules. Although these systems are most of the time very efficient, their construction requires strong knowledge of linguistics. Moreover, for some languages the number of rules and exceptions can be huge: 1500 rules for English [Bisani, 2008], over 600 rules for French [Bisani, 2008], etc. On the other hand, for languages such as Spanish [Bonaventura, 1998], Italian, Romanian [Toma, 2009], for which the pronunciation system is quite regular, the number of rules is lower and thus the system is simpler.

The systems that use machine learning are based on the idea that having a smaller set of examples of phonetic transcriptions, we can build a system that will incorporate knowledge from these examples (called training set) and, based on the generalization of the rules, will be able to predict the transcription of words which are not found in the training set [Bisani, 2008]. In practice, these systems are trained using hand built transcription dictionaries covering the most common words for that language. The most widely used systems are based on decision trees or neural networks.

A more novel approach of converting graphemes to phonemes uses Statistical Machine Translation (SMT) principles [Laurent, 2009; Karanasou, 2010]. The graphemes are regarded as words in the source language and the phonemes as words in the target language. A machine translation system is trained based on an initial phonetic dictionary and afterwards this system can be used to convert any other word to its phonetic form.

The most trivial graphemes-to-phonemes conversion approach, which seems to work quite well for some languages, consists in simply modeling graphemes instead of phonemes [Billa, 2002], [Bisani, 2003]. The "phonetic translation" of the word is, in fact, its written form. The graphemes are used instead of real phonemes. These systems have decent results only for languages with low grapheme-to-phoneme ambiguities.

Over the past decade, several research groups have created graphemes-to-phonemes tools for the Romanian language. These tools are regarded as indispensable modules within text-to-speech systems [Burileanu, 1999; Jitcă, 2003; Ordean, 2009; Ungurean, 2011] or for the generation of phonetic dictionaries [Toma, 2009; Domokoş, 2011]. The main methodologies utilized are still the ones used for other languages: machine learning [Burileanu, 1999; Domokoş, 2011], rule-based [Toma, 2009; Ungurean, 2011] and hybrid (machine learning and conversion rules) [Jitcă, 2003; Ordean, 2009]. All these papers report evaluation results in terms of word error rate (WER) or phone error rate (PhER). Generally, there is a single phone error per word [Burileanu, 1999], and thus the PhER is smaller than the WER (as the total number of phones is larger than the total number of words). Even if the results reported in the above papers are not directly comparable due to the different experimental setups (different set of phonemes, different evaluation words, different number of evaluation words, etc.) and the lack of complete evaluation metrics (PhER and WER), we have summarized them in Table 4.5.

Table 4.5 Graphemes-to-phonemes tools for the Romanian language

System	Evaluation words	PhER	WER
[Burileanu, 1999]	1000	n/a	2.9%
[Jitcă, 2003]	400	n/a	~ 5%
[Ordean, 2009]	1000	n/a	5.2%
[Toma, 2009]	4779	0.72%	4.79%
	15599	n/a	9.46%
[Domokoş, 2011]	100	7.17%	n/a
[Ungurean, 2011]	11819	n/a	3.01%

4.1.3.2 Graphemes-to-phonemes method description

In our work, an SMT-based approach, similar to the ones presented in [Laurent, 2009; Karanasou, 2010], has been adopted for the task of automatically creating phonetic transcriptions. This type of approach has not been used before for the Romanian language. An SMT system generally translates text in a source language into text in a target language. Two components are required for training: a) a parallel corpus consisting of sentences in the source language and their corresponding sentences in the target language, and b) a language model for the target language.

For our specific task (graphemes-to-phonemes), 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. Table 4.6 lists a few examples of parallel "sentences" within the training corpus.

Table 4.6 Examples within the phonetic dictionary (parallel corpus)

Example	Source language (graphemes)	Target language (phonemes)
1	d e z n o d ă m â n t u l	d e z n o d a1 m i2 n t u l
2	a c h i t â n d	a c i t i2 n d
3	t a p i ț e r i e	t a p i t1 e r i e

The implementation of the SMT system is based on the Moses Translation Toolkit [Koehn, 2007]. Moses is a widely known toolkit which is mostly used for SMT tasks, but can also solve generic transduction problems as the one presented above. The training of a graphemes-to-phonemes translation model is similar to the one of a general translation model, as described in Section 3.3.

4.1.3.3 Experimental setup and results

The already available phonetic dictionary described in Section 4.1.2 is exactly the parallel corpus needed for SMT training. It was randomly split into three parts: a) a training part (580k words), b) an optimization (tuning) part (10k words) and c) an evaluation part (10k words). The same phonetic dictionary, specifically the phonetic representations, served as training corpus for creating the language model for the target language.

The translation model's optimization should have been made by minimizing the phone error rate (PhER), but this type of optimization module was not available. Therefore, for this process, we chose to use both the two available tuning methods: a) maximization of the BLEU score [Papineni, 2002] (the default in Moses) and b) minimization of the position independent phone error rate (PIPhER) [Bertoldi, 2009]. The evaluation of the translation results has been made using the sclite tool in the NIST Scoring Toolkit. Table 4.7 lists these results, in terms of BLEU score, phone error rate (PhER) and word error rate (WER). Please note that BLEU score is the default evaluation metric for machine translation systems, but is not suitable for our specific task (graphemes-to-phonemes).

Table 4.7 SMT-based graphemes-to-phonemes conversion results

Exp	Optimisation	BLEU	PhER	WER
1	none	98.89	0.53%	4.79%
2	BLEU	99.49	0.33%	3.24%
3	PIPhER	99.39	0.31%	2.76%

4.1.3.4 Conclusion

The graphemes-to-phonemes tool created using the SMT-based approach shows better results (Table 4.7) than all the other systems previously developed for Romanian (Table 4.5). The large, 10k words, evaluation database assures us that the results are conclusive.

The graphemes-to-phonemes tool solves the problem of updating the phonetic dictionary for a new ASR task. The new speech recognition task generally comes with a specific language model and a specific vocabulary. The words in the specific vocabulary need to be phonetically transcribed before the actual recognition process can be started. This process is performed in two steps:

a) all the words within the specific vocabulary which are found in the 600k words phonetic dictionary are transcribed using the 600k words phonetic dictionary,
b) all the other words are transcribed using the graphemes-to-phonemes tool.

Using the above methodology, the conversion error rate will probably be much lower than the one reported in Table 4.7 because most of the words in the specific vocabulary will be found in the 600k words phonetic dictionary, while the graphemes-to-phonemes tool will only be used for proper names and uncommon words.

4.2 Speech Databases and Acquisition Tools

Section 2.4 has explained the acoustic modeling process and has argued that large amounts of data are required to train the various parameters of an HMM-GMM speech recognition system. The Baum-Welch training paradigm needs speech audio clips along with their textual transcriptions in order to estimate the models parameters. Consequently, speech databases are critical resources and their characteristics (number of hours of speech, number of speakers, noise level in audio clips, type of speech, etc.) are very important to the development of a speech recognition system.

CMU Sphinx acoustic training tutorial gives some empirical database numbers for creating good speech recognition systems (Table 4.8). The command and control task is a typical small-vocabulary pseudo-continuous speech recognition task, while the dictation task is a typical large-vocabulary continuous speech recognition task. Note that the speaker-independency desiderate seems very difficult to achieve as resources from approximately 200 speakers are required. This number could seem exaggerated, but inter-speaker speech variability is indeed an important factor and can be overcome only by thoroughly modeling the various possible pronunciations of every phone. This can, in turn, be achieved by recording speech from many different speakers.

Table 4.8 CMU Sphinx suggested database sizes

ASR Task	Speaker dependent system	Speaker independent system
command and control (SV-CSR)	1 hour of recordings, 1 speaker	5 hours of recordings, 200 speakers
dictation (LV-CSR)	10 hours of recordings, 1 speaker	50 hours of recordings, 200 speakers

4.2.1 Speech databases review

As previously remarked, Romanian has very few speech resources, all created by research groups and neither freely, nor commercially available. The authors of [Munteanu, 2006; Dumitru, 2008; Petrea, 2010; Kabir, 2011] explicitly assert that there are no speech resources available for Romanian and that they were required to create speech databases before starting any research in speech recognition. The speech resources created and used by the various Romanian speech recognition research groups are listed in Table 4.9.

Neither of the speech databases described in the previous table is available to other research groups. The size of the databases (in hours of speech) is not mentioned for these speech databases, but, if we estimate an average of 10 seconds per phrase, the largest databases are still smaller than 11 hours of speech. Consequently, given the CMU Sphinx suggestions, none of these databases, even if they were available, would not be large enough for a speaker-independent large-vocabulary speech recognition task. In conclusion, larger speech databases have to be acquired; this process is the next section’s subject.

Table 4.9 Romanian speech databases summary

Database name	Usage/Acquisition reported by	Domain	# phrases	# words	# unique words	# speakers
OCDRL	[Oancea, 2004], [Dumitru, 2008]	telephone dial	850	n/a	3000	8M / 2F
CDRL	[Gavăt, 2007], [Dumitru, 2008]	information, sports, geography, history	3500	n/a	4000	7M / 4F
-	[Munteanu, 2008]	n/a	4000	n/a	n/a	11
METEO	[Militaru, 2009]	forecast news	760	13500	535	15M / 15F
RO-GRID	[Kabir, 2011], [Giurgiu, 2011]	six-words commands	8400	50400	36	12M / 9F

4.2.2 *Speech databases acquisition*

A complete speech database consists of the following components:

- a set of speech signal samples;
- a set of transcription files marking the text which is spoken in each speech sample; these files may include information regarding the temporal boundaries between the speech units;
- additional information regarding speech type (isolated words, continuous, spontaneous), speaker identity, etc;

A database can be collected via several different methods [Petrea, 2008]:

- direct recording; this yields a series of particular issues: choosing the recording place and recording workstation, choosing the microphone, etc.;
- labeling audio books or other spoken materials; the particular issues in this situation are: leveling the differences in sampling rate for the different spoken materials, splitting the audio and labeled content into smaller parts, detecting and correcting labeling errors.

We have started to build speech databases using the second method listed above. The first output was a continuous speech database (CS_BOOKS) obtained by labeling the audio content of several audio books. Initially, the long audio clips were split into smaller files (approximately 60 seconds). Secondly, transcription files, containing the group of words uttered in the audio files, were created. This resulted in a database of approximately 11 hours of speech. It comprises read, continuous speech uttered by 7 different speakers (4 males and 3 females). The domain is Romanian literature.

The second database (PHONES) was developed strictly for the initialization of the acoustic models. It was created by time-stamping and labeling the phones in the "Sounds of the Romanian Language Corpus" [Teodorescu, 2009]. Smaller, single-phone audio clips were created and labeled. This database contains isolated utterances (20 to 500) of the Romanian phones, spoken by 10 speakers (7 males and 3 females).

During the acquisition of these two databases, several acquisition-method issues have gained our attention. Firstly, the audio files and the corresponding transcription have to be split into smaller files (5 seconds to 25 seconds, as CMU Sphinx suggests). This is typically a time-consuming process. Secondly and more importantly, the labeling process is very sensitive to human errors. Typos often occur, creating unknown words or substituting different known words or, even worse, substituting diacritical characters with non-diacritical characters. Moreover, the labeling process is even more time-consuming in case there is no prior draft transcription of the audio file.

These issues triggered us to try a different database acquisition approach: to directly record some prior selected texts. For this purpose we have designed and implemented a speech recording Java application (see Section 4.2.4). The application keeps track of the audio and transcription file names and displays a simple graphical user interface showing the sentence to be recoded, thus making the recording process easier and faster. We have estimated that the recording time was 5 times smaller comparing to the time it would have taken to record the same databases with the operating system's default recorder.

We have used this second approach to record a large isolated words database (WORDS) and several continuous speech databases (CS_01, CS_02 and CS_03). For the isolated words database we have recorded a list of 10000 different words covering all the syllables in Romanian. 9 speakers (3 males and 6 females) have recorded the whole list of words, 2 speakers (2 males) have recorded only a subset of 7000 words and 6 more speakers (2 males and 4 females) have recorded an even smaller subset of 1000 words. Summarizing, the WORDS database consists of 110000 single-word audio clips recorded by a group of 17 speakers (7 males and 10 females). The total size of the database, expressed in hours of speech, is 42 hours.

Some of the authors of the WORDS database have also recorded continuous speech audio clips. The first and the largest continuous speech database is CS_01. The first step we have performed consisted in selecting a set of 1000 phrases from online newspapers, journal interviews, etc. The domain is quite broad. The set of 1000 phrases contain a total amount of 16300 words among which 5175 are different. These phrases were afterwards recorded, one per audio clip, using the speech recorder application, by 11 speakers (4 males and 7 females). The resulted database (CS_01) consists of 11000 single-phrase audio clips with an average size of 6.4 seconds, summing up to a total size of 20 hours of continuous speech.

The CS_02 continuous speech database has been recorded for evaluating a digital library speech recognition system. A set of possible dialogues between a human user and a computer system were created and 244 phrases (the human user part) were selected for future recording. The set of 244 phrases contain a total amount of 1903 words among which 611 are different. These phrases were recorded, one per audio clip, using the speech recorder application, by 3 speakers (2 males and 1 female). The resulted database (CS_02) consists of 732 single-phrase audio clips with an average size of 3.5 seconds, summing up to about of 45 minutes of continuous speech.

The CS_03 continuous speech database has been recorded for evaluating a tourism speech recognition system. The recorded text was manually translated from a tourism French corpus. This database is used in Section 6.2 only for testing purposes: to evaluate an ASR system which makes use of a language model built using a machine-translated text corpus. For this purpose 300 phrases were manually translated to Romanian and recorded, one phrase per audio clip, by 3 speakers (2 males and 1 female). A set of 300 phrases contain a total amount of 1872 words, among which 358 are different. The resulted database (CS_03) consists of 900 single-phrase audio clips with an average size of 3.1 seconds, summing to about 1 hour of continuous speech.

The last continuous speech database which was acquired is called CS_04. It was created only or speaker-independency evaluation purposes. The evaluation part of the CS_01 database (100 phrases with ids between 500 and 599) were selected and recorded by 8 *other* speakers (4 males and 4 females). We have selected this methodology in order to make a fair comparison between the errors made by the ASR system when it was required to recognize speech uttered by known speakers (the ones which recorded all the 1000 phrases in the CS_01 database) versus speech uttered by unknown speakers (the ones which only recorded the 100 evaluation phrases in the CS_04 database). The set of 100 phrases contain a total amount of 1803 words, among which 867 are different. The resulted database (CS_04) consists of 800 single-phrase audio clips with an average size of 8.5 seconds, summing up to a total size of about 2 hours of continuous speech.

Table 4.10 Created speech databases

Database name	Text				Speech		
	Domain	**Phrases**	**Words**	**Unique words**	**Type of speech**	**Hours of speech**	**Speakers**
BOOKS	literature	n/a	89266	14240	continuous	11	4M / 3F
PHONES	n/a	n/a	n/a	n/a	isolated phones	n/a	7M / 3F
WORDS	n/a	110000	110000	10000	isolated words	42	7M / 10F
CS_01	news, interviews	11000	179300	5175	continuous	20	4M / 7F
CS_02	library dialogue	732	5709	611	continuous	0.75	2M / 1F
CS_03	tourism (booking)	900	5616	358	continuous	1	2M / 1F
CS_04	news, interviews	800	14424	867	continuous	2	4M / 4F

All audio clips in the databases share the same sampling frequency (16 kHz) and the same sample size (16 bits). The most important information regarding the speech resources are summarized in Table 4.10.

In [Burileanu, 2008] a detailed description and analysis of these speech databases is made. The article gives a glimpse of the database development status as of 2010. The PHONES database was used for some preliminary ASR tests in [Burileanu, 2010a]. The PHONES and WORDS databases were used in the work reported in [Petrea, 2010; Buzo, 2011a; Buzo, 2011b; Cucu, 2011a]. The WORDS, CS_01, CS_02 and CS_03 databases were used in [Cucu, 2011b] and [Cucu, 2011c].

The author of the thesis was one of the main speakers and coordinators for the acquisition of these speech resources and guided several groups of students with the purpose of achieving this goal.

4.2.3 Speech databases phonetic analysis

As argued in Section 4.1.1 the phonetic balance of the speech databases (according to the real phonemes occurrence distribution in Romanian) is extremely important when their size is relatively small. This section aims to present the phonetic statistic analysis for the various speech databases described in the previous section.

The PHONES database was developed strictly for the initialization of the acoustic models. We did not take into account the Romanian phonemes occurrence distribution when we have developed this database and we focused only on getting a minimum number of occurrences for every phone. Table 4.11 shows the number of occurrences acquired for the 36 phones in Romanian. Note that for some phones (the least frequent, as shown in Table 4.3) we have not acquired any data.

The BOOKS continuous speech database has been acquired by labeling several audio books. The total amount of phones in the selected audio clips is quite high: 384656 phones. Figure 4.3 presents the occurrence distribution for these phones in comparison with the real occurrence distribution for Romanian (according to Section 4.1.1).

The two phonemes occurrence distributions presented in Figure 4.3 are relatively similar. There are a few significant differences for the most frequent phones: the vowels “a1”, “e”, “i” and “o” and the consonant “s”. The BOOKS database contains 61% more “a1”, 13% less “e”, 19% less “i”, 31% less “o” and 34% more “s”.

Table 4.11 Phones occurrences in the PHONES database

Phone	Occurrences		Phone	Occurrences		Phone	Occurrences
a	561		i1	0		o2	0
a1	30		i2	26		p	7
b	31		i3	22		r	38
d	32		j	22		s	38
e	52		k	26		s1	31
e1	26		k1	33		t	23
f	26		k2	0		t1	40
g	25		l	38		u	21
g1	38		m	35		v	32
g2	0		n	48		w	31
h	26		o	13		y	0
i	27		o1	22		z	26

The WORDS database was acquired by recording a list of 10k words which cover all the syllables in Romanian. The amount of phones in a set of recorded audio clips (10k audio files) is relatively small: 95805 phones. Figure 4.4 presents the occurrence distribution for these phones in comparison with the real occurrence distribution for Romanian (according to Section 4.1.1).

Figure 4.4 shows that even for an isolated-words database of approximately 100k phones the relative occurrence of the phones is very similar to that of the Romanian language. Significant differences can be noted only for a couple of phones: the vowels “a” and “e” (the WORDS database contains 22% less “a” and 20% less “e”) and the consonants “l” and “s” (the WORDS database contains 42% more “l” and 43% less “s”). These differences are explicable: the database contains 10k different words with one occurrence each, while the relative distribution of the words in Romanian is highly unbalanced (see Section 4.3.5).

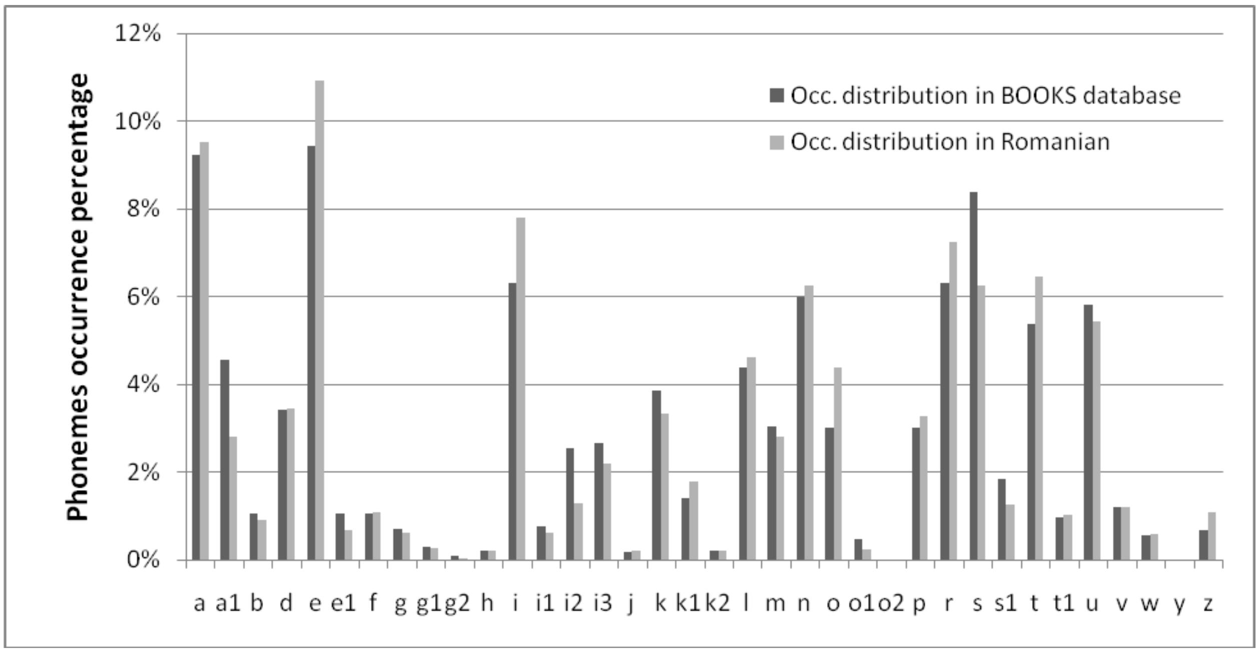


Figure 4.3 The phonemes occurrence distribution in BOOKS database

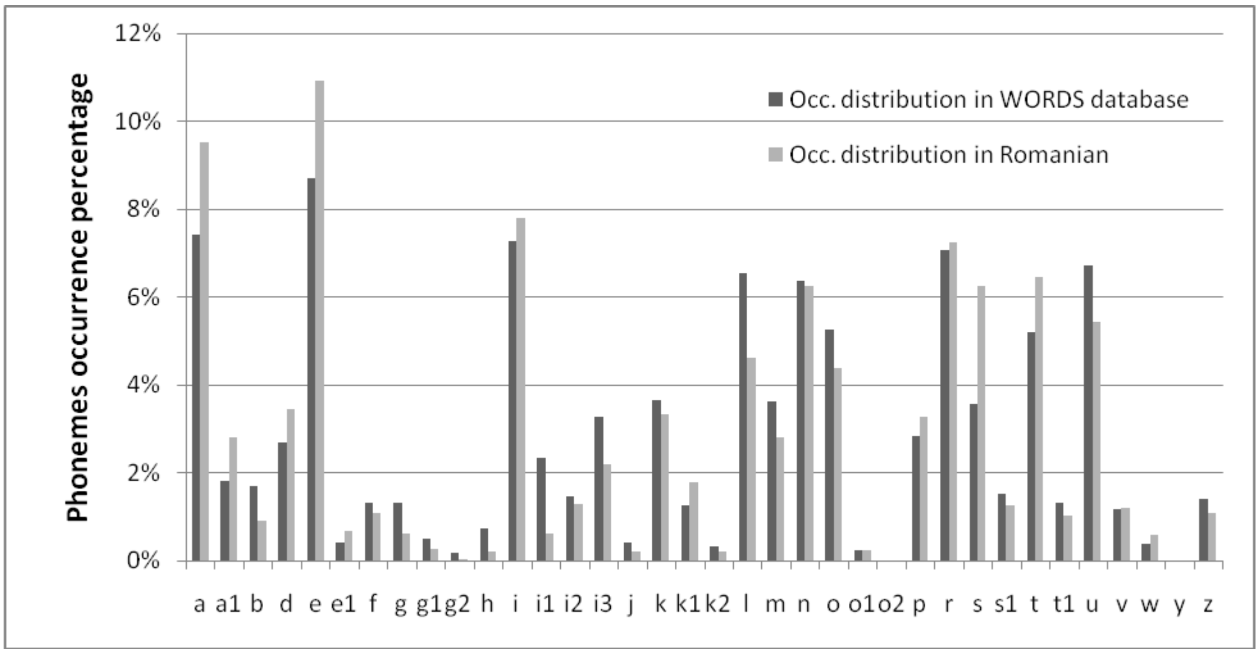


Figure 4.4 The phonemes occurrence distribution in WORDS database

The CS_01 continuous speech database was acquired by recording a set of 1000 phrases selected from Romanian newspapers. The amount of phones in one set of audio clips (1000 audio files) is 13% smaller than the one reported for the WORDS database: 82873 phones. Figure 4.5 presents the occurrence distribution for these phones in comparison with the real occurrence distribution for Romanian (according to Section 4.1.1).

Figure 4.5 shows a very similar occurrence distributions for the CS_01 database and for the Romanian phones. This allows us to assert that the acoustic models that will be trained with this database will be very well adapted to general-domain Romanian speech recognition. The only significant exception is with the phoneme “s”: the CS_01 database contains 33% less “s” than expected for the Romanian language.

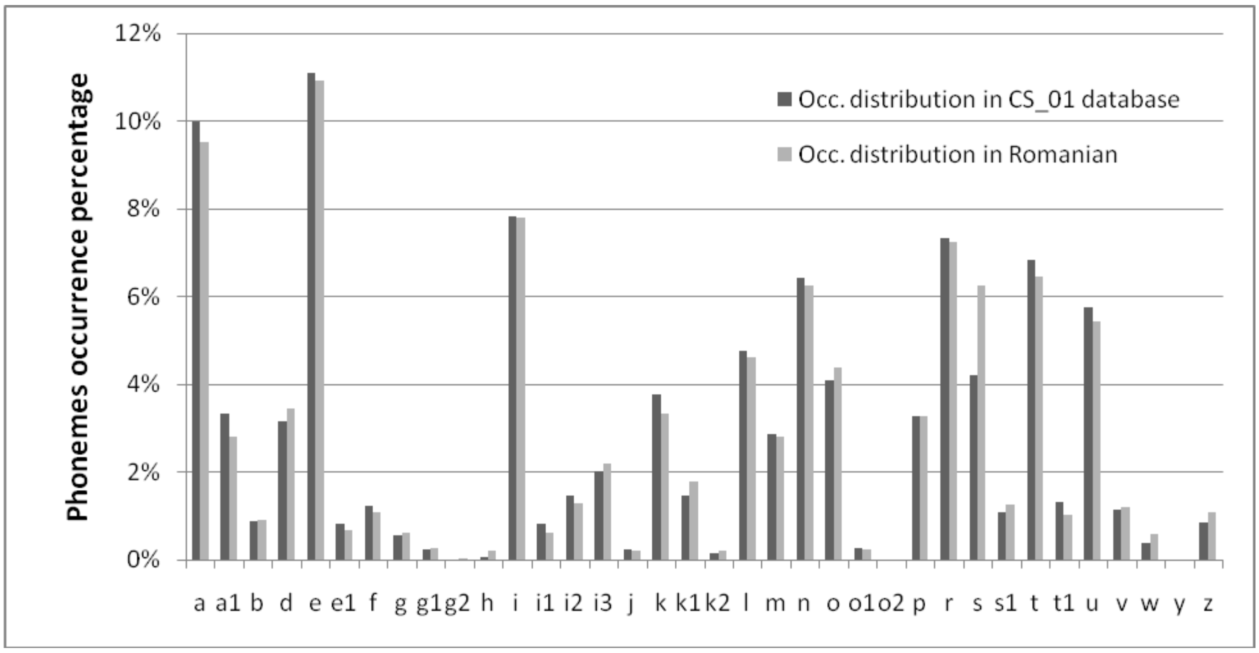


Figure 4.5 The phonemes occurrence distribution in CS_01 database

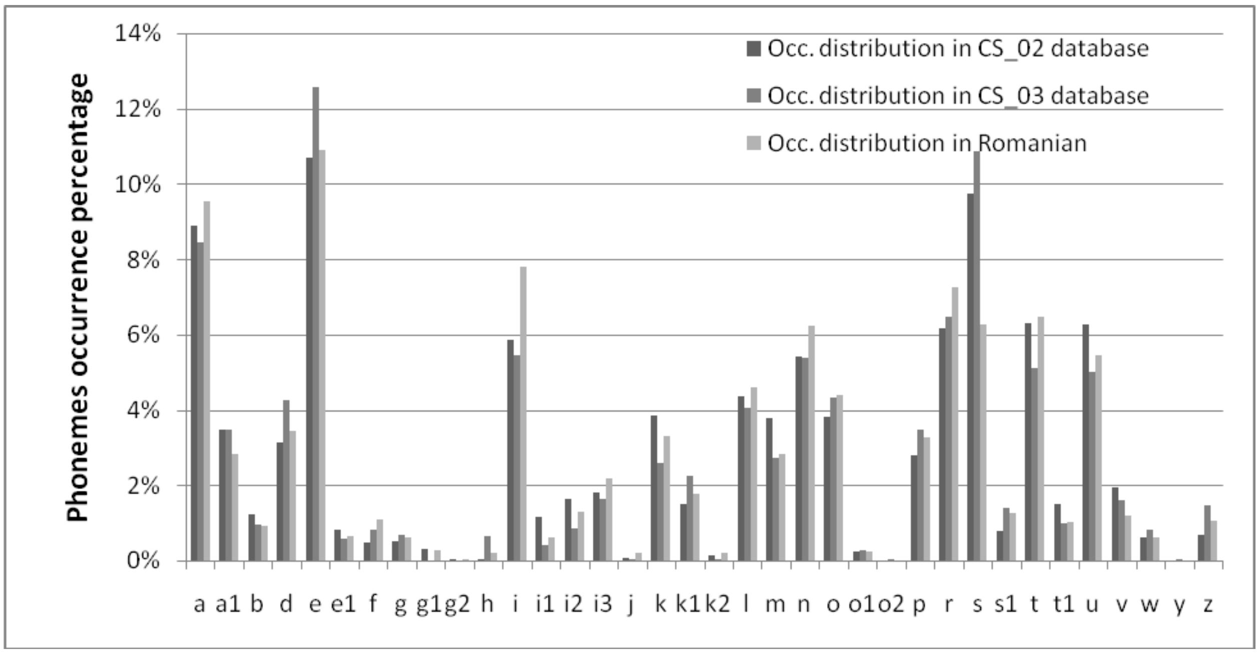


Figure 4.6 The phonemes occurrence distribution in CS_02 and CS_03 databases

The CS_02 and CS_03 continuous speech databases are relatively small compared to the previous analyzed speech databases (9592 phones, respectively 8735 phones). They were created only for testing purposes therefore their phonemes occurrence distribution is not critical. Still, we present these distributions and compare them with the one expected for a general speech signal in Romanian in Figure 4.6. The figure shows that the testing continuous speech databases (CS_02 and CS_03) are also phonetically balanced according to the actual phonemes occurrence distribution in Romanian. Even though the differences are larger than those observed for the CS_01 database, the distribution follows the same main trend line.

The goal of creating phonetically balanced speech databases was not among our objectives when we have started to acquire the databases, due to the lack of data regarding the real phonemes occurrence distribution for Romanian (these statistics were only available after the text corpora were acquired and the graphemes-to-phonemes tool was implemented). Regardless, the phonetic analysis of the speech databases shows that even when small databases such as CS_02 or CS_03 are acquired they tend to be quite well phonetically balanced. The reason for this is that all our continuous speech databases (BOOKS, CS_01, CS_02 and CS_03) contain whole phrases. As the analysis showed, the most unbalanced database is the isolated words database (WORDS) and this is due to the artificial manner in which the words were selected.

In conclusion, if a large-vocabulary continuous speech recognition system is the target, then the most proper training database should comprise free-speech phrases and not isolated words (such as our WORDS database) or artificially constrained phrases (such as in the RO-GRID corpus). Also, in order to have a better control over the phones occurrence distribution, the recommended acquisition method is direct recording of phonetically balanced phrases.

Table 4.12 The speech acquisition tool - input file excerpt

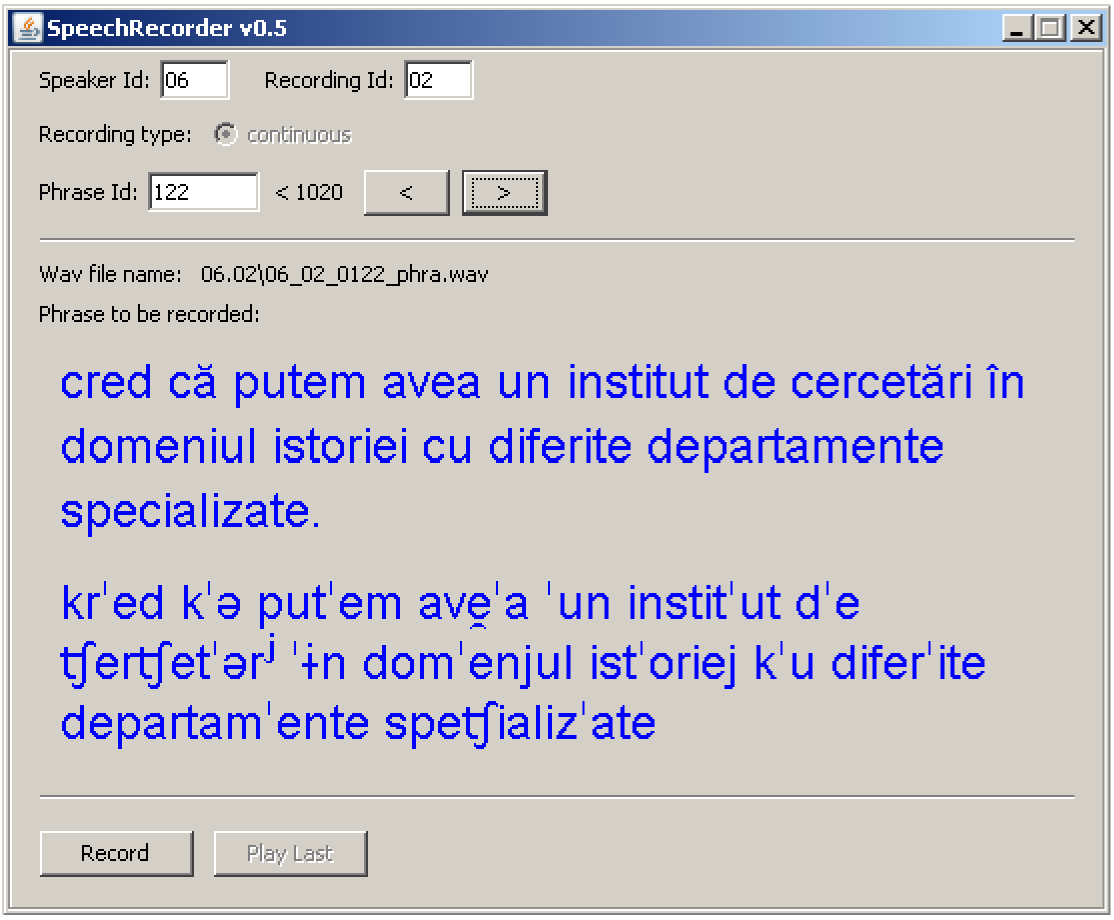
index		written form		phonetic form
…				
0030	#	când pot găsi această carte?	#	kɨnd pot gəsi aʧastə karte
0031	#	voi reveni săptămâna viitoare.	#	voi reveni səptəmɨna viitoa̯re
0032	#	mulțumesc. la revedere.	#	mulʦumesk la revedere
…				

4.2.4 Speech acquisition methodology and tools

The previous two sections argue in favor of direct recording as being the best speech database acquisition method. Reasons such as efficiency and control over the spoken materials, speakers, etc. are invoked as being decisive. Consequently, the need for a personalized recording methodology and recording tool motivated us to implement a Java speech recorder application.

The speech acquisition tool is a graphical user interface (GUI) application which reads in a simple text file comprising the phrases to be recorded, provides the user the means to record the phrases and eventually outputs .wav files. The input file is structured as shown in Table 4.12. It stores one phrase per line and, for each phrase, it lists its id (a 4 digit number), its written form and its phonetic form. The input file is created once for every new database.

The recording methodology is very simple. The user starts the application, selects his speaker id (a two digits number which was previously provided by the database acquisition responsible), selects the database id (a two digits number, also provided by the database acquisition responsible) and the first phrase in the list of phrases is displayed in the GUI. He can now read the phrase and its phonetic form making sure he understands how to pronounce all the words (the phonetic form is displayed using the IPA symbols). Next, the user is able to press the "Record" button, he speaks the current phrase and eventually he presses the "Stop" button to finish the recording. The recorded phrase is automatically saved. The user has the possibility to listen to the previously recorded phrase and, if necessary, re-record it. Once the current phrase is saved (the "Stop" button was pressed by the user), the application displays the next phrase. The graphical user interface of the speech acquisition tool is presented in Figure 4.7.


SpeechRecorder v0.5

Speaker Id: 06 Recording Id: 02

Recording type: (●) continuous

Phrase Id: 122 < 1020

Wav file name: 06.02\06_02_0122_phra.wav

Phrase to be recorded:

cred că putem avea un institut de cercetări în domeniul istoriei cu diferite departamente specializate.

krˈed kˈə putˈem ave̯ˈa ˈun institˈut dˈe tʃertʃetˈərʲ ˈɨn domˈenjul istˈoriej kˈu diferˈite departamˈente spetʃializˈate

Figure 4.7 The Speech acquisition tool – graphical user interface

The users were asked to speak as natural as possible and to repeat the recording if the phrase was wrongly read or if they stutter. The application provides an easy way to listen to the previously recorded phrases and re-record them, if necessary. Moreover the application provides an easy way to browse through all the phrases by writing their indexes in the “Phrase index” field. The same feature enables a user to break up the recording session into as many sub-sessions as needed. The only thing he needs to remember is the last phrase that he recorded so he can continue with the next.

The audio files are saved in the standard .wav format with a very intuitive and information-full name: xx_yy_zzzz.wav. The “xx” part represents the speaker id and the “yy” part stands for the database id, while “zzzz” is a four digits number marking the phrase id. The audio is recorded with a sampling frequency of 16 kHz and a sample size of 16 bits (the usual figures employed in speech recognition).

The recording sessions for the speech databases described in the previous sections have been mostly done in laboratory environment on the same type of workstations, but some speakers also recorded at home on their own desktop or laptop. The microphone used for the recordings was a high-quality Sennheiser microphone.

Each speaker recorded the phrases in his own rhythm and repeated each phrase as many times as it was necessary. The average time it took a user to record the set of 10000 isolated words (one word per audio file) in the WORDS database was 15 hours. The average time needed by a user to record the 1000 phrases in the CS_01 database was approximately 6 hours. Thanks to the application’s ease of use and to the auto-save feature we estimate that the recording tool lowered the speech database acquisition time by 3 to 5 times.

4.3 TEXT CORPORA AND NLP TOOLS

The purpose of any automatic speech recognition (ASR) system is to transcribe a speech signal into a corresponding sequence of words. As described by the general architecture of an ASR system in Section 2.1, one of the indispensable components of such a system is a language model. Section 2.2 describes several types of possible language models and asserts, based on [Koehn, 2010], that n-gram language models represent the state-of-the-art in language modeling. N-gram language models are created using a single resource: text corpora. They are statistical, data-driven models and thus more training data always leads to higher performance. A decent general-purpose n-gram language model needs training corpora of hundreds of millions words. In conclusion, even if not initially obvious, the performance of a general-purpose automatic speech recognition system depends, indirectly, on the amount of general text available for the particular language.

4.3.1 Text corpora review

For Romanian, the target language of this thesis, the available text resources are sporadic and most of the time not publicly available. In [Cristea, 2006], the authors make an extensive review of all the language resources and tools created for Romanian as of 2006. Their review reports not only on plain text corpora, but also on annotated language resources. However, for building n-gram language models, plain text corpora are enough and, regarding this type of resources, the review reports that the largest corpus is RoCo - a news corpus created by AR-ICIA (Research Institute for Artificial Intelligence, Romanian Academy). This corpus (RoCo) is described by its authors in [Tufiş, 2006], where it is said to consist of about 35 million lexical tokens.

In 2010, [Macoveiciuc, 2010] reports on the acquisition of RoWaC (Romanian Web-as-Corpus), a 50-million-word Romanian corpus, and its availability within Sketch Engine [Sketch]. The authors state that prior to their work in 2010, there were no large, publicly-accessible, general-language corpora for Romanian. This is generally true as most of the developed corpora are subsequently used only within the research group which authored them. It's the case for a 21-million-words grammatically-annotated news corpus whose usage is reported in [Bick, 2010], for an 11-million-words corpus which is used for diacritics restoration in [Ungurean, 2008] and for a 9-million-words literary corpus which is subject to a letter-structure statistical analysis in [Ciucă, 2010].

Table 4.13 summarizes the data regarding the Romanian corpora used by different research groups. The second column "Usage/Acquisition reported by" lists articles which mention the usage of these corpora, not necessarily the corpora authors. The first conclusion which emerges from this table is that there are not enough available text corpora to develop a satisfactory general language model for Romanian. Second, the standardization and publication of such corpora, even if not for free, would be very beneficial for the research community.

Table 4.13 Romanian text corpora summary

Corpus name	Usage/Acquisition reported by	Domain	Words	Availability
RoCo	[Tufiş, 2006]	news, literature, law	35M	no
-	[Ungurean, 2008]	journal, literature, other	11M	no
-	[Bick, 2010]	business news	21M	no
-	[Ciucă, 2010]	literature	9.1M	no
RoWaC	[Macoveiciuc, 2010]	news, literature, other	50M	yes (for a price)

4.3.2 Text corpora acquisition

Given the lack of availability of Romanian text corpora, as presented in the previous section, and the need of large corpora to create a language model suitable for LV-CSR, one of the goals of this thesis was to acquire this type of language resources.

The process of corpora acquisition is at this moment dominated by the Web-as-resource or Web-as-Corpus (WaC) approach. For the most common languages there are numerous web pages with large amounts of texts. These can be accessed, processed and used mainly by using the various search engines[1] available. Several methods [Baroni, 2006; Sharoff, 2006] which work well for high-resourced languages were proposed in the last decade. For under-resourced languages, which are less present on the web, special approaches [Draxler, 2007; Scanell, 2007] must be considered and, even so, the size of the acquired corpora is generally much smaller.

Even if for the Romanian language there are not many already-created text corpora, Romanian is very well represented on the web. For a rough estimate consider that a simple Google search for one of the most frequent Romanian words (şi) returns over 500 million entries. Given this, it is obvious that the acquisition of a Romanian corpus should be considering the Web-as-Corpus approach as the first option. In fact, the largest available Romanian corpus [Macoveiciuc, 2010] has been also acquired using this approach.

The process of acquisition started with a freely available parallel corpus: the europarl corpus. This corpus is available online [Europarl] and comprises the discussions in the European Parliament, as recorded in all the European Union's (EU) languages. It was created mainly for the development of Statistical Machine Translation (SMT) systems and this is the reason why the corpora for the different languages are aligned. For our task, we do not need the foreign corpora and will use only the Romanian version of europarl. The English or French europarl corpora are larger as these countries were part of the EU from its birth, while the Romanian corpus consists of 225 thousand phrases summing up to a total of 5.3 million words with correct diacritics. More details about the europarl corpus are given by its authors in [Koehn, 2005].

[1] Google (http://www.google.com), Yahoo (http://www.yahoo.com), Bing (http://www.bing.com)

Secondly, several online newspapers were investigated and analyzed especially from the point of view of the ease of access. By ease of access, we mean the possibility of automatically downloading articles in a simple, batch-mode approach. For example, an online newspaper which publishes its articles using their full names in the URL (Uniform Resource Locator) is more difficult to crawl, then a newspaper which uses only numerical article ids in their URLs. We have exploited this advantage for two Romanian online newspapers [9am] and [Hotnews]. The application created to download the articles ran automatically through all possible article URLs (with all possible articles ids) and, consequently, we managed to create two huge corpora comprising of all the news which were published by these newspapers between November 2004 and March 2011 (9am) and between November 2007 and March 2011 (hotnews). The corpora collected using this method are larger than any other text corpora available for Romanian. The 9am corpus consists of 3.5 million phrases, summing up to a total of 63 million words and the hotnews corpus consists of 6 million phrases, summing up to a total of 100 million words. The credits for this idea go to Miruna Camara [Camara, 2007], who created a much smaller 9am corpus with all the articles available on the website in 2007.

One of the most important problems, that is mandatory to be solved for the news corpora, is the absence of diacritics. This is a general issue with almost all the online newspaper articles and also with other WaC resources such as blog pages, personal pages, etc. The diacritics restoration will be regarded as a preprocessing operation and will be detailed in Section 4.3.4.

Besides the corpora that were acquired as mentioned above, for the various studies and experiments performed for this thesis, we were also given access to a corpus which was created by our research group and was successfully utilized in [Ungurean, 2008] for diacritics restoration. This corpus was split, just as in [Ungurean, 2008], into a training corpus and an evaluation corpus and these parts are denoted, for future reference, misc1 (miscellaneous 1) and misc2 (miscellaneous 2). These two corpora consist of a total of about 11 million words with correct diacritics.

Table 4.14 summarizes the data regarding the 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 following section).

Table 4.14 Created/Acquired Romanian text corpora

Corpus name	Domain	Phrases	Words	Unique words
europarl	EU discussions	225k	5.3M	57k
9am	news	3.5M	63M	397k
hotnews	news	6.0M	100M	503k
misc1	journal, literature, other	300k	9.8M	179k
misc2	journal, literature, other	100k	1.2M	48k

4.3.3 Text corpora processing

N-gram language models are used in wide range of fields and, within these fields, by potentially different applications. For example, in statistical machine translation the language model has the role of estimating the *next token* probability, similarly to automatic speech recognition, where it has the role of estimating the *next word* probability. In [Ungurean, 2008], due to the lack of sufficient training data, a system based on word-suffixes n-grams is created and exploited to estimate the *next suffix* probability. In conclusion, it is very important to decide which are the actual tokens (the grams) we need to model and, based on this, to “clean” the text corpora accordingly.

According to [Huang, 2001; Jurafsky 2009; Renalds, 2010] an automatic speech recognition system is required to output plain text. It's out of its duties to capitalize proper names or to place punctuation marks or other special characters (round/square brackets, percent, etc). Also, numbers will be output in their full, written form, accordingly to what was uttered within the input speech signal. Keeping this in mind, the decision regarding which are the tokens whose occurrence probability we want to estimate is straightforward: the tokens are limited to the language's words, as they are being spoken. In order to create an n-gram language model with these tokens only, we need to adequately process the training corpora.

Before going into cleaning a corpus we first need to obtain the actual data in plain text. This operation is closely linked to the corpora acquisition process and, consequently, is included in the corpora download tool. This tool is a Java command line application [Java], which also makes use of the lynx Unix utility [Lynx]. Its main role is to convert the downloaded html files to plain text files. Normally, the WaC approach of data acquisition produces a set of html files which have to be parsed in order to retrieve and save only the data of concern. For example, imagine the html file of an online article. Besides the actual news text, which we are interested in, the file will contain all sorts of html tags, URLs, the website's menu items, many text advertisements, etc. All these parts have to be removed, because we are interested only in the actual news text.

Another NLP tool created for the purpose of conditioning text corpora is the text cleaning application. This application is also written in the Java programming language and, consequently, can run on any operating system which has a Java Virtual Machine (JVM) installed. The cleaning application takes a corpus as an input and consequently applies several user-specified processing operations with the final goal of obtaining plain text, without any digits, punctuation marks or special characters.

The first cleaning operation has the role of replacing all diacritics characters with a unique character per diacritic. In different corpora and even in different parts of the same corpus, different Unicode characters are used to express the same meaning ("ş" cedilla vs. "ș" comma, "ţ" cedilla vs. "ț" comma, etc.). This inconsistency is approached by the first cleaning operation.

In many news articles and especially in the European Parliament corpus numerous abbreviations are used. For example, *dnă*, *dnă.* and *d-nă* all stand for *doamnă* (*misses,* in English), *dl* and *dl.* stand for *domnul* (*mister,* in English), *art.* stands for *articolul* (*article,* in English), etc. Even if for other NLP tasks, such as machine translation for example, these words are categorized as "normal words", they are not useful at all for an ASR language model. The reason is that when these words are spoken, the speaker utters the *unabbreviated* form. In conclusion, we cannot use a language model that predicts abbreviations and expect to accurately recognize unabbreviated word forms. Moreover, more abbreviation forms for the same word in the training data leads to inconsistent probability estimation for that particular word. The second cleaning operation replaces abbreviated word forms with full word forms based on a list of abbreviations. At this point only a list of about 30 entries has been created and used, but the cleaning operation will get more effective as the abbreviations list will be enlarged.

Numbers written with digits instead of letters represent a similar problem, just as abbreviations. Think, for example, of the spoken representation of the number *1984*. The speech signal has nothing in common with the actual digits *1*, *9*, *8* and *4*, but is, in fact, the spoken representation of the word sequence *o mie nouă sute opt zeci şi patru* (*one thousand nine hundred eighty four*, in English). The ASR language model has the role of predicting the different words which compose this number and not the digits-written form. Moreover, the range of numbers is potentially infinite, but the range of words used to compose the numbers is limited. These words occurrence probability can be estimated with sufficient data, while for the digits-written form we would never have enough data. Besides simple numbers as the one previously exemplified, the corpora are filled with all sorts of other numbers with different meanings and formats: *01.02.2004* (dates), *al 30-lea* (ordinal numeral: *the 30th*, in English), *2.59%* (fractional numbers), *3 000 000, 3.000.000, 3,000,000* (large numbers with various formats), 10:20, 10h20, 10.20 (time stamps with different formats), etc. Only the most frequent number formats have been converted to text, due to the high number of various formats. These issues were approached by the third corpus processing operation.

The fourth processing operation deals with 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 example, a) commas are removed, b) dots, question marks and exclamation marks are replaced with a new line character (this way we'll have one sentence per line in the output file), c) brackets are deleted and the text within them is placed on a new line, d) hyphens, except for the ones within words (such as *într-o*, *dă-mi*, etc.) are removed, e) percent characters are replaced with the phrase *la sută* (*percent*, in English), etc.

The fifth cleaning operation removes all lines (one line represents, in fact, one sentence) with less than tree words. These very short sentences are most of the time harmful, because they are abbreviations or numbers that the previous operations were unable to handle.

In particular, for the europarl corpus, we had to employ a special sixth cleaning operation. The corpus is, for some unknown reason, not entirely in Romanian. Probably due to an acquisition error, some computer-generated sentences, which mark the time when the meetings began, were suspended, resumed, etc., are randomly written in another language (Spanish, German, English, Greek, etc.). Therefore we needed a method to remove these sentences. The sixth processing operation uses the list of words in the extensive phonetic dictionary (described in Section 4.1.2) to remove all the sentences which have an out of vocabulary words (OOV) percentage of more than 30%. The percentage was empirically chosen to balance the removed Romanian sentences and the not-removed foreign sentences.

In the end all letters were lowercased and the empty lines were removed.

4.3.4 A more demanding text processing operation: diacritics restoration

Romanian is a language that makes intensive use of diacritics. Even though it uses only 5 diacritical characters (ă, â, î, ş, ţ), their occurrence frequency is vey high: about 30% to 40% of the words in a general text are written with diacritics. A text that lacks diacritics would generally have these characters substituted by their non-diacritical forms: a, a, i, s, t. Even though for a human reader the meaning of a text without diacritics is most of the times obvious (given the paragraph context), the diacritics restoration task is not trivial for a computer.

The words in Romanian can be grouped into two categories, based on the ambiguity caused by lack of diacritics:

a) non-ambiguous words (words that are either written without any diacritics or written with a fixed diacritics pattern): *alb* (*white*, in English), *astfel (like this*), *pădure* (*forest*), *ştiinţific* (*scientific*),

b) ambiguous words (words that can be written with several diacritics patterns): *casa* / *casă* (*the house* / *a house*), *pana* / *pană* / *până* (*the feather* / *a feather* / *until*).

The word-level context ambiguity is usually not bothering for a human reader if the phrase-level context is known. There are only a few cases in which, given the phrase-level context, a human would not be able to infer the correct meaning of the phrase. Such an example is *Am văzut o fată frumoasă* (*I saw a beautiful girl*), versus *Am văzut o faţă frumoasă* (*I saw a beautiful face*), where the text-level context is required in order to handle the ambiguity *t* / *ţ* in *fată* / *faţă* (*girl* / *face*) [Ungurean, 2008].

Most of the Romanian news corpora which are acquired using the WaC (Web-as-Corpus) approach come without diacritics. For a news article, the diacritics are not very important since any reader has access to the full text-level context and ambiguities appear seldom. On the contrary, the output of an ASR system could be very short and elliptical and the lack of diacritics could make it ambiguous or even incomprehensible. Therefore, an automatic diacritics restoration system is definitely needed. It can be used to restore the diacritics on the output of a diacritics-lacking ASR system or to restore the diacritics on all the corpora before language modeling and thus to create an ASR system which directly outputs texts with diacritics.

4.3.4.1 Related Work

Several, fundamentally different, diacritics restoration methods were developed for the Romanian language. Some of them are knowledge-based, while others use pure-statistical approaches. Some methods are only interested in the character-level context, while others perform better if the full word-level context is given. The amount of training resources is also an important factor, as this generally approximates the cost of developing a diacritics restoration system, given the method.

A *pure-statistical approach* is described in [Mihalcea, 2002]. This method uses a *character n-gram model* and experiments with a *memory-based learning system* (TiMBL) and a *decision tree classifier* (C4.5). A very important plus for the method is its language independency. Only a medium size text corpus is required to train the character n-gram model and no word-level assumptions are made. The method is applied for four different languages: Czech, Polish, Hungarian and Romanian and reports similar results (precision between 97.04% and 99.02%). For Romanian, the reported precision is 98.30%.

A more elaborate, *knowledge-based diacritics restoration method*, using *part-of-speech (POS) tagging* to disambiguate the different diacritical words hypotheses, is introduced in [Tufiş, 1999] and refined in [Tufiş, 2008]. In the development phase, this system requires more NLP resources than the one presented in [Mihalcea, 2002], but also achieves a better performance: a word error rate (WER) of 2.25% and a character error rate (ChER) of 0.60%. The method was integrated into a standalone diacritics restoration system (Diac$^+$), which is available online as a MS-Office add-on [Diac].

In [Ungurean, 2008] the diacritics restoration system is regarded as a *sequential filtering process* based on *unigrams and bigrams of diacritical words* and *trigrams of diacritical word-suffixes*. This method needs only a medium size text corpus to train the various language models and to create a map connecting the non-diacritical word forms to all their diacritical word forms. The authors insist on the fact that this method is adapted to Romanian thanks to the usage of word-suffixes trigrams. The results reported for this method are similar to the ones reported by [Tufiş, 2008]: a word error rate of 2.13% and an overall F-measure of 99.34%. The same research group has recently published updated diacritics restoration results. On a 2 million words part of the europarl corpus, [Ungurean, 2011] reports a word error rate of 1.4% and a character error rate of 0.4%. These are the best diacritics restoration results reported so far for Romanian.

4.3.4.2 Diacritics restoration method description

A statistical language modeling method which has been successfully used for several disambiguation tasks (including true-casing [Liță, 2003]) is proposed in this thesis for diacritics restoration. The training and restoration methodologies are depicted in Figure 4.8.

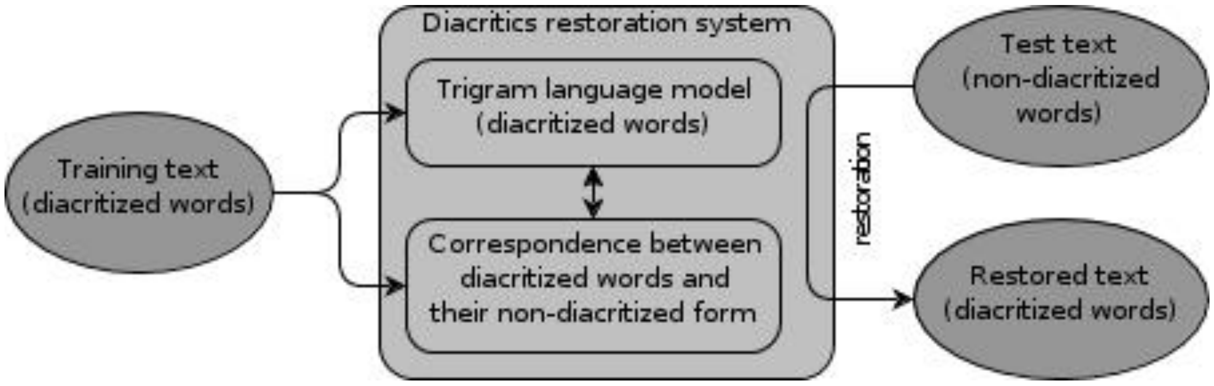

Diacritics restoration system
Trigram language model (diacritized words)
Correspondence between diacritized words and their non-diacritized form
Training text (diacritized words)
Test text (non-diacritized words)
Restored text (diacritized words)
restoration

Figure 4.8 Diacritics restoration system architecture

The only resource needed by this method is a text corpus with correct diacritics. Based on this corpus, two higher-level structures are built: an n-gram language model and a probabilistic map (which links all non-diacritical word forms to all their possible diacritical word forms). An excerpt of a probabilistic map is shown in Table 4.15.

Table 4.15 Probabilistic map excerpt

…
dacia: dacia 1.0
fabricand: fabricând 1.0
pana: pana 0.005, pană 0.008, până 0.987
sarmana: sărmana 0.847, sărmană 0.153
tari: tari 0.047, țari 0.002, țări 0.942, târî 0.008
…

Given a text corpus in which the diacritics are partly or entirely missing we estimate the diacritical form of every word in the corpus in a word-by-word manner. If the non-diacritical word form *ndw* is not found in the probabilistic map we leave it unchanged. Otherwise, we estimate the diacritical form dw^*, given the preceding sequence of *N* diacritical words *dws*, by finding the diacritical word form dw_i that maximizes this formula:

$$dw^* = \arg\max_{dw_i} p(dw_i \mid W) \times p(dw_i \mid ndw) \tag{4.1}$$

The first factor in the equation is estimated by the n-gram language model, while the second one is estimated by the probabilistic map.

4.3.4.3 Experimental setup

For development and evaluation, several corpora were used, just as shown in Table 4.16. Details about these corpora were given in Section 4.3.2. Note that one of the training corpora (misc1) and one of the evaluation corpora (misc2) have also been used in [Ungurean, 2008] and were provided by the authors. The 9am30art corpus comprises the first 30 articles of the 9am corpus for which the diacritics were manually restored (in the reference text).

Table 4.16 Diacritics restoration development and evaluation data

Corpus name	Size [words]	Usage
europarl	5.3M	development
misc1	9.8M	development
9am30art	13k	evaluation
misc2	1.2M	evaluation

The n-gram language models were constructed using the SRI-LM Toolkit [Stolcke, 2002] and the probabilistic map was created using a Java command line application. The disambiguation method which uses these two resources is also provided by the SRI-LM Toolkit (the disambig tool). This tool actually translates a stream of tokens from a vocabulary V_1 (in our case: words without diacritics) to a corresponding stream of tokens from a vocabulary V_2 (in our case: words with diacritics), according to a probabilistic, 1-to-many mapping. Ambiguities in the mapping are resolved by finding the V_2 sequence with the highest posterior probability given the V_1 sequence. This probability is computed from pair wise conditional probabilities $p(V_1|V_2)$, as well as a language model for sequences over V_2.

Given that the Diac$^+$ system is freely available online we were able to evaluate this system on our own data as well. The results are presented in the following section.

For evaluation (words/characters alignment and word/character error rate computation) we have used the sclite tool provided in the NIST Scoring Toolkit.

4.3.4.4 Comparative results

The diacritics restoration systems were evaluated in terms of word error rate (WER), character error rate (ChER) and F-measure. The word error rate is calculated exactly as for an ASR task (see Section 2.5). The character error rate is computed similarly, but, instead of word insertions, deletions and substitutions, the character measures are evaluated. For these evaluation metrics (WER and ChER) the ideal score is 0. The F-measure is defined as the harmonic mean of precision and recall. Precision is the ratio between the number of correctly inserted diacritics and the number of diacritics in the hypothesis text, while recall is the ratio between the number of correctly inserted diacritics and the number of diacritics in the reference text. For these evaluation metrics (precision, recall and F-measure) the ideal score is 100%. Note that precision and recall values are different when the results are given individually per character (Table 4.18), but when the overall results are presented (as in Table 4.17 and last line of Table 4.18) they are equal, and consequently equal to the F-measure value.

In order to find the best setup for the language model we have performed the following experiments (Table 4.17): 2-gram up to 5-gram language models with probabilistic maps (experiments 1 to 4). The conclusion was that the 3-gram language model performs best and it is also simpler than the 4-gram and 5-gram language models, which have similar results. In the end a plain map which assigns equal probabilities to all existing diacritics patterns for a word, was tested, but the results were significantly worse. Our conclusion was that, given the amount of training data, our method works best if it makes use of a 3-gram language model and a probabilistic map.

Table 4.17 Diacritics restoration parameter tuning results

Exp	LM	Prob Map	Evaluation corpus: misc2			Evaluation corpus: 9am30art		
			WER	ChER	F-measure	WER	ChER	F-measure
1	2-gram	probabilistic	2.07%	0.50%	98.69%	1.55%	0.32%	99.15%
2	3-gram	probabilistic	1.99%	0.48%	98.73%	1.50%	0.31%	99.18%
3	4-gram	probabilistic	1.99%	0.48%	98.73%	1.48%	0.31%	99.19%
4	5-gram	probabilistic	2.00%	0.49%	98.71%	1.49%	0.31%	99.18%
5	3-gram	plain	2.24%	0.54%	98.58%	1.51%	0.33%	99.13%

Table 4.18 lists the various performance metrics for the individual characters that are subject to diacritics restoration. The first conclusion that can be drawn based on these results is that the method exhibits better performance metrics for the non-diacritical characters (*a*, *i*, *s*, *t*). Also, there are ambiguity classes (*i* / *î*; *s* / *ş*) which are almost perfectly solved, while others (*a* / *ă* / *â*) pose serious problems. The *a* / *ă* ambiguity is a specific and difficult problem for Romanian, because all the feminine nouns and adjectives whose singular, indefinite form ends in *ă* have their singular, definite forms ending in *a*. Consequently, these word forms cannot be disambiguated easily and we would probably need higher order n-gram models or some linguistic, knowledge-based method to approach this ambiguity.

The individual character results and the above conclusions are consistent with the ones presented in [Mihalcea, 2002] and [Ungurean, 2008].

Table 4.18 Diacritics restoration individual character evaluation

Ambiguity class	Char	Evaluation corpus: misc2			Evaluation corpus: 9am30art		
		Precision	**Recall**	**F-measure**	**Precision**	**Recall**	**F-measure**
a / ă / â	a	98.28%	97.71%	97.99%	99.32%	98.20%	98.75%
	ă	94.42%	96.13%	95.27%	93.73%	97.54%	95.60%
	â	98.79%	97.55%	98.16%	97.00%	99.08%	98.03%
i / î	i	99.97%	99.88%	99.92%	100%	99.96%	99.98%
	î	99.26%	99.65%	99.45%	100%	100%	100%
s / ş	s	99.75%	99.62%	99.69%	99.21%	99.89%	99.55%
	ş	98.71%	99.14%	98.92%	99.66%	97.67%	98.65%
t / ţ	t	99.52%	99.62%	99.57%	99.78%	99.96%	99.87%
	ţ	97.74%	97.21%	97.47%	99.70%	98.51%	99.10%
all	all	98.73%	98.73%	98.73%	99.18%	99.18%	99.18%

Table 4.19 presents comparative results between our method (Exp 1) and the methods proposed in [Tufiş, 2008] (Exp 2, 3) and [Ungurean, 2008] (Exp 4).

The evaluation of the method presented in [Ungurean, 2008] on the misc2 corpus is already available in their article (this corpus was provided by the authors). The performance metrics obtained on the 9am30art corpus were also provided by the authors on our demand.

The evaluation of the Diac$^+$ system was somehow more complicated, because the freely-available MS Office add-on is *not a fully unsupervised system*. The method can work in an unsupervised scenario, as the authors assert in [Tufiş, 2008] ("the S-words were automatically dealt with"), but the MS Office add-on does not; it deals with most of the unambiguous and ambiguous words and prompts the user to select the right word only for the POS ambiguous words. To evaluate this system we have selected POS ambiguous words alternatives in two different ways:

- Exp 2: we have manually selected the first alternative provided by the system,
- Exp 3: we have left the POS ambiguous words in their non-diacritical form (as if the system did not know how to deal with these words).

For the 9am30art corpus we were needed to make 160 manual selections for Exp 2.

Due to the fact that the Diac$^+$ MS Office add-on was not able to restore the diacritics on the large, 1.2 million words evaluation corpus, we were not able to fill all the cells in this table (these cells were marked n/a). For the same reason we have also evaluated the three systems on a smaller corpus, namely 9am30art.

Table 4.19 Diacritics restoration comparative results

Exp	Diacritics restoration methods	Evaluation corpus: misc2			Evaluation corpus: 9am30art		
		WER	ChER	F-measure	WER	ChER	F-measure
1	this method	1.99%	0.48%	98.73%	1.50%	0.31%	99.18%
2	Diac$^+$ [Tufiş, 2008]*	n/a	n/a	n/a	1.61%	0.34%	99.11%
3	Diac$^+$ [Tufiş, 2008]**	n/a	n/a	n/a	2.28%	0.51%	98.67%
4	[Ungurean, 2008]	2.13%	0.25%	99.34%	2.37%	0.50%	98.95%

The comparative results presented in Table 4.19 clearly show that for the larger evaluation corpus the method presented in [Ungurean, 2008] is the best in terms of character-level evaluation. Yet, the method we propose outperforms it in terms of word-level evaluation. This mismatch between the character-level and word-level evaluation metrics leads us to the conclusion that one method manages to correctly restore some diacritics, but fails to restore some other, which are correctly restorer by the second method. Consequently a combined method would probably output even better results.

As regarding the Diac$^+$ system, if the ambiguous words are left non-diacriticized, the results are a lot worse comparing to both the other diacritics restoration systems. If the first alternative, as provided by the system, is selected, then the results are comparable to the ones obtained by our diacritics restoration method.

One important conclusion that can be drawn based on Table 4.19 is that the evaluation corpus is very important: experiments 1 and 4 exhibit significant differences in the diacritics restoration performance on different evaluation corpora. In Exp 1 we observe better performance on the 9am30art corpus, while in Exp 4 we notice better figures for the misc2 corpus. The results obtained on the misc2 corpus are more trustful given the larger size of this corpus (see Table 4.16), but the significantly different results obtained on the 9am30art corpus (which is also satisfactory large) underline the fact that different diacritics restoration systems cannot be directly compared when the experimental setups are different.

4.3.4.5 Conclusion

Romanian texts are filled with diacritical words: about 30% to 40% of the words in a general text are written with diacritics. Most of the corpora acquired via the web lack diacritics, because a human reader can infer the sense of the text even without diacritics. Even though the diacritics restoration task is an NLP task that could seem unrelated to the subject of this thesis, the restoration process is critical for automatic speech recognition systems which are required to output correct Romanian texts. These ASR systems benefit from LMs and their output is directly dependent on the language models which, in turn, must be trained with correct Romanian texts.

This thesis proposes an easy-to-build statistical diacritics restoration system which is constructed using a single resource: a medium-size text corpus with correct diacritics. A brief review of all the other Romanian diacritics restoration systems is also made. The proposed method is evaluated and compared with the most recent systems and the conclusion that we reach is that our method is one of the best diacritics restoration methods available for Romanian.

Moreover, we also conclude that statistical methods (this method and the one presented in [Ungurean, 2008]) are at the moment better than knowledge-based methods (the one presented in [Tufis, 2008]), and could perform even better if more training data would be available. However the statistical methods are limited when it comes to ambiguities that can only be solved using linguistic information (see the *a* / *ă* ambiguity class).

The diacritics restoration system was in the end used to restore the diacritics in the two large news corpora (9am and hotnews) which were acquired via the web. These two corpora were critical resources (see Section 6.1.2) needed to create the large-vocabulary continuous speech recognition system.

4.3.5 Text corpora analysis

The corpora that will be further used for language modeling are europarl, 9am and hotnews. This section will focus on a short analysis of these corpora, discussing the most common words, the words occurrence distribution, the words length distribution, etc. Note that these are general Romanian texts, and the news corpora (9am and hotnews) are the largest reported for this language (see Section 4.3.1 for other corpora sizes). Their size (63M words and respectively 100M words) makes them good candidates for characterizing written Romanian.

Table 4.20 lists the most frequent words in the three corpora and their relative occurrence frequency. The table shows that the most frequent 4 words (*de*, *şi*, *în*, *a*) are the same for the three corpora. The smaller corpus (europarl) exhibits an inflated frequency for the word *în*, but, as regards the other three most frequent words, the occurrence percentage is very similar among the three corpora. The europarl corpus is probably too small to estimate real word occurrence percentages. As listed in Table 4.20, the most frequent 15 words ranking differs quite a lot for this corpus when compared to the other two.

Table 4.20 The most frequent words in the corpora

europarl (5.3M words)			9am (63M words)			hotnews (100M words)		
Rnk	**Word**	**Occ. [%]**	**Rnk**	**Word**	**Occ. [%]**	**Rnk**	**Word**	**Occ. [%]**
1	de	4.66%	1	de	4.93%	1	de	5.01%
2	în	3.46%	2	şi	2.95%	2	şi	3.13%
3	şi	3.35%	3	în	2.55%	3	în	2.59%
4	a	2.26%	4	a	2.33%	4	a	2.27%
5	să	2.13%	5	la	1.88%	5	la	1.93%
6	la	1.42%	6	o	1.19%	6	o	1.07%
7	care	1.39%	7	din	1.06%	7	din	1.04%
8	pentru	1.38%	8	mii	1.01%	8	cu	1.00%
9	că	1.23%	9	cu	1.00%	9	să	0.99%
10	o	1.12%	10	care	0.97%	10	care	0.95%
11	este	1.09%	11	să	0.95%	11	că	0.90%
12	cu	1.04%	12	că	0.93%	12	pe	0.89%
13	din	0.96%	13	pe	0.88%	13	pentru	0.82%
14	nu	0.95%	14	pentru	0.86%	14	mai	0.78%
15	mai	0.78%	15	două	0.81%	15	nu	0.77%

As regarding the 9am and hotnews corpora we observe that the first 15 word ranking is almost the same. We noticed that the words *două* (*two*) and *mii* (*thousands*) appear in the 9am corpus more times than usual due to the fact that every article in this news corpus is dated with dates between 2004 and 2011. These numerical dates are translated to plain text and thus the inflated frequency for the two words. Regardless, we see that, besides these two words, the top 13 most frequent words ranking is practically identical for the two corpora. Moreover, the occurrence percentages are very similar for these 13 words (the most significant difference is noticed for the word *o* which appears 10% more in the 9am corpus). Given this, we can conclude that these words are the most frequently used words in Romanian and that their occurrence percentage is very similar to the one presented in Table 4.21.

Table 4.21 The most frequent words in Romanian (all corpora – 169M words)

Rank	**Word**	**Occ. [%]**
1	de	4.97%
2	şi	3.07%
3	în	2.60%
4	a	2.29%
5	la	1.89%
6	o	1.12%
7	din	1.04%
8	să	1.01%
9	cu	1.00%
10	care	0.97%
11	că	0.92%
12	pe	0.88%
13	pentru	0.85%
14	mii	0.84%
15	mai	0.79%

Table 4.21 shows an interesting fact: the words occurrence percentage decreases very fast. If this trend continues through-out the rest of the top, then it means that a small number of unique words will make up a large percentage of the corpus. Table 4.22 tries to give us a glimpse on how important are the first most frequent words in the three corpora. The table shows the coverage of the most frequent 100 words, 1k words, 10k words and 64k words.

Table 4.22 Most frequent words corpora coverage

	Occurrence in corpus [%]			
Word groups	**europarl**	**9am**	**hotnews**	**all**
most frequent 100	45.8%	46.7%	43.0%	43.6%
most frequent 1000	72.6%	67.7%	64.7%	65.6%
most frequent 10k	95.5%	89.2%	87.6%	88.1%
most frequent 64k	100%	98.3%	97.9%	98.0%

Given the data in the Table 4.22 we can conclude that only a small fraction of the words in a corpus covers a large part of that corpus. The number of unique words for the three corpora was shown before in Table 4.14. This unbalanced word distribution is discussed also in [Koehn, 2005] and a famous formula (the Zipf's law) is emphasized. According to this, the product of the rank *r* of each word (sorted by occurrence percentage) and its number of occurrences (frequency *f*) is roughly a constant. Table 4.23 makes an attempt to verify this formula and displays a comparison to results presented in [Koehn, 2005]. Just as in the experiments done in [Koehn, 2005], we conclude that the Zipf's law is mostly verified. The product of rank *r* and frequency *f* is roughly constant for most words, but drops off for the lowest ranked words.

Table 4.23 Zipf's law verification attempt

All Romanian corpora (169M words)					English europarl corpus (29M words)		
Rank	Word	Frequency	r*f		Rank	Frequency	r*f
1	de	8375953	8375953		1	1929379	1929379
10	care	1630568	16305680		10	424552	4245520
100	minus	170814	17081400		100	30384	3038400
1000	americană	16171	16171000		1000	2793	2793000
10000	biletul	1398	13980000		10000	70	700000
100000	restrictivi	24	2400000		86999	1	86999

Another interesting fact we observe in Table 4.21 is the short length of the most frequent words. Although this observation could seem trivial and not important, the statistics regarding the frequency of short vs. long words is very important for speech recognition. One of the tuning factors of an ASR system is the word insertion penalty – a parameter which controls the word insertion errors by assigning a lower or higher probability penalty to word splitting. If this penalty is larger than required for the language, then the ASR system will output longer, wrong words instead of two or three correct, short words. On the other hand, if the penalty is smaller than required for the language, the ASR system will assign higher probabilities to two-three wrong short words instead of a single, longer correct word.

The word length distribution is different for different languages and, therefore, it needs to be estimated for the target language. The estimation can only be done using large enough text corpora, such as the ones analyzed in this section.

Figure 4.9 displays a word length distribution analysis made on the three corpora. The analyzed word lengths are from 1 to 15. Although there are a few words longer than 15 characters, their number is not significant.

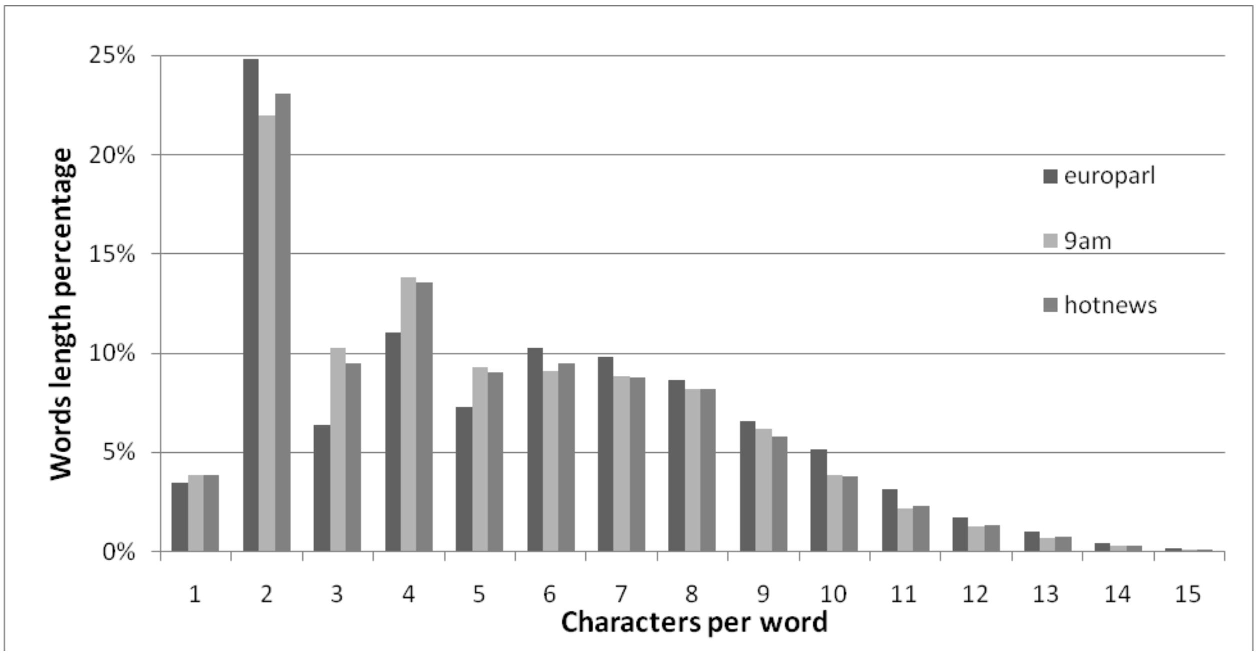


Figure 4.9 Words length distribution for the corpora

Several important conclusions can be drawn from the words length statistics. First, we observe that the three distributions are very similar. The 9am and hotnews distributions are almost identical, while the europarl distribution presents a higher peak for 2-character words and lower values for 3-character and 4-character words. In the end we conclude that even a small corpus, such as europarl (5.3M words), is enough to create a satisfactory estimate of the words length distribution.

The second observation regards the strange shape of the words length distribution. The extremely high number of 2-character words is unexpected. Also unexpected is the small number of 3-character words (the 3-character words are fewer than 2-character words and also fewer than 4-character words). We could assume that this strange shape is caused by specific text corpora or data sparseness, but, as previously observed, the distribution is similar for the three corpora and, to argument even more, tests on other corpora show similar words length distributions.

The average word length calculated based on the data in Figure 4.9 is 5.42 characters per word for europarl, 5.12 characters per word for 9am and 5.11 characters per word for hotnews.

As the words length distributions for the three different corpora are very similar we assume that this is the general case for the Romanian language and show a unified Romanian words length distribution in Figure 4.10.

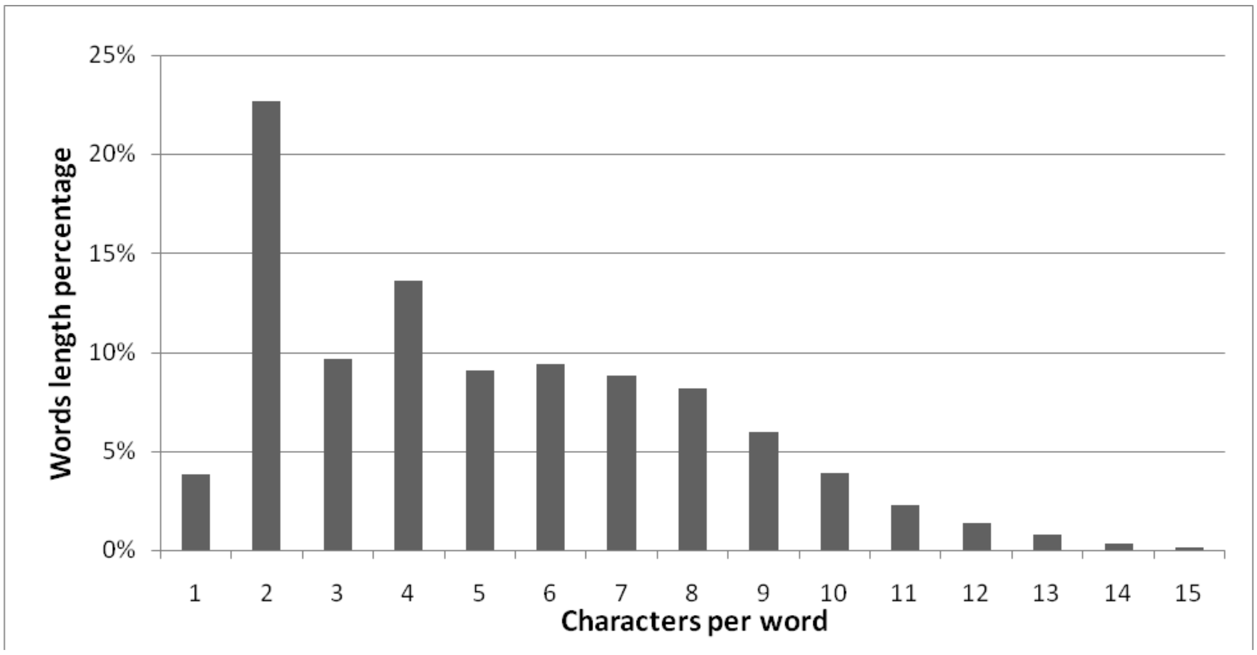


Figure 4.10 Words length distribution in Romanian

4.4 SUMMARY

This chapter has presented the various resources acquired with the purpose of building a speaker-independent large-vocabulary speech recognition system. These resources are indispensable and generally unavailable, thus we invested a lot of time in creating them. Acquisition tools and processing tools were also developed given the need for phonetic, speech and text resources.

The phonetic resources and tools were discussed at the beginning of this chapter. We benefited from an already existing, extensive phonetic dictionary with about 600k phonetically transcribed word forms. Consequently, a huge amount of time that would have been invested in manually creating a phonetic dictionary was saved. However, after the text corpora have been acquired, we reached the conclusion that the existing phonetic dictionary was not enough. Tens of thousands of words, for which the dictionary did not provide any phonetic transcription, were encountered in the texts. This motivated the development of an automatic graphemes-to-phonemes conversion tool. This tool, which was evaluated to be the best among the graphemes-to-phonemes tools available for Romanian, is one of the main contributions of the author of this thesis.

The speech resources and acquisition tool were presented in the second section of this chapter. The section begins with a review of the existing speech databases and, based on the CMU Sphinx suggestions regarding database sizes, concludes that neither of the existing speech materials are enough to create a speaker-independent large-vocabulary CSR system. Consequently, the tedious task of speech database acquisition was approached. A speech acquisition tool, fully designed and implemented by the author of this thesis, was used to acquire most of the databases. To the best of our knowledge the isolated words database (WORDS) and the continuous speech databases (CS_01, CS_02, CS_03 and CS_04) form the largest group of speech materials available for Romanian. In the speech databases acquisition process several people were involved as coordinators and/or speakers. The author of this thesis was one of the main coordinators and speakers.

The chapter ended with the section that presents the text corpora acquired and the processing tools implemented with the final goal of creating a general language model for Romanian. The section begins with a review of the existing text corpora for Romanian and concludes that only the combination of all the corpora could be enough to build a decent, general language model for Romanian. Still, most of the corpora reported to be used by different research groups are not available (neither for research, not for commercial purposes). Consequently, the need of large amounts of written data motivated us to approach the task of collecting several text corpora. After the acquisition process, a text cleaning tool, fully designed and implemented by the author of this thesis, was employed to address several important “cleaning” operations, such as html-to-text conversion, numbers-to-text conversion, special characters handling, etc. A very important and more demanding processing operation was diacritics restoration. This was mandatory for the news corpora which lacked diacritics and required the construction of a diacritics restoration tool. The tool, which is also one of the main contributions of the author, was evaluated and compared to other existing diacritics restoration systems for Romanian and turned out to be one of the best.

Every section in this chapter ends with a statistical analysis of the resources, yielding interesting conclusions regarding phonemes distribution, words distribution, words length distribution, etc. all for the Romanian language.

CHAPTER 5

ACOUSTIC MODELS CONSTRUCTION AND OPTIMIZATION

This chapter presents the various experiments and acoustic models optimizations made during a period of two years, consequently revealing *the evolution* of our ASR system. The experiments were started when the speech databases acquisition process was still in progress, therefore the reader will note that the earlier experiments refer only to some parts of the WORDS database and some parts of the CS_01 database. Secondly, the language models employed in the experiments presented in this section are basic word-loop language models because, at that time, we had no text corpora to build n-gram language models. Nevertheless, the usage of basic language models was also beneficial because the acoustic models improvement could have been emphasized. Finally, the evolution revealed in this chapter was also caused by the knowledge and the experience the author has gained during this period of time.

In conclusion, this chapter starts with some general, introductory issues regarding speech units (selection, context-dependency and clustering), development strategy and database homogeneity. The chapter continues by presenting the optimization experiments made on the WORDS database, including model topology optimizations and decoding speed optimizations. The chapter concludes with continuous speech recognition experiments made on the CS_01 and CS_02 databases. These experiments show the evolution and improvement of our acoustic models.

5.1 Introductory Issues

The main theoretical issues regarding acoustic modeling for automatic speech recognition were presented in Section 2.4. In the end, the section concluded that state-of-the-art large-vocabulary speech recognition systems use Bakis-type Hidden Markov Models (HMMs) with Gaussian Mixture Models (GMMs) as output pdfs to model sub-words speech units such as context-dependent phones (triphones) or senones. The HMMs model these speech units using perceptual acoustic features (MFCCs or PLP coefficients) extracted out of the original time-domain speech signal.

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, as shown above, and to tackle all the other issues regarding the design of the models, acoustic features selection, speech units selection, etc. in an empirical manner.

5.1.1 Speech units selection

The selection of the speech units was the first issue that we approached. It is obvious, as argued in Section 2.4.3, that for a large-vocabulary continuous speech recognition system we cannot use words as basic speech units. They are neither trainable (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). Consequently, sub-words speech units such as context-independent phones (simply called phones), context-dependent phones (triphones) or syllables have to be employed. The trainable attribute (there should be enough data to estimate the parameters of the unit) of a properly chosen speech unit, as discussed in Section 2.4.3, limited our possibilities to phones and triphones. The reason: we do not have enough occurrences to train syllable models neither in the WORDS database, nor in the CS_01 database (Table 5.1).

Table 5.1 Speech units in the WORDS and CS_01 speech databases

	WORDS database			**CS_01 database**		
	phones	**triphones**	**syllables**	**phones**	**triphones**	**syllables**
number of different models	36 models	7359 models	8321 models	34 models	4524 models	n/a
total number of occurrences	95801	95801	37078	82873	82873	n/a
less than 10 occurrences	1 type	5235 models	8076 models	0 models	2670 models	n/a
10 – 20 occurrences	0 models	980 models	82 models	1 models	685 models	n/a
20 – 50 occurrences	1 models	832 models	64 models	0 models	710 models	n/a
50 – 100 occurrences	0 models	210 models	42 models	1 models	269 models	n/a
more than 100 occurrences	34 models	103 models	58 models	32 models	191 models	n/a

As Table 5.1 shows, the isolated words database (WORDS) is large enough to suitably train phone models, and could be used to train triphone models if some efficient parameter-sharing techniques are adopted in order to overcome the data sparseness problem (a very small number of occurrences for over 5000 models). Nevertheless, syllable models have no chance of being suitably trained on this isolated words database as over 6000 models have less than two occurrences. Although this could be regarded as a disadvantage for the isolated words database, we have to remember that its development purpose was to cover all the syllables in Romanian and not to create sufficiently numerous occurrences for syllable modeling.

Regarding the continuous speech database CS_01, one can clearly see that, as it was built from a relatively small number of phrases (which contain only common words), it does not include all the speech units' types that the isolated words database contains. Two of the least frequent phones in Romanian (o2 and y), which had only a few occurrences in the WORDS database, do not appear in the CS_01 speech database at all. As for the triphones, about 2600 types that had less than 10 occurrences in the isolated words database do not appear in the continuous speech database at all.

Another important issue to point out in Table 5.1 is that the total number of phones (and triphones) occurrences is similar between the two databases; from this point of view the isolated words database is only 13.5% larger.

In conclusion, the purpose of the above argumentation was to demonstrate that, given these speech database, only phones or triphones could be employed as basic speech units. Even if most of the studies and books [Huang, 2001] vote in favor of triphones, we have decided to experiment with both types of speech units (context-independent and context-dependent). Moreover, one of our isolated words recognition studies [Cucu, 2011a] investigates the possibility of using a mixed phones-triphones system.

Even though triphones are shown to yield better results [Huang, 2001] thanks to their more suitable context adaptation, their usage has to overcome an important caveat: due to their constructive nature, the number of different triphones is a lot larger than the number of different phones. This is a drawback because it affects trainability on small databases (the same speech database could be large enough to train phone HMMs, but could be insufficient to properly train triphone HMMs). Triphone modeling assumes that every triphone context is different. Actually, many phones have similar effects on the neighboring phones. All the triphones that share the central phone are acoustically very similar; hence their models central states have similar parameter values, as shown in [Beulean, 1997; Liu, 1999; Steward, 2002]. Taking these into account we have decided to use the tied-states technique: triphones that share the central phone are to be modeled with HMMs that share the central states. This approach reduces the number of parameters that have to be trained for the triphones ASR system and thus reduces the risk of ending up with undertrained HMMs.

The tied-states technique has been used in all the isolated words recognition experiments which were made using the HTK Toolkit. After the migration to CMU Sphinx, which implicitly uses senones (state-dependent output pdfs across different phonetic models), we have used only context-dependent triphone models with clustered states for all the continuous speech recognition experiments.

Regardless of the chosen speech units, the speech pauses have to be modeled in a similar manner using HMMs. We need to differentiate two types of pauses: the long pause at the beginning of the phrase or between phrases, and the short pause that one usually makes between two words. The short pause has been modeled by cloning one of the central states of the long pause model and using the previously-mentioned tied-states technique, resulting in a one-emitting-state HMM.

5.1.2 The hierarchical training strategy

The previous section presented the issues that have to be taken into consideration when designing an HMM-based ASR system. The output of this process is an initial set of HMMs that have to be trained using the training speech database (the HMMs enclose model parameters which now have to be estimated).

In order to design the training strategy for the ASR system several matters have to be considered. First of all, the isolated speech unit training would definitely work well if there were a large enough database of isolated speech units. In the case discussed here (phones are being used as basic speech units), the isolated phones database (PHONES) has been acquired for initialization purposes only and is not large enough to be used on its own for training the system. Moreover, building a large isolated phones database is a tedious and time-consuming process.

These being said, the isolated speech unit training will only be used for initialization, while the embedded-training technique will be employed for the system training. This technique (embedded-training) uses the Baum-Welch re-estimation algorithm and only requires information about the order (the sequence) of the speech units in a given utterance to perform the HMM training. Every basic speech unit is typically modeled with one HMM and all these basic HMMs can be concatenated to form words HMMs, which can be further concatenated to form word sequences HMMs. This is the mechanism which allows us to eventually estimate the probability of a sequence of words given the initial speech data. Moreover, this mechanism is the basis for the embedded-training process: the estimation of parameters for multiple, concatenated HMMs, given a speech signal composed of a sequence of speech units.

Although embedded-training is quite comfortable from the database point of view, it is very vulnerable to wrong speech units' alignments. The estimation process stops once the probability that the estimated models generate the given speech utterance reaches a maximum. If, for some reason, this maximum is a local and not the global one, then the training will not be optimal. In order to have the best alignment, we have proposed a hierarchical system development strategy that involves several steps, which will be presented further on.

First step: HMM system design. This step involves 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 will be a set of "prototypes" – all the models parameters are given default values.

Second step: System initialization. The set of "prototypes" resulted from the previous step 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 consists in computing the voice features for each frame of each speech utterance, and using the statistical results as initial model parameters. The output of this step will be a set of roughly initialized HMMs.

Third step: Embedded phone HMM training. At this point the initialized models are trained using the isolated words database (WORDS). The training method employed is obviously embedded-training, 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 step is a robustly trained HMM set. Its quality can be assessed and then, based on this evaluation, several design parameters can be adjusted. Of course, any design adjustment would mean restarting the development from step 1 and ending with evaluating the performance again. Several step-by-step design adjustments have been made (as shown in Section 0) in order to reach the best performance.

Fourth step: Embedded triphones HMM training. This step consists of two parts: the first one is a design adjustment that aims to create triphone models 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 tied-states parameter-sharing technique is also used as explained in the previous section. 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.

Fifth step: Embedded triphone HMMs training. This step uses the models obtained at step four and retrains them using the WORDS database and the CS_01 database. Finally, the continuous speech ASR system performance is evaluated.

In conclusion, this hierarchical training strategy is designed to initialize and train the HMMs with small, but finer-grained annotated databases at first and large, but coarser-grained annotated databases as we go further. In this way we try to minimize the possibility of wrongly aligning the speech units while applying the embedded-training method.

5.2 Acoustic Models for Isolated Words Recognition

This section presents the isolated words recognition experiments we have employed in order to find the best acoustic models. The experiments follow closely the hierarchical training strategy presented in the previous section.

5.2.1 Experimental setup

For all the experiments in this section we have used two speech databases: the WORDS database and the PHONES database. The PHONES database contains single-phone audio files and will be used only for the initialization step (as specified in the hierarchical training strategy). This database was complete at the time these experiments were performed. A more detailed description and analysis of this database were made in Section 4.2.2 and Section 4.2.3.

The WORDS database is an isolated words database. At the time these experiments were performed, only 5 speakers (2 males and 3 females) had recorded the list of 10000 different words. Most of the experiments in the following sections were made using the whole database, but some of them (will be outlined at the right time) were made only for a certain speaker. A more detailed description and analysis of this database were made in made in Section 4.2.2 and Section 4.2.3.

These two databases had to be split into a training part and an evaluation part. Due to the small size of the PHONES database we have decided to use it all for training. With no separate evaluation part left, the phones recognition experiment made after the initialization is not very "fair". Regardless, we will present these results also and outline their moderate importance.

The WORDS database consisted of 50000 audio files (10000 per speaker) which were split into a training part comprising 45000 files (9000 per speaker) and an evaluation part comprising 5000 files (1000 per speaker). The split was made based on the files ids as follows: the files with ids between 5000 and 5999 were used for evaluation and all the other files were used for training. This split was made without any theoretical basis and could influence the recognition results if the database is not homogeneous. Consequently, we were required to make some experiments and validate the homogeneity of the WORDS database (Section 5.2.2 presents these results), just to make sure all the other results will be conclusive.

All the experiments in the following sections employ basic word-loop language models. This means all possible phones (for the phones recognition experiment) or all possible words (for all the other experiments) are equally probable to be outputted at all times, regardless of the output history. Of course, the size of the phones/words vocabulary is a key factor here: we used 36 possible phones (for the phones recognition experiment), respectively 10000 possible words (for all the other experiments).

5.2.2 Isolated words database enlargement and homogeneity experiments

The isolated words database (WORDS) was subject to a set of experiments that aimed to determine whether the database is large enough and whether the phonemes that make up the words in the database are uniformly distributed over the whole set of files.

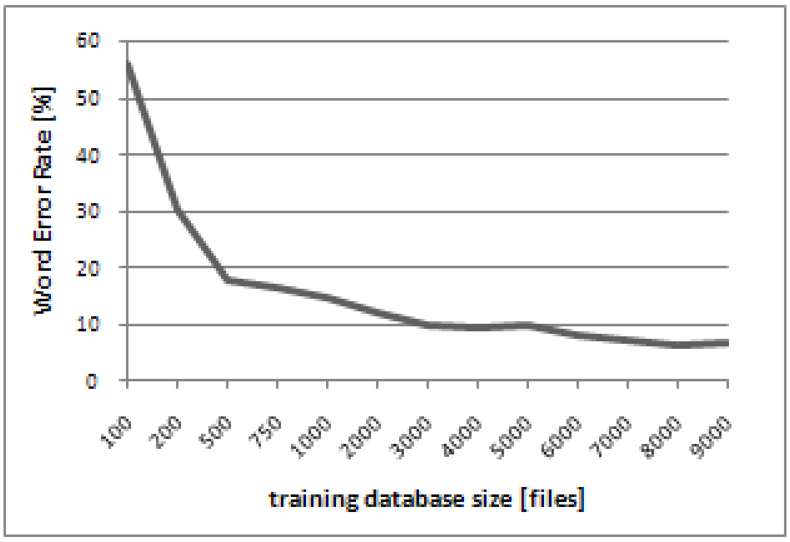


Figure 5.1 Database enlargement experiment

Section 0 will approach the ASR system design and will conclude with a “most suitable acoustic models design” for the phones ASR system. This design setup is used in this section to perform nine tests that aim to decide whether a smaller training database (of less than 9000 words) is enough to obtain the same ASR performance as in the case when the whole training database (all 9000 words) is utilized. The tests were performed only for speaker 1 because the other audio files for the other speakers were not available at the time. Figure 5.1 presents the experimental results. The figure shows, the ASR system performance significantly grows when the training database size increases from 100 up to 6000-7000 words, but starts to saturate at about 8000-9000 words. Consequently, the first few hundred audio files are very important, while the last few thousands audio files do not bring to much performance gain.

All the experiments in Section 5.2 were performed by splitting the database into an evaluation part (the audio files with ids in the range 5000 – 5999) and a training part (all the other files). The next experiment aimed to guarantee that splitting the database into two parts (a training database and an evaluation database) can be done in a random manner without artificially increasing or decreasing the ASR system performance. This experiment has also been performed only for speaker 1. Table 5.2 summarizes the results for the different database splitting experiments.

Table 5.2 Database homogeneity experiments

Speaker	**HMM configuration**	**Database**		**WER**
		training files	**testing files**	
1	“most suitable acoustic models design” for phones (see Section 0)	all except testing files	0000 – 0999	5.94%
			1000 – 1999	5.92%
			2000 – 2999	5.81%
			3000 – 3999	5.22%
			4000 – 4999	5.51%
			5000 – 5999	6.74%
			6000 – 6999	6.73%
			7000 – 7999	7.72%
			8000 – 8999	6.71%
			9000 – 9999	9.44%

The results presented in Table 5.2 show some important aspects regarding the WORDS database and uncover a method that can be used to detect database inconsistencies possibly caused by acquisition errors. For example the ASR system presented on the last row has a 50% higher word error rate when compared to almost all of the other ASR systems in the table. One of the reasons for the inconsistency could be the fact that the speaker was anxious to finish the database acquisition task (this ASR uses the last recorded 1000 files in the database for testing purposes). This assertion is also sustained by Figure 5.1 which exhibits an increase in the descending trend of the word error rate when these last 1000 files are appended to the training database. This means that these last 1000 files do not help, but actually harm the training process.

5.2.3 *ASR design experiments*

The first step in the hierarchical training strategy presented in Section 5.1.2 consists in choosing the HMM system design. The first configuration we have experimented with was based on Bakis-type context-independent phone (simply called phones) HMMs and Mel-Frequency Cepstral Coefficients (MFCCs). In the initial configuration, which was supposed to be optimal for speech recognition ([Jurafsky, 2009]), every HMM had 6 states (among which 4 were emissive) with 2 Gaussian mixtures per state.

The phone recognition results (obtained after the second step in the hierarchical training strategy) were computed, as mentioned in the experimental setup section, on the same PHONES training database. Due to the small amount of data the same database was used for training and evaluation. Nevertheless, this experiment is only intended to roughly evaluate the phones classification power of the ASR system after initialization. The performance figures (in terms of phones error rates) are presented individually, per model type (phones) in Table 5.3.

Table 5.3 Individual PER for the phones in the PHONES database

Symbol	PER [%]		Symbol	PER [%]		Symbol	PER [%]		Symbol	PER [%]
a1	20		g	12		k	0		s1	16
a	22		h	8		l	26		s	11
b	13		i1	-		m	20		t1	10
d	22		i2	13		n	29		t	13
e1	8		i3	5		o1	0		u	19
e	23		i	11		o2	-		v	22
f	15		j	9		o	0		w	13
g1	8		k2	-		p	0		y	-
g2	-		k1	3		r	8		z	23

Going further to the third step in the training strategy, we have used the training part of the WORDS database to train the 6-states HMM system. A speaker-independent system was trained and evaluated using training and evaluation files from all the 5 speakers and, for comparison, 5 speaker-dependent systems were also developed and evaluated using training and evaluation files in a “per speaker” manner. The performance figures (in terms of word error rates) are presented in Table 5.4. This table shows that, as expected, the word error rate for the speaker-independent ASR is higher than the word error rate for the speaker-dependent ASR. This issue will be overcome by enlarging the database.

Table 5.4 Speaker-dependent vs. speaker-independent ASR systems

Speaker	HMM configuration	WER
1, 2, 3, 4, 5	models for phones 6-state HMMs 2 GMs per state 12 MFCCs + energy + 1st order derivatives	51.07%
1		22.33%
2		21.93%
3		30.58%
4		25.55%
5		28.57%

The results presented in Table 5.4 will be regarded as a baseline for further experiments. As mentioned in Section 5.1.2, the third step in our development strategy involves numerous ASR design experiments that aim to find the best HMM configuration for the phone models. The tests for finding the best HMM configuration involved modifying the number of states in the HMMs, modifying the number of Gaussian mixtures per state and using the first and second order temporal derivatives for the voice features.

Table 5.5 presents the word error rates for ASR systems that differ by the number of voice feature coefficients. The results clearly show that the most powerful ASR system is the one that takes into account only the MFCC coefficients and their first order temporal derivatives. These results are closely related to the size of the database: a small database can be used to suitably train only a few parameters, while a large database can be used to train more voice features.

Table 5.5 Varying the number of voice features

Speaker	HMM configuration		WER
1, 2, 3, 4, 5	models for phones 6-state HMMs 2 GMs per state	12 MFCCs + energy	80.34%
		12 MFCCs + energy + 1st order derivatives	51.07%
		12 MFCCs + energy + 1st and 2nd order derivatives	65.03%

The HMM’s number of states and the number of Gaussian components per state are important parameters which might influence the performance of an ASR system. We experimented starting from the classical Bakis model (6 states HMMs with 2 GMs per state) and we varied the number of states per HMM as well as the number of GMs per HMM state, in order to find the optimal model with respect to the performance of the ASR system. The number of states in an HMM is a design parameter that has also a physical interpretation: an HMM with more states models longer phonemes better than an HMM with fewer states, and vice-versa for shorter phonemes. Having this in mind, after reaching an optimal performance for an ASR system with a fixed number of states for the HMMs, we tried to vary the number of states in a “per model” manner.

Table 5.6 Varying the number of GMs per HMM state

Speaker	HMM configuration		WER
1, 2, 3, 4, 5	models for phones 6-state HMMs 12 MFCCs + energy + 1st order derivatives	2 GMs per state	51.07%
		3 GMs per state	43.96%
		4 GMs per state	64.77%
2		1 GM per state	21.93%
		2 GMs per state	21.93%
		3 GMs per state	23.64%

A lot of experiments were performed in order to determine the optimal numbers for these two parameters. Table 5.6 and Table 5.7 summarize some results of these design experiments.

Several conclusions can be drawn from the results in Table 5.6. It is obvious that the optimum number of Gaussian mixtures for the speaker-independent ASR system and for the given database is three. Also, it is clear that for a speaker-dependent ASR system this parameter is less important and could be set to one in order to optimize the computation time. The results are not surprising at all:

- For a speaker-dependent system one Gaussian mixture is enough to model the variability of a single voice. Even though the phones are not uttered exactly the same, they are very similar and so are the cepstral coefficients for every state in the model.
- For a speaker-dependent system (and in our case only five speakers in the training database) three Gaussian models are needed to best model the variability of these five voices. The phones are uttered in a particular manner by every speaker and thus the cepstral coefficients are also quite different. We expect that for a database with more than five speakers the optimum number of Gaussian mixtures to be larger.
- As the HMMs better model the training database, the recognition rate difference between the speaker-independent and speaker-dependent ASR systems gets smaller.

Table 5.7 Varying the number of states per HMM

Speaker	HMM configuration		WER
1, 2, 3, 4, 5	models for phones 2 GMs per state 12 MFCCs + energy + 1st order derivatives	6-state HMMs	51.07%
		7-state HMMs	52.48%
		8-state HMMs	43.60%
		9-state HMMs	36.23%
		10-state HMMs	32.21%
		11-state HMMs	30.33%
		12-state HMMs	26.95%
		13-state HMMs	30.12%
		14-state HMMs	28.72%

The results presented in Table 5.7 show that the ASR system performance grows with the increase of the number of states for the HMMs and reaches an optimum when the number of states is 12. The results are somehow surprising because 12 states is quite a large number compared to the Bakis model (with only six states), about which the literature says to have the smallest word error rate for small vocabularies.

Up to this point the number of states has been kept constant over all the phone models which is not necessarily right. Some phones (such as consonants, for example) are clearly shorter that other phones (such as vowels, for example). The optimum number of states obtained so far (12) is in fact a weighted average of all the phones optimum number of states. This idea led to our next experiment: modifying the number of states in a “per model” manner. In order to decide which would be the optimum number of states for a particular model, we have used the average length for that phone, as given by the 12 states ASR system. This average length (expressed in number of states) is summarized for some phones in Table 5.8.

Table 5.8 Average length for some phonemes

Phoneme	k1	l	m	p	r	s	u
Average length [no. states]	17	8	10	9	6	15	9

This data has been used to re-design the 12-state HMMs ASR system in order to have different HMM models for the seven phones listed in Table 5.8. Seven different ASR systems have been generated and trained using the same development strategy and their performance was evaluated. The results are presented in Table 5.9.

Table 5.9 Varying the number of states in a "per phone" manner

Speaker	HMM configuration		WER
1, 2, 3, 4, 5	models for phones 12-state HMMs 3 GMs per state 12 MFCCs + energy + 1st order derivatives	all HMMs – 12 states	15.11%
		HMM for *k1* – 17 states	15.53%
		HMM for *l* – 8 states	16.46%
		HMM for *m* – 10 states	15.31%
		HMM for *p* – 9 states	15.92%
		HMM for *r* – 6 states	17.08%
		HMM for *s* – 15 states	16.34%
		HMM for *u* – 9 states	15.39%

Although the preliminary observations seemed right, the performance of the ASR system did not increase when we modified the number of states for one of the HMMs, to take into account the most appropriate length for that particular phone. In conclusion, the model size and topology should be kept identical over all the models.

While all these design experiments were in progress, the WORDS database was also being extended and, in the end, we were able to evaluate the ASR systems with an 8-speaker database. Of course, for the 8-speaker ASR we had to redo the number of Gaussian mixtures per state optimization process. It was no surprise to see that better results were obtained with more than 3 GMs per state.

Table 5.10 Best isolated words recognition results

Speaker	HMM configuration		WER
1 – 5	models for phones 12-state HMMs 12 MFCCs + energy + 1st order derivatives	3 GMs per state	15.11%
1			6.74%
2			5.53%
3			8.25%
4			6.74%
5			4.83%
9			11.67%
12			3.22%
15			29.38%
1 – 5, 9, 12, 15		3 GMs per state	23.64%
1 – 5, 9, 12, 15		6 GMs per state	15.47%

The best results for the phone models that have been obtained for the eight speakers are presented in Table 5.10. As the table shows, the results are really good for most of the speaker-dependent systems. The poorer results for speaker 9 and 15 are probably due to inconsistent recordings and they are probably also responsible for the relatively large WER for the 8-speakers ASR system.

The embedded-training technique does not issue the best results after the first training iteration of the Baum Welch algorithm. Consequently, several training iterations have to be employed until the word error rate for the ASR system reaches an optimum. Figure 5.2 presents the performance variation over several training iterations for the optimal speaker-independent (5-speaker) ASR system.

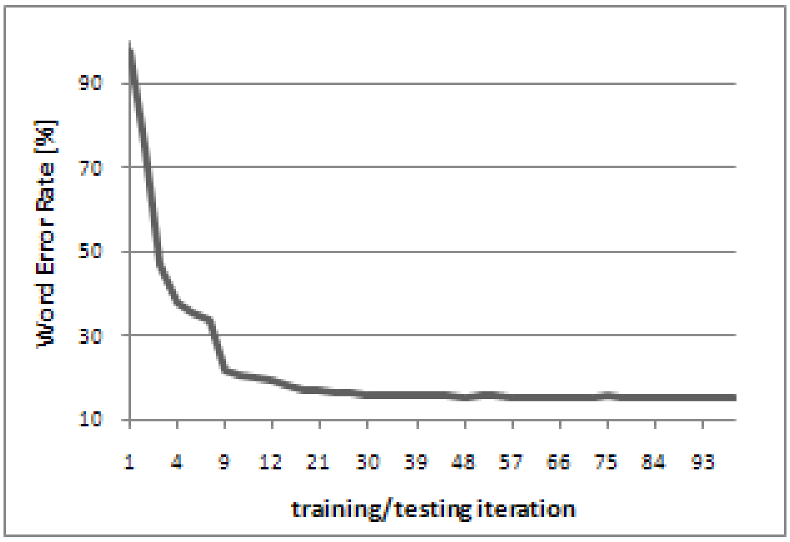


Figure 5.2 WER variation over the training iterations

Going further, we have continued with the development steps (step 4 in the hierarchical training strategy), and created context-dependent phone (triphone) models out of the context-independent phone models. Table 5.11 compares the performance of the phones ASR system and the triphones ASR system.

Table 5.11 Varying the type of models: phones vs. triphones

Speaker	HMM configuration		WER
1, 2, 3, 4, 5	12-state HMMs 3 GMs per state 12 MFCCs + energy + 1st order derivatives	models for phones	15.11%
		models for triphones	11.41%

Although the results presented in Table 5.11 show that the triphones ASR system performs better than the phones ASR system, a more in-depth analysis proved that the gain in recognition rate is not purely incremental. In other words, the triphones ASR system fails to recognize some of the words that are recognized by the phones ASR system and vice-versa. Namely, the set of words recognized by the triphones ASR system (88.59% of the total words) does not include all the words recognized by the phones ASR system (84.89%). Figure 5.3 illustrates the actual intersection of the word sets. Figure 5.3 shows that if the triphones ASR system would also be able to recognize the words recognized by the phones ASR system or vice-versa we would obtain a system with a recognition rate of 95% (or a word error rate of 5%).

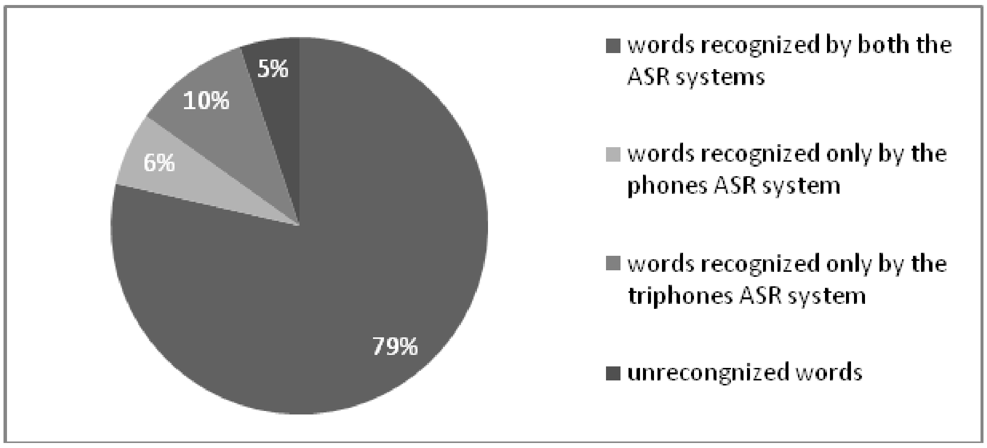


Figure 5.3 Recognized and unrecognized word sets for the ASR systems presented in Table 5.11

The previous result supports the idea of creating a mixed ASR system that could make use of both phones and triphones as basic speech units. The two sets of models trained and evaluated so far can be utilized to perform a double recognition. In the end, the two recognition results are compared and the better recognized word is selected. The selection of the better recognized word is in this case very simple, because the two recognition probabilities outputted by the two systems can be directly compared.

The experiments showed a significant performance improvement for the mixed phone-triphone ASR system: word error rate of only 7.28% (two times lower than the phones ASR word error rate). The downside of this system is that its recognition time is the sum of the recognition times for the composing systems plus a phones/triphones decision time.

5.2.4 Language restrictions experiments

All the experiments presented in the previous section were done using a basic word-loop language model with no restrictions. The ASR systems were required to recognize among the words in the extended 10k words vocabulary, although the testing audio files contained only a subset of 1000 words. Moreover, the ASR systems were allowed to output any combination of any words for every audio file, although every audio file contained only one word. Thanks to the fact that we know that we are dealing with single-word audio files and that the words uttered in these evaluation files are all part of a 1000-words vocabulary, we can create more restrictive language models. Obviously, these restrictions can be imposed only in an isolated words recognition scenario (continuous speech recognition will be dealt with separately).

Table 5.12 presents the word error rates for the phones-based ASR system, for the triphones-based ASR system and for the mixed-models ASR system, given three types of language restrictions. The first experiment (lines 1, 4 and 7 in the table) uses an unrestricted word-loop language model (the same as in the experiments presented in the previous section). The second experiment (lines 2, 5 and 8) uses a language model which “knows” that every audio file contains one single word, while the third experiment (lines 3, 6 and 9) employs a word loop language model with a reduced vocabulary (1000 words).

Several conclusions can be drawn from Table 5.12. First, for the phones-based ASR system the language restrictions have a significant impact, decreasing the word error rate about four times. Second, for the triphones-based ASR system these language restrictions seem to be less significant. Third, a conclusion that is, in fact, an outcome of the first two: the triphones-based ASR system performs better that the phones-based ASR system only when loose language restrictions are being applied. Consequently, triphones-based systems should be even better in large-vocabulary continuous speech recognition, but in the case of isolated words recognition, when strict language restrictions are imposed, phone-based systems are better. And finally, regardless of the language restrictions, the mixed-models ASR system is the best.

Table 5.12 Imposing a set of language restrictions

Speaker	HMM configuration	Word-loop language model restrictions		WER
		Vocabulary size	Words sequence restrictions	
1, 2, 3, 4, 5	models for **phones** 12-state HMMs 3 GMs per state 12 MFCCs + energy + 1st order derivatives	10000 words	any combination of any words	15.11%
		10000 words	only one word	8.47%
		1000 words	any combination of any words	3.76%
1, 2, 3, 4, 5	models for **triphones** 12-state HMMs 3 GMs per state 12 MFCCs + energy + 1st order derivatives	10000 words	any combination of any words	11.41%
		10000 words	only one word	10.99%
		1000 words	any combination of any words	7.08%
1, 2, 3, 4, 5	**mixed phone-triphone** models 12-state HMMs 3 GMs per state 12 MFCCs + energy + 1st order derivatives	10000 words	any combination of any words	7.28%
		10000 words	only one word	5.61%
		1000 words	any combination of any words	2.27%

5.3 Time Optimizations for Isolated Words Recognition

All the experiments presented in Section 5.2 aimed to find the best models from the recognition accuracy perspective and did not raise the *real-time* question. In some speech recognition applications real-time might not be a must, but in most of the cases it is. In speech recognition, the real-time factor is defined as the ratio between the processing (recognition) duration and the audio file length or duration. A lower-than-1 real-time factor is desirable for most common speech recognition applications such as dictation, human-computer dialogue systems, etc.

This section introduces an innovative three-step recognition method that helps achieve the real-time desiderate for the HTK-developed isolated words recognition systems. The baseline to compare our method with is presented in Table 5.13. These are, in fact, the ASR systems presented in Table 5.11 for which the real-time factor was also computed. The real-time factor (RTF) was computed as follows: the durations of all the audio files in the evaluation database were summed up (we obtained approximately 2 hours of speech) and the duration of the recognition process is divided by this number. For example, for the two experiments presented in Table 5.13, the recognition process took approximately 6.5 hours, respectively 3.5 hours.

Two important things should be noted about the results presented in Table 5.13. First, the triphones ASR system is more accurate and also twice as faster as the phones ASR system. And second, both ASR systems are still far from achieving the real-time recognition goal.

In order to better understand these results and try to get closer to the real-time recognition goal, the recognition process algorithm has to be analyzed and tuned.

Table 5.13 Best recognition results using the usual recognition method

HMM models	Language information		Recognition algorithm		WER	RTF
	Vocabulary size	Words sequence restrictions	Type	Steps description		
phones	10000 words	any combination of any words	one step recognition	recognize words	15.11%	3.24
triphones					11.41%	1.77

5.3.1 Token passing algorithm analysis

The process of speech recognition is to find the best possible sequence of words (or units) that will fit the given input speech. It is a search problem, and in the case of HMM-based recognizers, a graph search problem. To solve this problem, the well known Token Passing algorithm [Young, 1989] (a specific version of the Viterbi algorithm) is used.

Before the actual recognition starts, a search graph is constructed based on the set of acoustic models and the language model (or language restrictions). The nodes in this search graph are HMM states, while the transitions are of several types: a) intra-HMM transitions, b) inter-HMM transitions and c) inter-words transitions. The search graph is meant to represents all possible sequences of phonemes in the entire language of the task under consideration. For example, the search graph in Figure 5.4 is designed for a digit recognition task. It can decode any speech input which contains the words *zero* (zero), *unu* (one), *doi* (two), ..., *nouă* (nine). As you can see the search graph is composed of HMMs for basic speech units, such as the phones *u*, *n*, *d*, *o*, *i3*, *w*, *a1*, etc., which are concatenated to form the words in the task vocabulary: *unu*, *doi*, *nouă*, etc.

All the transitions in the search graph are probabilistic: the intra-HMM transition likelihoods have been computed during the acoustic model training (Baum-Welch algorithm), the inter-HMM transitions are straight forward and finally, the inter-word transitions likelihoods are given by the language model (or language restrictions). For example, a basic word-loop language model would generate a search graph in which all inter-word transitions are as probable (all words are as probable and any word can follow any other word with the same probability). On the contrary, an n-gram language model would have specific probabilities for every word and specific probabilities for sequences of three words and would generate a search graph in which the inter-word transitions will not be as probable.

Constructing the above search graph requires knowledge from various sources. It requires a dictionary, which maps the word *unu* to the phonemes *u*, *n* and *u*, the word *doi* to *d*, *o* and *i1*, etc., it requires the acoustic model to obtain the HMMs for the phonemes and finally, it requires the language model to obtain the inter-word transition likelihoods.

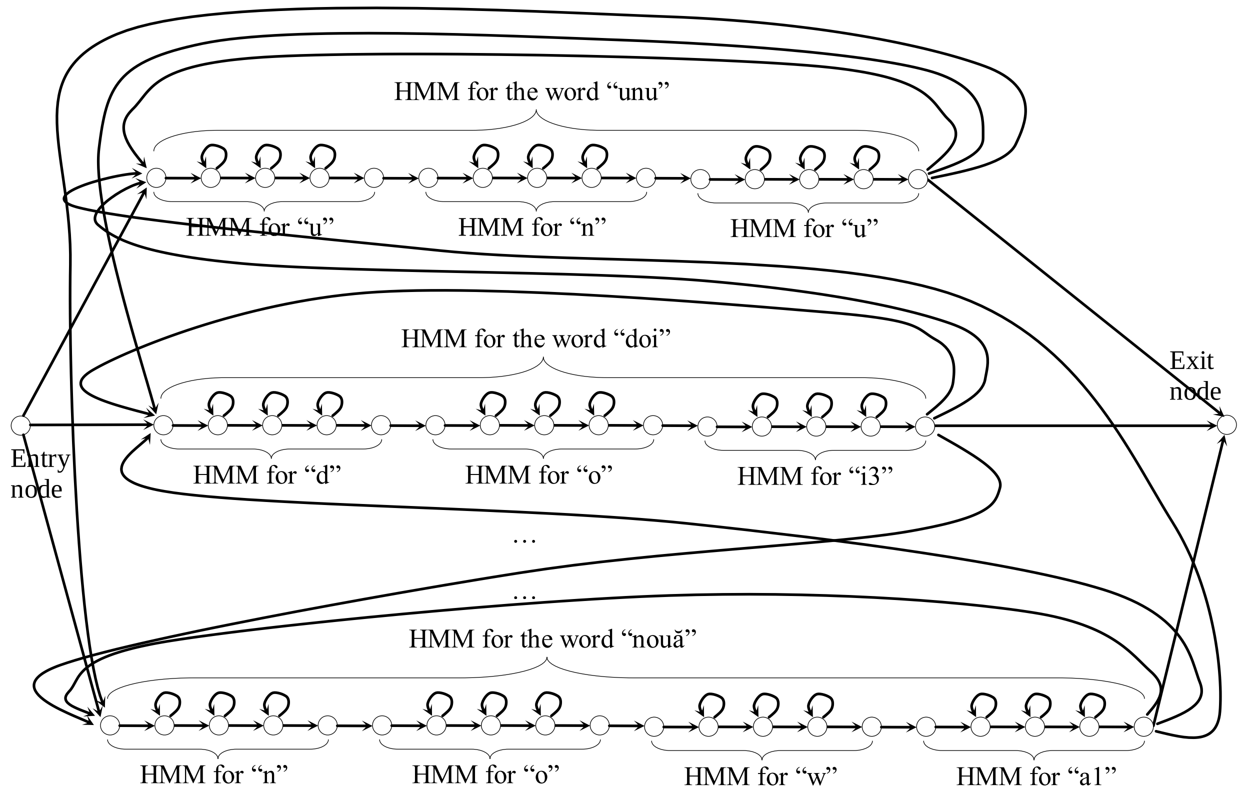


Figure 5.4 A simple search graph example

Once this search graph is constructed, the sequence of parameterized speech signals (i.e., the features) is matched against different paths through the graph to find the best fit. The best fit is usually the least cost or highest scoring path, depending on the implementation.

As you can see from the above graph, a lot of the nodes have self transitions. This can lead to a very large number of possible paths through the graph. Moreover, note that this is a tiny search graph, constructed with a 10 words vocabulary, while for large-vocabulary tasks we would generally have 64k words vocabularies. This leads to an even larger number of possible paths through the graph. As a result, finding the best possible path can take a very long time. The number of search paths can be reduced during the search, using heuristics like pruning away the lowest scoring paths.

After the last feature vector (the one created for the last frame of the input speech signal) is decoded, we look at all the paths that have reached the final exit node. Among these, the path with the highest score is the best fit, and a result taking all the words of that path is returned.

Taking these into account it is clear that the recognition time increases with the complexity of the search graph and this complexity depends on several factors:

- The number of words in the task-specific vocabulary. A larger vocabulary leads to a more complex search graph.
- The complexity of the language model (or restrictions). A word-loop language model is a trivial, but very general language model (it allows transitions from every word to every other word in the search graph) and this leads to an increased complexity for the search graph. For a particular application, for example single-word recognition, the transitions from any word-end to other words would have null probability (only the transition to the exit node will be non-null), thus enormously simplifying the search graph. If an n-gram language model is employed, the search graph would be even more complex because the transitions from a sequence of words to a new word need to be taken into account (the search graph becomes more complex as the history increases).
- The number HMMs or the total number of *different* HMM states. On various search paths the n^{th} voice features vector (corresponding to n^{th} frame in the input speech signal) might be matched against the same HMM state. For example, the words *şase* (six) and *şapte* (seven) start with the same phone, consequently the same HMM, consequently the same HMM state. The match of the first voice feature vector, while trying to decode the speech signal as *şase* or *şapte*, will not be done twice because, in fact, the same computations are employed. Moreover, if the acoustic model uses state-sharing techniques then these situations, when the computation is done once for more search paths, might occur even more frequently. In conclusion, a system with less unique HMM states would be faster. Even if this does not simplify the search graph, it leads to a shorter computation time.

The recognition time also increases depending on the time it takes to match a voice feature vector to a given HMM state. Less complex HMM states would lead to a shorter computation time. Consequently, the recognition time is also influenced by:

- The size of the voice feature vector. If the feature vector is composed of only 12 MFCCs than there would be three times less mathematical operations to be made compared to the case in which the first and second order temporal derivatives would be used.
- The number of Gaussian mixtures per state. Less mixture components per HMM state leads to less mathematical operations to be made.

Now we can better understand and explain why the triphones ASR system is faster than the phones ASR system. Although the number of HMMs is higher (7372 triphone models compared to 41 phone models) and apparently increases the number of unique HMM states, the tied-state parameter-sharing technique assures this number remains quite low. Six out of the 10 emitting states per triphone HMM are shared, thus the number of unique HMM states is only 40% higher than in the phones ASR case.

The algorithm analysis presented above unfolds a range of optimization possibilities. The first one that arises regards the possibility of investigating only some search paths instead of investigating them all. A simple way to prune the search space for the Token Passing algorithm is the beam search. Beam search is a widely used search technique for speech recognition systems. It is a breadth-first style search, but, unlike traditional breadth-first search, beam search only expands search paths that are likely to succeed at each level. Only these search paths are kept in the beam, and the rest are ignored (pruned out) for improved efficiency.

The beam search algorithm keeps track of the most probable search paths by creating and updating (at each stage) a sorted list according to the search path likelihoods. From time to time these stacks are pruned out and the worst search paths are discarded. Of course, a search path which was pruned out at some moment could have turned out to be the best output word sequence, but this risk has to be taken. The most popular pruning method employs a threshold by which a search path is allowed to be worse than the best one in the stack. All other search paths are pruned out. This threshold is also denoted beam width, thus the name of the algorithm. The beam search algorithm assures that the computational complexity becomes manageable and even a long speech sample can be decoded in a decent amount of time. In fact, the beam width or the threshold can be tuned for better performance or speed. Our experiments proved the above assertions. Table 5.14 presents some experiments done with various beam widths.

Table 5.14 Using different beam width values

HMM models	**Language information**		**Recognition algorithm**		**WER**	**RTF**
	Vocabulary size	**Words sequence restrictions**	**Type**	**Beam width**		
phones	10000 words	any combination of any words	one step recognition	no beam	14.68%	15.84
				250	14.75%	4.66
				200	14.81%	4.48
				150	15.11%	3.24
				100	18.41%	1.85

Based on the results presented in the above table, we have decided that the optimal value for the beam width is 150. This value was used for all the experiments presented in Section 5.2. Consequently, the baseline we have set for this section (Table 5.13) already benefits from this type of optimization. Without it, the real-time factor would have been as worse as 15.84 (instead of 3.24).

The next optimizations consider the influence of the vocabulary size and the language restrictions on the complexity of the search graph and, implicitly, on the processing time. Taking into account that our target, in this section, is isolated words recognition and thus we a priori know that the speech sample consists of a single word it’s natural to allow only one-word recognition candidates. Consequently, the transition probability from every word-end to other words will be set to null. The results obtained by applying this language restriction are presented in Table 5.15.

Table 5.15 Applying "only one word" language restriction

HMM models	Language information		Recognition algorithm		WER	RTF
	Vocabulary size	**Words sequence restrictions**	**Type**	**Steps description**		
phones	10000 words	only one word	one step recognition	recognize words	8.47%	0.88
triphones					10.99%	0.71

Two important things should be noted about the results shown in Table 5.15. First, and as expected, the average recognition time decreases a lot. The real-time factor is now less than 1, which means the real-time desiderate is satisfied. Second, the recognition rate for phones ASR system overcomes the recognition rate for the triphones ASR system. Note that these results have also been presented in Table 5.12, but at that time we were only interested in the relative performance of the two systems.

The second optimization possibility implied decreasing the number of words in the vocabulary. In the following experiment we have reduced the vocabulary size from 10000 words to 1000 words in a supervised fashion: we chose the 1000 words based on the reduced list of words used for evaluation.

Table 5.16 Reducing the vocabulary size

HMM models	Language information		Recognition algorithm		WER	RTF
	Vocabulary size	**Words sequence restrictions**	**Type**	**Steps description**		
phones	1000 words	any combination of any words	one step recognition	recognize words	3.76%	0.19
triphones					7.08%	0.15

Table 5.16 presents the results obtained using a smaller vocabulary (1000 words). It's important to note that the ASR systems are more accurate because they have fewer words to choose from and also that the ASR systems are faster because the number of possible HMM combinations decreases.

Although the results in Table 5.16 are appealing we should not forget that this type of language restriction can be applied only to small-vocabulary speech recognition applications (for large-vocabulary continuous speech recognition we would generally use a 64k words vocabulary). Moreover, for specific small-vocabulary applications it would be very useful to find an unsupervised method to reduce the vocabulary size. In the above case we have reduced the vocabulary size from 10000 words to 1000 words thanks to the fact that we knew which words were part of the evaluation set. In the general case we do not have this a priori information and we would like to reduce the vocabulary size in an unsupervised fashion. That is the main focus of the three-step recognition method presented in the next section.

Both language restrictions optimizations are useful in many applications where human-machine interaction is implemented through single commands. In this case each command is represented by one word and the number of commands cannot be very high (less than 1000 commands).

5.3.2 Three-step isolated words recognition

Taking into account the results presented in the previous section and the conclusion that a small vocabulary could improve both the accuracy and the recognition speed of an ASR system we've designed the following three step algorithm (these steps will be applied in real-time to the test speech audio clip):

Step 1. Phone recognition. A basic phone-loop language model is used, along with a 41-phones vocabulary (36 Romanian phones and 5 silence models) to perform speech recognition. It is supposed that this step will be very fast because the number of tokens in the vocabulary is very small (41). The result will be a set of time-ordered phones along with their recognition probability.

Step 2. Vocabulary selection. Based on the most probable phones ranking resulted after step 1, a small words vocabulary is selected. This vocabulary will contain only the words that contain the phones with the highest recognition probability.

Step 3. Words recognition. Isolated words recognition, using a word-loop language model along with the previously selected vocabulary, is performed.

This three-step recognition method raises several issues that will be presented and approached in the following subsections.

5.3.2.1 Step 1 - Selecting the best recognized phones

A first thing that has been noticed is that the ASR system usually recognizes different phones with quite different recognition probabilities. The recognition probability depends, among others, on how well a model has been trained. Thus, a raw comparison between the recognition probabilities of phones *a* and *b*, for example, is not suitable for our purpose.

In order to get more quantitative information about this aspect, we have performed an unrestricted phones recognition on the whole training speech database. Then a DTW algorithm was used to align and compare the reference text with the hypothesis text and select all the correctly recognized phones. The recognition probabilities for all these phones were utilized to compute the average recognition probability for each phone. These probabilities are a measure of how well the acoustic model managed to learn the phones in the training database. Some of the results are presented in Table 5.17.

Table 5.17 Comparison of the recognition probabilities

Phoneme	a	b	d	e
Logarithmic recognition probability average ($\overline{recProb}$)	-54.54	-58.06	-60.74	-55.49

These results sustain the above observation and are being used further on to complete step 1. The following measure, called recTrust, is proposed to be used to select the best recognized phones:

$$recTrust = recProb - \overline{recProb} \tag{5.1}$$

where $recProb$ is the current recognition probability for the phone and $\overline{recProb}$ is the average recognition probability for that particular phone.

Clearly, a positive and high *recTrust* means that the phone is recognized with a high probability relative to its average recognition probability and vice-versa, a negative *recTrust* means that the phone is recognized with a lower probability relative to its average recognition probability. Using this measure, the output of step 1 would be a ranking of the best recognized phones within a given test speech audio clip.

It must be noted that the probabilities mentioned above are in fact log probabilities.

5.3.2.2 Step 2 - Using the *recTrust* ranking to select a small vocabulary

The process of selecting a small vocabulary, even when the *recTrust* ranking is available, is not trivial. Using the following notations: *ph1* – the top ranked phone, *ph2* – the second ranked phone and *ph3* – the third ranked phone, the vocabulary can be selected in various ways:

- all its words contain *ph1* and *ph2* and *ph3*
- all its words contain *ph1* and *ph2*
- all its words contain *ph1*
- all its words contain either *ph1* or *ph2*
- all its words contain either *ph1* or *ph2* or *ph3*
- other phones logics

In order to evaluate the different types of vocabulary selection we've experimented with some of them and we've summarized the results in Table 5.18. The average vocabulary size is computed as a weighted average of all the vocabulary sizes. The weights are the number of usages during the recognition process of the words in the testing database. We've measured the correct vocabulary selection rate as the percentage of correctly chosen vocabularies out of the total number of selected vocabularies. A vocabulary is considered to be correctly selected if it contains the word that is actually uttered in the audio speech sample.

Table 5.18 Vocabulary selection comparison

Vocabulary selection logic	Average vocabulary size [no. words]	Correct vocabulary selection rate
ph1 & ph2 & ph3	215	63.70%
ph1 & ph2	n/a	74.14%
ph1	3460	86.62%
ph1 | ph2	5470	97.97%
ph1 when ph1 is one of {a, e, i, i3, l, o, s1, u, z}, ph1 | ph2 otherwise	4756	96.84%
no vocabulary selection logic (original vocabulary is used)	10000	100%

As Table 5.18 shows, the vocabulary size and the correct vocabulary selection rate increase as the selection logic becomes looser. One thing that should be remembered is that the average vocabulary size selected by this method should be compared with the original vocabulary size which is 10000 words. A second important aspect is that the selection rate is only acceptable for the last two vocabulary selection logics, because the recognition rate of the ASR system is up-bounded by the selection rate.

5.3.2.3 Experimental results

As the two main issues raised by the three-step recognition method have been addressed, and the third step of this method is actually isolated words recognition that uses the small vocabulary selected at step 2, the following table summarizes the results (the results presented in Table 5.15 are repeated for comparison).

The results presented in Table 5.19 highlight significant recognition speed improvements (37% for the phones ASR system, 52% for the triphones ASR system and 42% for the mixed-models ASR system) with little word error rate drawbacks. The word error rate was expected to get worse due to the imperfect vocabulary selection, and it certainly does, but only to a small degree. The mixed-models ASR system presented in Section 0 is still the most accurate system and has an acceptable real-time factor (slightly less than 1), even though it performs two successive recognitions.

Table 5.19 Comparison between one-step and three-step recognition methods

HMM models	Language information		Recognition algorithm		WER	RTF
	Vocabulary size	Words sequence restrictions	Type	Steps description		
phones	10000 words	only one word	one step recognition	recognize words	8.47%	0.88
triphones					10.99%	0.71
mixed-models					5.61%	1.65
phones	**Step 1:** 41 phones **Step 3:** average of 4756 words	**Step 1:** any combination of any phones **Step 3:** only one word	three step recognition	**Step 1:** recognize phones **Step 2:** select vocabulary **Step 3:** recognize words	10.64%	0.55
triphones					13.18%	0.34
mixed-models					8.05%	0.95

The real-time factor is lower than one for all these ASR systems. Consequently, we can assert that the real-time recognition goal has been achieved.

5.4 Acoustic Models for Continuous Speech Recognition

Section 5.2 has presented the first speech recognition experiments we have employed. The experiments strictly followed the hierarchical training strategy described in Section 5.1.2, but have only reached the fourth step in the development strategy. The last step envisioned by our strategy required continuous speech data, which was not available at that time.

This section continues with the fifth step of the training strategy: continuous speech training and continuous speech recognition evaluation. The first experiments were made only for a single speaker for which the continuous speech data was available. Going further, this section presents some results which question the effectiveness of the training strategy in the context of continuous speech. Moreover, the last part of this section presents some ASR systems created with a different development toolkit, which implements by default a different training strategy. This toolkit switch is motivated and explained, and in the end some conclusions are drawn.

5.4.1 Speaker-dependent continuous speech recognition experiments

For all the experiments presented in this section we have used three speech databases: the WORDS database, the CS_01 database and the CS_02 database. From these three databases we have selected only the files recorded by speaker 06, the author of this thesis.

The WORDS database is an isolated words database which contains 10000 audio clips comprising of one word each. The CS_01 database is a continuous speech database comprising 1000 phrases and the CS_02 database is a continuous speech database comprising 244 phrases. A more detailed description and analysis of these speech databases were made in Section 4.2.2 and Section 4.2.3.

The WORDS database and the CS_01 database were split into a training part (90% of the files) and an evaluation part (10% of the files). The CS_02 database was only used for evaluation.

All the following experiments employ basic word-loop language models. For the phones recognition experiments the vocabulary consisted of 36 phones. For the words recognition experiments the words vocabulary consisted of 15k words.

As performance figures, phone error rate (PER) is reported for phones recognition, while word error rate (WER) is reported for words recognition.

The hierarchical training strategy was employed to create a continuous speech recognition system for this speaker. The ASR systems obtained at the end of every step in the development strategy are individually denoted and evaluated. The initial acoustic model configuration is the best design configuration found thanks to the experiments in Section 0: 12-state HMMs with 3 GMs per state using MFCCs and their first order derivatives to model speech units.

The isolated words recognition system obtained after the third step (embedded phone HMM training) in the training strategy is denoted ASRS-1 and is evaluated only on the WORDS test set. Continuing with the fourth step (embedded triphone HMM training) in the training strategy, we have developed ASRS-2. This second isolated words recognition system is also evaluated on the WORDS test set. Finally, we have used the CS_01 training set to develop a continuous speech recognition system (as specified by the fifth training step) and denoted it ASRS-3. This continuous speech recognition system is evaluated on the WORDS, CS_01 and CS_02 test sets. All the evaluation results are summarized in Table 5.20.

Table 5.20 Speaker dependent ASR performance evolution

Exp	ASR system	HMM configuration		Evaluation test set	WER
1	ASRS-1	12-state HMMs 3 GMs per state 12 MFCCs + energy + 1st order derivatives	models for phones	WORDS	14.08%
2	ASRS-2		models for triphones	WORDS	18.51%
3	ASRS-3		models for triphones	WORDS	22.69%
4				CS_01	46.72%
5				CS_02	56.98%
6	ASRS-4		models for phones	WORDS	33.30%
7				CS_01	68.27%
8				CS_02	76.14%

The slightly worse results obtained for ASRS-2 compared to the results obtained for ASRS-1 motivated us to try to skip the fourth step in the hierarchical training strategy and create a phone-based continuous speech recognition system. This is how the system ASRS-4 (also evaluated Table 5.20) appeared.

Please note that for speaker 06, as opposed to other speakers, the isolated words recognition system which is based on phones is better than the one which is based on triphones (exp 1 compared to exp 2 in the table). On the contrary, for continuous speech recognition systems triphones are preferred (exp 3, 4, 5 compared to exp 6, 7, 8). Another interesting thing to be noted is that after the fifth step in the training strategy (when ASRS-3 is developed) the word error rate on the WORDS test set deteriorates (exp 2 compared to exp 3). This is because in step 5 the system is trained with continuous speech which is slightly different than isolated spoken words. This behavior was expected to occur, being a downsize of the fact that this new system (ASRS-3) can now recognize continuous speech also.

Analyzing the raw hypothesis text given by the ASR systems we have observed that an important amount of errors appeared due to the fact that long words were split into several smaller words. These smaller words were acoustically-similar to the long words they were wrongly replacing (an obvious language model problem). At this point we had no means of improving the language model, so we have tried a different technique to solve this problem. The next experiment we have done consisted in varying the word insertion penalty – a parameter which controls the word insertion errors by assigning a lower or higher probability penalty to words splitting. For this experiment we have used only ASRS-3. The results are reported in Table 5.21.

Table 5.21 Varying the word insertion penalty

Exp	ASR system	HMM configuration	Evaluation setup		WER
			Test set	WIP	
1	ASRS-3	models for triphones 12-state HMMs 3 GMs per state 12 MFCCs + energy + 1st order derivatives	WORDS	0	22.69%
2			CS_01	0	46.72%
3			CS_02	0	56.98%
4			WORDS	10	18.76%
5			CS_01	10	42.87%
6			CS_02	10	52.38%
7			WORDS	20	17.3%
8			CS_01	20	40.88%
9			CS_02	20	49.54%
10			WORDS	50	14.48%
11			CS_01	50	40.25%
12			CS_02	50	48.32%
13			WORDS	100	51.5%
14			CS_01	100	54.67%
15			CS_02	100	62.95%

The results table outlines the importance of this parameter. The word error rate significantly improves when the insertion penalty value is increased, and reaches an optimum around the value 50. The relative WER improvement is 36% on the WORDS test set, 14% on the CS_01 test set and 15% on the CS_02 test set. As discussed in Section 4.3.5 the word insertion penalty is a language dependent parameter. Some languages usually use shorter words, other longer words. Consequently, this parameter needs to be empirically optimized.

Regardless of the comparison between the four ASR systems presented in Table 5.20 or the word insertion penalty optimization shown in the above table, please note that the word error rate for continuous speech is still quite high. Of course, the fact that a very basic language model is being used for these experiments is also contributing to the bad results, but a 40% – 50% words error rate is just unacceptable.

Given these bad results and the important advices from our coordinator teachers, we have reiterated the acoustic models optimization experiments. The most relevant observations of our coordinators regarded the high number of states for the HMMs. State-of-the-art ASR systems use *5-state* HMMs or *7-state* HMMs while our experiments revealed an optimum recognition accuracy for *12-state* HMMs systems.

Several other speech recognition systems were developed from scratch following the same hierarchical training strategy, but using different HMM configurations (6-state HMMs, 8-state HMMs and 10-state HMMs). The rest of the configuration parameters (number of Gaussian mixtures per state and voice features) were kept fixed. The resulted systems were evaluated on the same test sets: WORDS, CS_01 and CS_02 and their performance, expressed as both WER and PER, is reported in Table 5.22.

Table 5.22 Varying the number of states per HMM for CSR systems

Exp	ASR system	HMM configuration		Evaluation test set	WER	PER
1	ASRS-5	models for triphones 3 GMs per state 12 MFCCs + energy + 1st order derivatives	6-state HMMs	WORDS	28.27%	27.20%
2	ASRS-6		8-state HMMs		25.48%	22.81%
3	ASRS-7		10-state HMMs		20.30%	18.45%
4	ASRS-3		12-state HMMs		22.69%	34.82%
5	ASRS-5		6-state HMMs	CS_01	46.42%	30.96%
6	ASRS-6		8-state HMMs		36.55%	33.48%
7	ASRS-7		10-state HMMs		38.97%	29.72%
8	ASRS-3		12-state HMMs		46.72%	36.77%
9	ASRS-5		6-state HMMs	CS_02	53.39%	36.08%
10	ASRS-6		8-state HMMs		39.99%	34.93%
11	ASRS-7		10-state HMMs		45.32%	32.67%
12	ASRS-3		12-state HMMs		56.98%	41.13%

Table 5.22 presents a large number of experimental results. We first observe that the word error rates on the continuous speech test sets CS_01 and CS_02 are indeed lower for 8-state HMMs system compared to the others. We also observe that the word error rate on the isolated words test set (WORDS) is worse for the systems with a small number of HMM states. *Consequently, the optimizations employed for isolated words are verified, but the optimal HMM configuration obtained for isolated words is not optimal for continuous speech*. Phones error rates exhibit the same general trend as the words error rates, with a few exceptions (exp 6 and exp 10). The phones error rates for the continuous speech recognition systems have acceptable values and, given the fact that no language models were used in these experiments, these results confirm that the acoustic models have a decent accuracy.

The conclusion we reached after these experiments (the optimal HMM configuration obtained for isolated words is *not* optimal for continuous speech) was shaking the whole hierarchical training strategy we had used so far. The fact that the acoustic models' optimizations needed to be carried out separately for continuous speech implied a great amount of work and efforts. However, at this point our research benefited again from the good advice of one of the coordinators. The suggestion of using the CMU Sphinx toolkit to develop a state-of-the-art acoustic model proved to be outstanding.

5.4.2 CMU Sphinx-based continuous speech recognition experiments

The CMU Sphinx speech recognition toolkit helped a lot the development of the LV-CSR. The most important advantage proved to be the state-of-the-art training strategy which is incorporated by default in this toolkit. Moreover, the state-of-the-art parameter sharing technique (state-dependent output distributions across different phonetic models called senones) is used by default in CMU Sphinx, improving the trainability characteristic of context-dependent models.

The second remarkable advantage proved to be the tool's simple usability. The development of a decent continuous speech recognition system using CMU Sphinx requires only two things: the appropriate phonetic, speech and text resources and the knowledge regarding how to use the toolkit. If you have the know-how, then the usage of this toolkit becomes straightforward, almost trivial. For example, the whole process of training a continuous speech recognition system consists of configuring a list of parameters (which could also be left with the default values) and then running a single script which deals with the various (default) training steps: training context-independent models, training context-dependent models, building decision trees for models similarities, tying the models' states, training context-dependent models (second iteration).

Besides the fact that CMU Sphinx is very easy to use compared to HTK, it is also intensively supported on the online forum. Questions regarding usage, errors, training algorithms and other techniques are answered in a few hours, facilitating and supporting the development of new ASR systems. Therefore, a vast number of commercial and non-commercial ASR systems are developed today using CMU Sphinx and are incorporated in all sorts of applications such as automatic subtitling, voice web browsers, voice-controlled keyboards, command-and-control applications, games that help practice spoken English (point-and-say), etc.

Using the same experimental setup as for the experiments presented in Section 5.4.1 we have created a Sphinx-based speaker-dependent ASR system. The default training strategy and parameter values were used for this first experiment. The results are summarized and compared to the best ones obtained using HTK in Table 5.23.

Table 5.23 The first CMU Sphinx-based ASR system

Exp	ASR system	System configuration	Evaluation test set	WER
1	ASRS-6	HTK-based system 12-state HMMs (total of ~30000 states) 3 GMs per state models for context-dependent phones (triphones) 12 MFCCs + energy + 1st order derivatives	WORDS	25.48%
2			CS_01	36.55%
3			CS_02	39.99%
4	ASRS-8	CMU Sphinx-based system 5-state HMMs (total of 1000 states called senones) 8 GMs per state models for context-dependent phones (triphones) 12 MFCCs + energy	WORDS	3.3%
5			CS_01	31.4%
6			CS_02	33.8%

The results in Table 5.23 show that the CMU Sphinx-based system is much better at recognizing isolated words than the HTK-based system. The continuous speech recognition results are also better: relative word error rates improvements of 13.7% for the CS_01 test set and 15.5% for the CS_02 test set. Although we admit that the HTK-based system might be further improved and tuned to reach the performance figures of the Sphinx-based system, the usage easiness of the latter made it our preferred choice for all further experiments.

The next speech recognition systems developed with the Sphinx toolkit were *speaker-independent systems*. At this time the WORDS database, the CS_01 database and the CS_02 database were complete and, consequently, we were able to use speech data for 17 different speakers to train the speaker independent ASR system. The total training speech data summed up to about 54 hours of speech and the testing speech data had about 7 hours (4 hours of isolated words and 3 hours of continuous speech). The basic word-loop language model with the 15k words vocabulary is still used for the next experiments.

CMU Sphinx suggestions about the right number of senones and Gaussian mixtures per senones are not very specific. They strongly suggest tuning these parameters on every specific speech database. Following this suggestion, the first speaker-independent ASR experiments we made aimed at finding the best values for these two parameters. We started with 1000 senones and 8 GMs, just as in the speaker-dependent case, and continued up to 6000 senones with 32 GMs. Some of the experimental results are presented in Table 5.24.

Table 5.24 Varying the number of senones and Gaussian mixtures for CMU Sphinx-based CSR systems

Exp	ASR system	System configuration		Evaluation test set	WER
1	ASRS-9	CMU Sphinx-based system 5-state HMMs context-dependent models 12 MFCCs + energy	1000 senones with 8 GMs	WORDS	10.2%
2	ASRS-10		2000 senones with 16 GMs		8.9%
3	ASRS-11		4000 senones with 16 GMs		9.7%
4	ASRS-12		5000 senones with 16 GMs		10.4%
5	ASRS-9		1000 senones with 8 GMs	CS_01	49.5%
6	ASRS-10		2000 senones with 16 GMs		41.4%
7	ASRS-11		4000 senones with 16 GMs		37.3%
8	ASRS-12		5000 senones with 16 GMs		36.8%
9	ASRS-9		1000 senones with 8 GMs	CS_02	53.0%
10	ASRS-10		2000 senones with 16 GMs		45.2%
11	ASRS-11		4000 senones with 16 GMs		42.4%
12	ASRS-12		5000 senones with 16 GMs		43.1%

The results in Table 5.24 show that great improvements in continuous speech performance figures can be obtained if the right parameters are chosen. After analyzing the results in this table and all the other results for the tuning experiments, we have decided that, for our training speech material, the most suitable configuration is 4000 senones with 16 Gaussian mixtures. Table 5.25 reports the results obtained individually, for every speaker, for the continuous speech recognition system which uses 4000 senones with 16 Gaussian mixtures per senones (ASRS-11).

Table 5.25 shows that, even if the speech recognition system was trained with data from 17 different speakers, the system is not suitable to recognize any voice. The best recognized voice for the isolated words test set is that of speaker 12, with a word error rate as low as 2.2%. Still, for speaker 17, the word error rate is huge: 18.9%. For this speaker we have bad results for continuous speech also: a word error rate of 51.8% on the CS_01 test set. We observe that there are speakers which are recognized very well, such as speaker 12, speaker 06 and speaker 02 and speakers which have poor recognition rates, such as speaker 01 and speaker 17. The conclusion we can draw here is that the human voice variability is a very important factor that has to be taken into account for a speaker-independent system. The development of such a system requires data from more than 17 speakers.

Table 5.25 Per speaker results for ASRS-11

WER [%]		Speaker															
		01	02	03	04	05	06	09	10	12	15	16	17	18	19	20	all
Test set	WORDS	8.5	7.5	7.5	8.9	12.1	3.7	9.8	7.6	2.2	26.1	10.0	18.9	5.5	7.0	9.8	9.7
	CS_01	57.1	28.2	36.3	29.4		28.6		36.0			31.7	51.8	32.6	39.1	39.8	37.3
	CS_02	67.4					32.6									27.2	42.4

One other important conclusion regards the average word error rate for continuous speech: 37.3% for the CS_01 test set and 42.4% for the CS_02 test set. Given that for this experiment the language model is deficient (a 15k words word-loop language model has been used), the results are, in our opinion, very good.

5.5 SUMMARY

This chapter has presented a large set of experiments carried out with the purpose of creating and optimizing an acoustic model for continuous speech recognition. The chapter began with some introductory issues related to *the design of the acoustic models* and *a hierarchical training strategy* for creating better and better models.

The second section of this chapter was intended to follow the training strategy and develop isolated words recognition systems. A large set of experiments were carried out to optimize and tune the various parameters of the acoustic model and draw some conclusions regarding the best design setup. Unfortunately, the optimization experiments were not conducted directly on continuous speech and, as it was later shown, *the best design for isolated words recognition is not the best for continuous speech recognition*. Given this, it would have taken a huge amount of time to optimize for continuous speech.

The second section also presented two innovative ideas: *the mixed-models ASR system* and *the three-step isolated words recognition method*. The mixed-models ASR system uses both context-dependent and context-independent phone models and, although it is slower, it has a better performance in terms of word error rate than phones-based or triphones-based systems. The three-step isolated words recognition method is a time-optimized version of the usual, single step words recognition method. Its purpose is to speed up the system and eventually to create a real-time words recognition system. These two are direct contributions of the author of this thesis, as opposed to the hierarchical training strategy which is a contribution of the whole research team.

The chapter ends with some continuous speech recognition experiments. The best results are obtained using a new speech recognition toolkit: CMU Sphinx. A short motivation on why this new toolkit is used, why it helps and facilitates the development of ASR systems is given and afterwards the first continuous speech recognition results are presented. The last section ends with some important conclusions regarding the continuous speech results: a) the average results are good enough, considering that a basic word-loop language model was used for the experiments and b) even though the system was trained with speech data from 17 speakers, we observe really poor results for some types of voices. The language model issue will be addressed and solved in the following chapter, but the pseudo-speaker-independency issue can only be solved if more speech data will be acquired from various speakers.

CHAPTER 6

LANGUAGE MODELS CONSTRUCTION AND ADAPTATION TECHNIQUES

The previous chapter has ended with some encouraging speech recognition results. It showed that in terms of isolated words recognition we were able to obtain very good results. For the best recognized speaker we reached a word error rate as low as 2.2%, while the average word error rate over all speakers was 9.7%. Still, for continuous speech recognition the results were not that good: an average word error rate of 37.3% on one test set and 42.4% on another one.

One important cause for the huge difference between isolated words recognition and continuous speech recognition was the language model. In the first case (isolated words recognition) the basic word-loop language model with only 15k words vocabulary worked fine. However, in the second case, the larger vocabulary (45k words) and the lack of language restriction caused a lot of word substitutions and insertions. It is obvious that in the case of isolated words recognition a more sophisticated language model will not help. The reason: the recorded words do not follow any language or statistical rules; they are just randomly chosen words. On the contrary, for continuous speech recognition the language model has a very important role. Continuously spoken words follow some grammar, linguistic rules and, if implemented, these could help a lot.

This chapter presents the steps we took into developing other types of language models in our attempt to obtain a model suitable for continuous speech. We first describe a failed attempt to create a general finite-state-grammar for Romanian and then the success we had with state-of-the-art language models: n-grams. We have created several n-gram language models and used them in various large-vocabulary continuous speech recognition systems with quite satisfactory performance. Their evaluation is also presented in this section.

The last part of this chapter focuses on several methodologies we have developed to optimize the amount of time needed to create a domain-specific ASR system for a low-resourced language, such as Romanian. We use several statistical machine translation (SMT) methods to create Romanian text corpora out of a French text and we show that this kind of corpus can be successfully used for language modeling. We focus on optimizing the translation methodologies to obtain more adequate Romanian texts and language models (constructed based on these texts) and consequently more accurate Romanian domain-specific ASR systems.

6.1 CONSTRUCTING CONTINUOUS SPEECH LANGUAGE MODELS

After the acoustic model for continuous speech was created, the main goal of our work was to develop a language model suitable for Romanian continuous speech. The process of constructing this language model was, in part, affected by the lack of resources. We first tried to create a large finite-state-grammar (FSG) for Romanian. The most important element in this decision was the lack of large text corpora, which are absolutely necessary for state-of-the-art n-gram language modeling.

6.1.1 FSG language models

A finite state grammar or word network grammar is a graph in which the nodes are words and the directed links represent allowable word transitions. A finite state grammar explicitly specifies all the allowable sequences of words for a given task. If the task is relatively small (digits recognition, phone dial, etc.) than this type of language model can be successfully used. Moreover, finite state grammars can be successfully used in word spotting applications. The word graph defined by a finite state grammar is used by the speech recognizer as a starting point in creating the search graph discussed in Section 5.3.1. In fact, the search graph is an extended version of the word graph (the latter does not include phonetic transcriptions of words).

6.1.1.1 HTK implementation of FSGs

The development of a rough Romanian finite state grammar was also determined by the fact that, at that particular time, we were using HTK, which makes extensive use of this type of language models. In fact, HTK hardy supports other types of language models. Bigram models are also implemented using probabilistic finite state grammars, while trigram models are not supported by the default distribution.

Table 6.1 HParse format grammar meta-characters

Character	Significance
|	denotes alternatives
[ ]	encloses options
{ }	denotes zero or more repetitions
< >	denotes one or more repetitions
<< >>	denotes context-sensitive loop
$	denotes a variable

As finite state grammars are concerned, HTK provides two ways of creating them, two formats which can be used to define a word network. The first one, which is simpler, but less flexible is called HParse format. The HParse format grammar consists of an extended form of regular expression enclosed within parentheses. Expressions are constructed from sequences of words and the meta-characters presented in Table 6.1.

To better illustrate the use of all these meta-characters and the construction of a finite state grammar using the HParse format we will take a simple example. Let us consider the task of creating a telephone number recognizer. For this task the user would be allowed to utter as many digits as he wants, depending on the length of the number, but no other words. The word graph defining all words and allowable word transitions is illustrated in Figure 6.1.

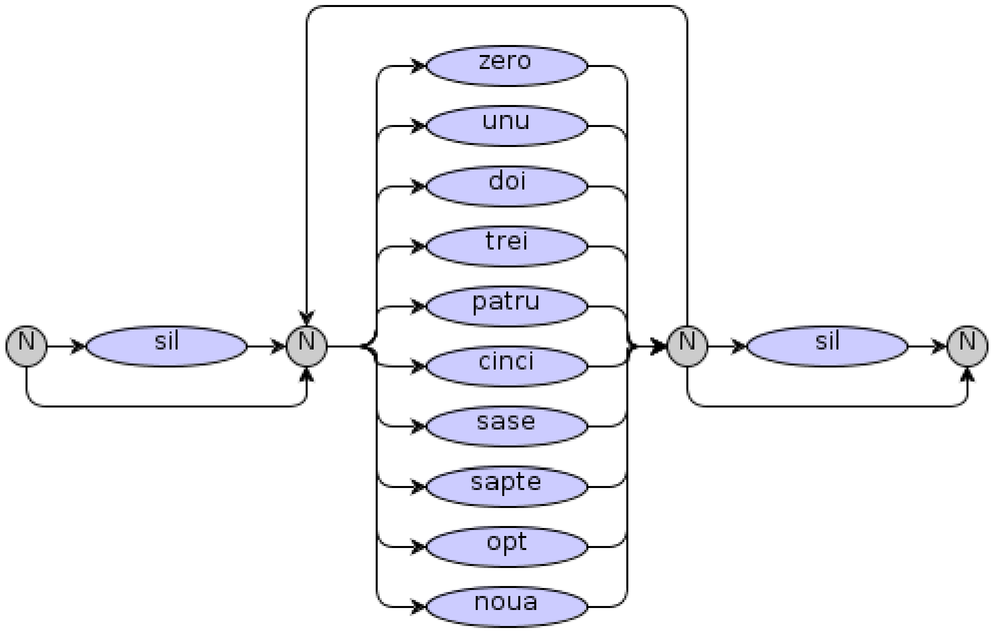


Figure 6.1 The telephone number word graph

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 or as word-loop markers. For example, the list of digits in the center of the figure can be regarded as a word-loop (any combination of any words is equally probable). The nodes marked as *sil* stand for silence zones in the audio data. These zones may appear or not, depending on how the audio file was end-pointed, thus the skip transitions are required. The equivalent HParse format grammar for this example is illustrated in Table 6.2.

Table 6.2 The telephone number HParse format grammar

```
$digit = unu | doi | trei | patru | cinci |
         şase | şapte | opt | nouă | zero;

( [sil] < $digit > [sil] )
```

As Table 6.2 shows, the HParse format grammar first defines a variable called *digit*. This variable can have any of these values: *zero* (zero), *unu* (one), *doi* (two), ..., *nouă* (nine). The second line in the definition explicitly describes the grammar between the round brackets: an optional starting *sil*, followed by one or more repetitions of a digit, followed by an optional ending *sil*.

HTK parsed this type of HParse format grammars at run-time up to version 3.4. In newer versions these grammars are compiled offline into a different, more machine-oriented format:

the so called Standard Lattice Format (SLF). SLF files are used for a variety of functions some of which lie beyond the scope of the standard HTK package. The description here is limited to those features of SLF which are required to describe word networks suitable for input to HTK.

A word network in SLF consists of a list of nodes and a list of arcs. The nodes represent words and the arcs represent the transition between words. Each node and arc definition is written on a single line and consists of a number of fields. Each field specification consists of a “name=value” pair. Field names can be any length but all commonly used field names consist of a single letter. By convention, field names starting with a capital letter are mandatory whereas field names starting with a lower-case letter are optional. Any line beginning with a grill sign is a comment and is ignored.

For an easier illustration, let us take the same telephone number example presented as a word graph in Figure 6.1. The equivalent SLF grammar is illustrated in Table 6.3. As observed in this table, the SLF grammar fully lists all the components in the word graph. For every node, the node id and the associated word is listed, while for every link (or transition), the link index, the start node and the end node are listed. By default, all transitions are equally likely. However, an optional field can be used to assign a log transition probability to any link.

6.1.1.2 The LMGraphX tool

The standard lattice format allowed by HTK is a flexible format which can be used to create finite state grammars for various applications. Our goal was to create a general Romanian language model. For this task we needed extensive knowledge of Romanian grammar. As a compromise, we started this task by implementing just a few phrase structures. We used a top-down approach defining at first the way several types of sentences are combined to form phrases. Going further we defined several types of sentences and their constitutive parts: subject, predicate, attribute, complement, etc. All these sentence constituents were further described. The various ways in which they can be formed were defined using combinations of parts of speech, which are eventually lists of words in the lexicon.

Table 6.3 The telephone number SLF grammar

```
# Define size of network: N=num nodes and L=num arcs
N=16 L=27
# List nodes: I=node-number, W=word
I=0 W=!NULL
I=1 W=sil
I=2 W=!NULL
I=3 W=zero
I=4 W=unu
I=5 W=doi
I=6 W=trei
I=7 W=patru
I=8 W=cinci
I=9 W=sase
I=10 W=sapte
I=11 W=opt
I=12 W=noua
I=13 W=!NULL
I=14 W=sil
I=15 W=!NULL
# List arcs: J=arc-number, S=start-node, E=end-node
J=0 S=0 E=1
J=1 S=0 E=2
J=2 S=1 E=2
J=3 S=2 E=3
…
…
J=4 S=2 E=4
J=5 S=2 E=5
J=6 S=2 E=6
J=7 S=2 E=7
J=8 S=2 E=8
J=9 S=2 E=9
J=10 S=2 E=10
J=11 S=2 E=11
J=12 S=2 E=12
J=13 S=3 E=13
J=14 S=4 E=13
J=15 S=5 E=13
J=16 S=6 E=13
J=17 S=7 E=13
J=18 S=8 E=13
J=19 S=9 E=13
J=20 S=10 E=13
J=21 S=11 E=13
J=22 S=12 E=13
J=23 S=13 E=2
J=24 S=13 E=14
J=25 S=13 E=15
J=26 S=14 E=15
.
```

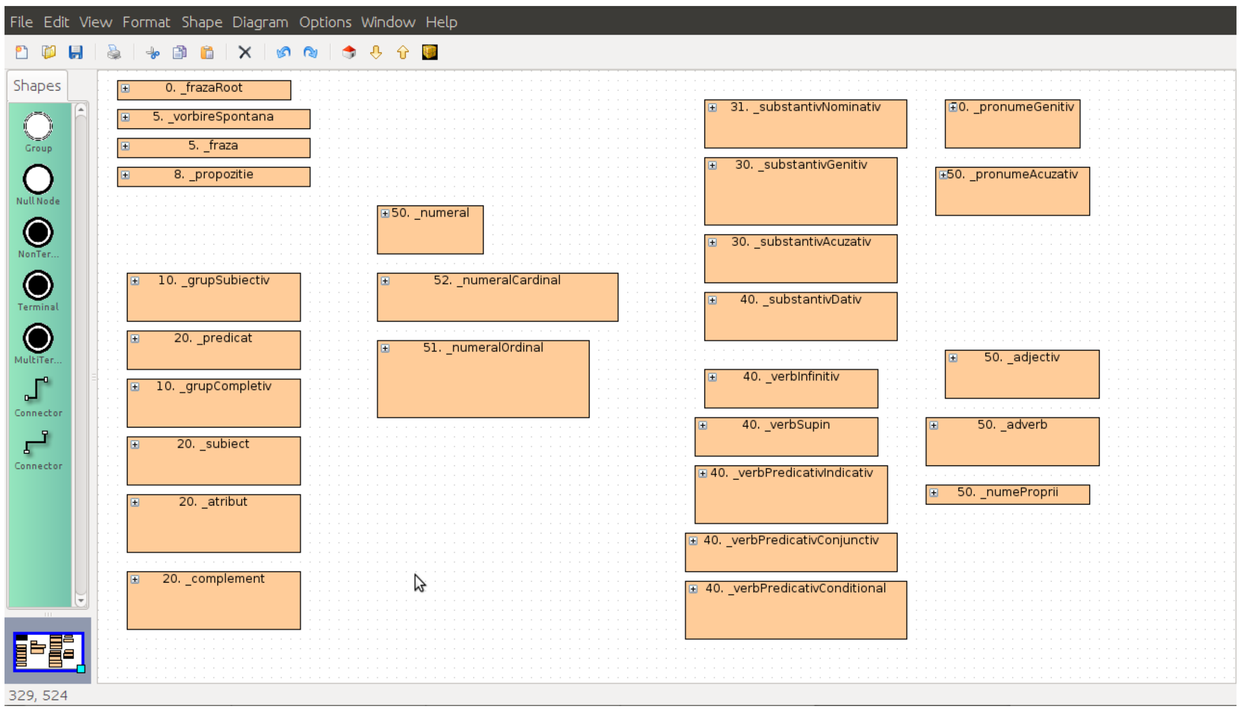


Figure 6.2 The LMGraphX tool and the top level of the Romanian grammar graph

In order to create such a complex finite state grammar we needed a user-friendly tool. Of course, all these rules can be written in plain SLF, but the process is tedious and extremely inefficient. Considering this, we developed a graphical user interface (GUI) Java application to serve the purpose of creating SLF grammars. The application, called LMGraphX, can run on any operating system and is very simple to use. It allows the user to create special graphs, based on the JGraph Java API [JGraph]. The GUI and the general Romanian language model created using this application is illustrated in Figure 6.2. The figure does not fully describe the entire FSG language model. It only displays the top view of the grammar (only non-terminals).

The LMGraphX has the final goal of creating a SLF file to be further used in HTK. To achieve this task, the graphical tool allows the user to create several types of nodes (non-terminals, terminals and multi-terminals) and the appropriate links. Non-terminal nodes are themselfes graphs of nodes which will be furthermore extended. Terminal nodes represent a single word, while multi-terminal nodes stand for a group of equally probable words.

The phrases are regarded as non-terminal blocks and are composed of different types of sentences. The sentences are also non-terminals, being composed of a subject (optional), attributes (also optional), a predicate (this is compulsory) and one or several complements (these are also optional). A simple example of a sentence structure is presented in Figure 6.3. These parts that compose a sentence are also non-terminals and are furthermore extended into parts of speech until simple terminals (or multi-terminals) are reached. The terminals and multi-terminals are words or groups of words that are part of the lexicon. An example of a multi-terminal block is presented in Figure 6.4.

Figure 6.3 illustrates a sentence non-terminal named *_sentenceType1* composed of several other non-terminals named *_subject*, *_attribute*, *_predicate* and *_complement*. The four non-terminals will have to be defined furthermore until terminals or multi-terminals are reached. Figure 6.4 describes the multi-terminal named *_preposition*. This multi-teminal is composed of several terminals, namely the words: *pe*, *către* and *de la*. The example is obviously not complete, but its purpose is to demonstrate that this graph decomposition ends at some point with the lexicon words introduced as terminals.

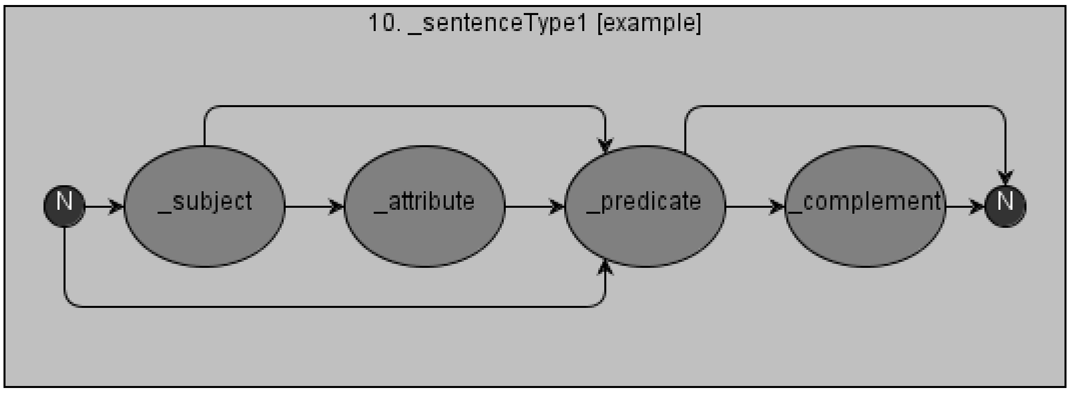


Figure 6.3 An example of a sentence composed of non-terminals

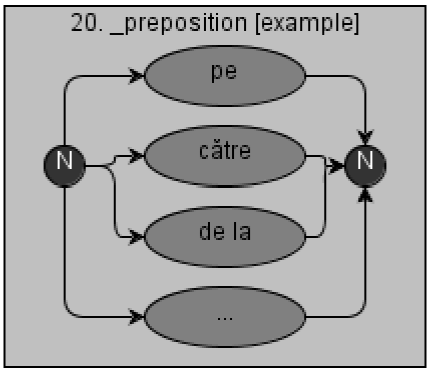


Figure 6.4 An example of a multi-terminal part of speech

The task of implementing a part of the Romanian grammar, even with the help of this newly developed tool, was itself very time-consuming and required a lot of resources. Several people worked on implementing the grammar rules (which was the easier part) and several others on creating multi-terminal lists. The model was desired to be as general as possible and consequently a large amount of Romanian words were introduced in this grammar.

6.1.1.3 FSG speech recognition experiments

The ASR system which employed the Romanian finite state grammar was a complete failure. The word error rate was worse than that of an ASR system which used a word-loop language model. The recognition process was ten times slower when compared to the same word-loop speech recognition process. Unfortunately, only at this point we started to analyze the various characteristics of the FSG language model.

Its size and complexity were responsible for the huge amount of time needed to do speech recognition. The complexity was caused by the way we designed the grammar. The multi-terminal nodes which comprised hundreds of words were used within the word graph several times. Consequently, many individual words appeared multiple times in the FSG increasing the number of nodes. The search graph constructed by the ASR system based on this grammar was implicitly also very large. For comparison, for a word-loop language model every word appears once in the search graph, while in an n-gram language model every word appears *n-1* times in the search graph (for example for a trigram language model every word appears twice).

The slightly similar word error rate compared to a word-loop is easily explained by the loose constrains imposed by the Romanian FSG. Even if we have implemented numerous grammar rules, the language is still very flexible allowing various sentence structures. The combination of rules and flexibility turned out to be very difficult to implement. We opted for flexibility, but in the end this forced the grammar to allow a lot of unusual-structured sentences. Moreover, the FSG was unable to implement word-sense rules which are very important in continuous speech. Consequently, the grammar allowed non-sense sentences such as *trenul doarme galben* (the train sleeps yellow). This is a syntactically and morphologically correct sentence, but it's a non-sense assert. Nobody will ever use these words in this particular sequence.

In conclusion, the finite state grammar attempt to build a Romanian language model was a failure and we abandoned the idea.

6.1.2 N-gram language models

The n-gram paradigm and the theoretical details about the construction of n-gram language models were given in Section 2.2. As mentioned in that section, a large amount of textual data is required to create a robust, domain-specific n-gram language model. Moreover, if the language model needs to be general (suitable for common phrasal structures used in various domains) the text corpora need to be even larger.

In the first experiment we made, we used all the textual transcriptions of the audio files in the continuous speech databases CS_01 and CS_02 to create a basic bi-gram language model. The total number of phrases was 1250. Of course, this experiment is not very conclusive because the phrases in the test set are used in the language model development stage also. Nevertheless, we did not intend the experiment to be conclusive, but only to get an idea of the importance of an n-gram language model, when compared to the previously used word-loop language model. Note that at this time we did not have any text corpora and we needed a good motivation to invest time in acquiring them. The results are reported in Table 6.4.

Table 6.4 Basic bigram continuous speech recognition experiment

Exp	ASR system	System configuration		Evaluation test set	WER
1	ASRS-3	HTK-based system 12-state HMMs 3 GMs per state models for triphones 12 MFCCs + energy + 1st order derivatives	word-loop language model	CS_01	46.72%
2				CS_02	56.98%
3	ASRS-13		bigram language model	CS_01	36.93%
4				CS_02	48.87%

The experiments presented in Table 6.4 were decisive. At this point the acquisition of the text corpora described in Section 4.3.2 began.

After the corpora were acquired and processed (cleaned), as described in Section 4.3.2 and 4.3.3, they were used to create trigram language models. As a reminder, the three general Romanian text corpora are: a) the small europarl corpus (5.3 million words), b) the medium sized 9am corpus (63 million words) and c) the medium sized hotnews corpus (100 million words). Using the SRI-LM toolkit, we have created state-of-the-art trigram language models (employing the Good-Turing smoothing technique) with these three corpora. The number of unigrams for these language models was limited to 64k words, because the Sphinx decoder cannot cope with larger language models.

A fourth language model was created using a corpus obtained by concatenating the three corpora. Before using them for continuous speech recognition, these four language models were evaluated in terms of perplexity (PPL) and out-of-vocabulary (OOV) rate on the textual transcriptions of the audio files in the continuous speech databases CS_01 and CS_02. The evaluation results are summarized in Table 6.5.

Table 6.5 Language model evaluation

Exp	Language model built with corpus	Evaluation test set	PPL	OOV
1	europarl	CS_01	672.3	3.7%
2		CS_02	812.7	7.1%
3	9am	CS_01	226.3	1.7%
4		CS_02	352.1	2.4%
5	hotnews	CS_01	201.0	1.9%
6		CS_02	288.0	2.5%
7	europarl + 9am + hotnews (all)	CS_01	183.2	1.8%
8		CS_02	285.6	2.4%

Table 6.5 presents some interesting results. As it was expected, the language model created with the smallest corpus (europarl) is the weakest, while the language model created with all the three corpora is the best. In fact, the whole table shows that the performance of the language model increases with the size of the training corpus. This assertion is based on the first performance figure reported (the perplexity). For the OOV rate there are some exceptions: the language model built with the hotnews corpus has a higher (worse) OOV rate than the one built with the 9am corpus on both the two evaluation sets. This means that the 9am corpus contains some key words which do not appear in the hotnews corpus (and which are found within the evaluation phrases).

Another interesting observation regards the performance figures obtained on the CS_01 evaluation set compared to the ones obtained on the CS_02 evaluation set. It is obvious that, regardless of the language model, the results obtained on CS_01 are better. This is a normal thing because the CS_01 database contains news and interviews, while the CS_02 database contains library dialogues. The language models, which are created out of news and parliament discussions corpora, were expected to be more suitable for the CS_01 phrases.

These four language models were also evaluated in the context of ASR. In fact, their initial purpose was to serve as general language models inside continuous recognition systems.

The experimental setup for the following experiments is the one presented in Section 5.4.2 for the Sphinx-based ASR systems optimizations. In fact, for these experiments we use the best acoustic model presented in that section, namely AM-11, together with a word-loop language model (exp 1 and 2) and the previously created n-gram language models (exp 3 to 10). The experimental results are presented in Table 6.6.

Table 6.6 The first large-vocabulary continuous speech recognition results for Romanian

Exp	Acoustic model	Language model	Evaluation test set	WER
1	AM-11	word-loop language model	CS_01	37.3%
2			CS_02	42.4%
3		europarl	CS_01	25.8%
4			CS_02	30.1%
5		9am	CS_01	20.2%
6			CS_02	20.4%
7		hotnews	CS_01	19.3%
8			CS_02	19.9%
9		europarl + 9am + hotnews	CS_01	19.0%
10			CS_02	19.4%

The first and most important thing we need to note about Table 6.6 is that it reports *the first large-vocabulary continuous speech recognition results for Romanian*. As mentioned in Section 1.3, up until now all papers regarding Romanian ASR systems reported small-vocabulary results, while the ASR systems evaluated in these experiments use *45k words vocabularies*.

Secondly, the results reported in this table underline the importance of a robust n-gram language model. Note that the word error rates for the ASR system that uses a word-loop language model (exp 1 and 2) are two times larger (two times worse) than the word error rates obtained by the best ASR system (exp 9 and 10).

Going further with the analysis, we observe that the word error rates reported for the four n-gram language models are correlated with the perplexities and the OOV rates presented in Table 6.5. In conclusion, the best ASR system (exp 9 and 10) is the one which makes use of the best language model, which is the one created with all the existing corpora.

Finally, we observe again that the results obtained on the CS_01 evaluation set are systematically better than those obtained on the CS_02 evaluation set. An improvement of the latter would probably be possible if a domain-specific language model would be used.

Table 6.7 reports the results obtained individually, for every speaker, for the continuous speech recognition system which uses the best n-gram language model (the one created using all the corpora).

Table 6.7 Per speaker results for AM-11 using the best language model

WER [%]		**Speaker**											
		01	02	03	04	06	10	16	17	18	19	20	all
Test set	CS_01	31.1	20.4	19.6	15.9	13.0	14.6	15.0	31.8	15.5	17.1	15.4	19.0
	CS_02	40.8				12.1	15.8					8.9	19.4

Table 6.7 shows again that, even if the speech recognition system was trained with data from 17 different speakers, the system is not suitable to recognize any voice. We easily observe that for the CS_02 evaluation set, for example, there is one poorly recognized speaker (01) and three very well recognized speakers (06, 10 and 20). The difference in terms of word error rate is huge. The 40.8% WER obtained for speaker 01 is totally unacceptable, while the 8.9% WER obtained for speaker 20 represents an outstanding result. As a language model improvement would have no effect over the WER for the different speakers (they all utter the same phrases), the conclusion is that we need to improve furthermore the acoustic model. To compensate for the lack of speaker-independency we need to acquire more speech materials for other various voices, preferably similar to speaker's 01 and 17, for which we obtained the worse results.

6.2 LANGUAGE MODEL ADAPTATION USING MACHINE TRANSLATED TEXT

In the previous section we saw that the language model is a key component in a large-vocabulary ASR system. The gain in performance, in terms of relative word error rate, can be as high as 50%. Moreover, we observed that a general n-gram language model can be used for general ASR tasks (the CS_01 evaluation set), but also for domain-specific tasks (the CS_02 evaluation set – library dialogues). Nevertheless, we assume that domain-specific language models will be more suitable for domain-specific ASR tasks. In fact, a 20% word error rate is a good performance only for a general, large-vocabulary ASR task, while for a domain-specific task, for which the vocabulary might contain only several thousands words, we expect to obtain much better results.

It is natural to expect better results for a simpler task (smaller vocabulary, simpler phrase structures, etc.), but these better results can be obtained only with a price: the cost of developing a domain-specific ASR system or adapting a general ASR system to the domain. In general, when the ASR task changes, the pronunciation system, the dialect (or even language), the speakers average age, etc. might change and we might be needed to adapt the acoustic model as well. However, when switching from a general task to a domain-specific task, the problem of ASR system adaptation is only a problem of language model adaptation.

Even if it is simpler than acoustic model adaptation, language model adaptation still requires a large amount of domain-specific textual data. The size of the needed text corpus is dependent on the complexity of the task. For example, for forecast news speech recognition we would need less data than for scientific discourse speech recognition. For some domains, specific texts can be easily acquired using the WaC (Web as Corpus) approach, but for others the acquisition represents a real problem. Moreover, the domain-specific texts acquisition is more complicated than the one for general texts because now we should be very picky about the data (we do not want out-of-domain data to be acquired and processed as being in-domain data).

The acquisition of domain-specific text corpora can be successfully avoided if the data is available in a foreign language. Using a decent statistical machine translation (SMT) system we could port the foreign text corpora to our target language (Romanian). We have performed several studies to find out if the machine translated text is useful for language modeling. In the end, we have designed an unsupervised and two semi-supervised SMT-based methods for porting a domain-specific corpus from one language to another, with the final goal of creating a domain-specific ASR system for the target language. Our experiments were based on a French tourism-specific corpus which was ported to Romanian.

6.2.1 The SMT-based language model adaptation methodology

We should emphasize the fact that the aim of this adaptation methodology is to create a domain-specific speech recognition system for Romanian. To achieve this goal we need an acoustic model and a domain-specific language model for Romanian, just as illustrated in Figure 6.5.

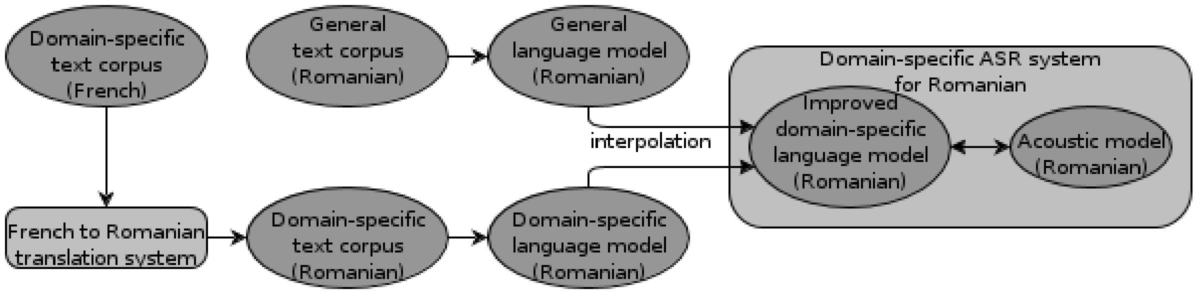

Domain-specific text corpus (French)
General text corpus (Romanian)
General language model (Romanian)
Domain-specific ASR system for Romanian
Improved domain-specific language model (Romanian)
Acoustic model (Romanian)
interpolation
French to Romanian translation system
Domain-specific text corpus (Romanian)
Domain-specific language model (Romanian)

Figure 6.5 The general translation-based ASR domain adaptation methodology

Figure 6.5 depicts the general methodology we propose to create the domain-specific language model for Romanian. Basically, a French-to-Romanian translation system is required to translate the French domain-specific corpus to Romanian. The translation system can be a human expert performing manual translation, an unsupervised machine translation system or a combination of the two.

The language model trained using only the machine translated corpus can be utilized for speech recognition as well, but, as Figure 6.5 illustrates, we expect better results if this domain-specific language model is interpolated with a general language model for the target language. Intuitively, the domain-specific language model will include many specific words and sequences of words, but it will probably have a poor coverage over the general language structures. On the other hand, the most frequent language structures are usually well modeled by a broader, out-of-domain language model, created using larger Romanian corpora. The interpolation of the two language models should lead to an improved domain-specific language model for Romanian.

The most straightforward language model interpolation is at the model level, which entails the estimation of $p_{spcc}(w_1,\ldots,w_N)$, the language model probability derived from the domain-specific LM for the word sequence under consideration. Of course, because of the limited amount of data involved, the domain-specific LM tends to be poorly trained, most of the time resulting in a rather inaccurate estimate. Nevertheless, the domain-specific language model outperforms the initial estimate $p_{gen}(w_1,\ldots,w_N)$ obtained from the general language model when it comes to certain specific word sequences, particularly frequent in the domain-specific task. This provides the justification for merging the two estimates, in order to take advantage of the new information as appropriate.

The simplest way to do so is via linear interpolation. Given the estimates for word w_q and the history h_q denoted by $p_{spec}(w_q|h_q)$ and $p_{gen}(w_q|h_q)$, the interpolated probability estimate can be computed as follows:

$$p_{interp}(w_q \mid h_q) = \lambda \times p_{spec}(w_q \mid h_q) + (1-\lambda) \times p_{gen}(w_q \mid h_q) \qquad (6.1)$$

where $0 \le \lambda \le 1$ serves as the interpolation coefficient. This parameter may be a function of the word history h_q, or, for better performance, some equivalence class thereof. It is typically empirically estimated on held-out data from (a subset of) the domain-specific corpus.

Several adaptation methods arise given the flexibility that the French-to-Romanian translation system block in Figure 6.5 offers. One unsupervised, fully-automatic method and two semi-supervised methods are presented in the next subsections.

6.2.1.1 Unsupervised SMT-based Adaptation Method

The first domain adaptation method translates the in-domain French corpus using the online Google (French-to-Romanian) MT system. This translation is utilized to create the domain-specific language model without any human intervention (in an unsupervised fashion), therefore the machine generated corpus contains several translation errors. Nevertheless, as we will show in the results section, the domain-specific language model created using this unsupervised method is much more suitable for the domain-specific ASR task than a general language model.

6.2.1.2 Semi-supervised SMT-based Adaptation Methods

For the unsupervised adaptation method described above we have used the imperfect Google MT system and obtained some speech recognition results. These results are better than the ones obtained using a general (out-of-domain) ASR system, which does not use any domain-specific information. However, the ideal (performance-oriented) scenario would imply having the French corpus manually translated to Romanian by a human expert. The question that arises is: how much better would have been the results in this *ideal* scenario (expert translation)?

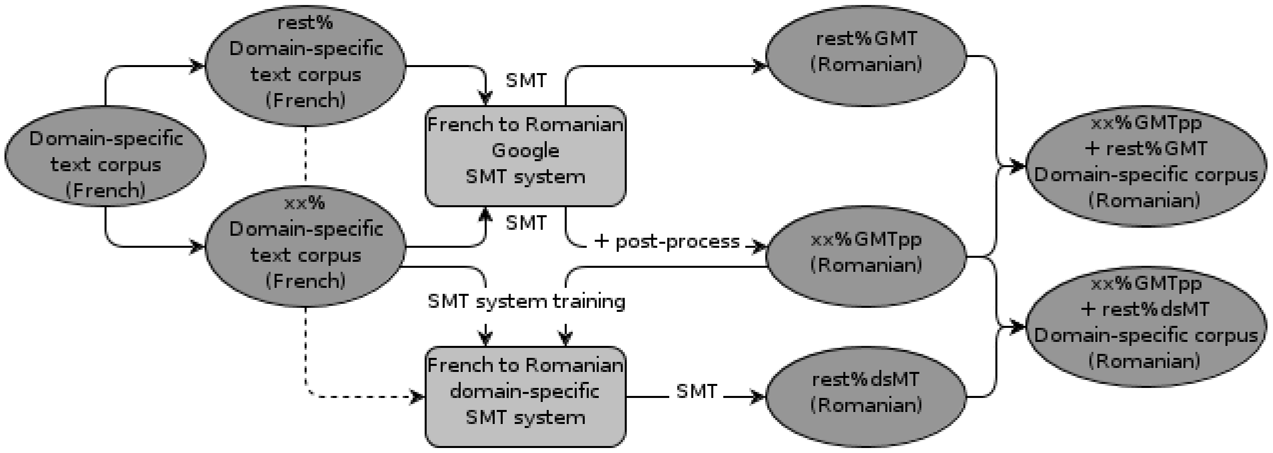

rest% Domain-specific text corpus (French)
Domain-specific text corpus (French)
xx% Domain-specific text corpus (French)
SMT
French to Romanian Google SMT system
SMT
+ post-process
SMT system training
French to Romanian domain-specific SMT system
SMT
rest% GMT (Romanian)
xx% GMTpp (Romanian)
rest% dsMT (Romanian)
xx% GMTpp + rest% GMT Domain-specific corpus (Romanian)
xx% GMTpp + rest% dsMT Domain-specific corpus (Romanian)

Figure 6.6 The two semi-supervised SMT-based language portability methods

To progressively answer this question, we started to *correct the Google translated corpus*. Ideally, the whole corpus should have been corrected, but, due to time constrains, we have only corrected a part of it (*xx%*). This post-processed part of the corpus is further called *xx%GMTpp*. This part was afterwards concatenated with the rest of the Google translated corpus (denoted *rest%GMT*) to obtain a complete (100%) domain-specific corpus. This is the first semi-supervised method of obtaining a Romanian domain-specific corpus and it is graphically represented in the upper part of Figure 6.6.

The second semi-supervised adaptation method regards the *xx%* of the French domain-specific corpus and the Romanian *xx%GMTpp* corpus as parallel corpora and uses them to train a domain-specific machine translation system. Undoubtedly, the resulting SMT system will be worse than Google's when *xx%* is small, but it may out-perform Google's as *xx%* increases. The trained SMT system is afterwards used to translate the rest of the domain-specific corpus. This part of the corpus, which is obtained by translation with our own domain-specific SMT system, is further called *rest%dsMT*. In the end, as Figure 6.6 depicts, *xx%GMTpp* is concatenated with *rest%dsMT* to create a complete (100%) domain-specific corpus.

Figure 6.6 describes the whole methodology for obtaining the *two Romanian partly-post-processed corpora*, given the French domain-specific corpus. These two corpora have been further used to create domain-specific language models and, eventually, domain-specific ASR systems, using the general methodology presented in Figure 6.5.

6.2.2 The domain-specific SMT system

The second semi-supervised domain adaptation method discussed in the previous section assumed the existence of a domain-specific SMT system. This SMT system has been developed with the Moses Toolkit. No minimum error rate training (mert) has been done due to the small amount of available data. The *xx%* of the French domain-specific corpus and the *xx%* Google translated and then post-processed Romanian corpus were regarded as parallel corpora and were used for training. The same post-processed corpus was also used to create a domain-specific language model (also needed to train the SMT system).

The size of the training corpus had varied from 500 phrases (5% of the domain-specific corpus) to 4000 phrases (40% of the domain-specific corpus). There was no optimization corpus and no test corpus because neither optimization, nor evaluation was performed. The method and consequently the translation system were evaluated only in the context of ASR adaptation (see next sections).

Obviously, this domain-specific SMT system could have been further ameliorated by improving the language model for the target language and/or by optimizing BLEU or some other metrics. These optimizations haven't been performed yet, because our main interest in this study was to validate the methodology.

6.2.3 Experimental setup

A domain-specific French corpus was required for the various domain adaptation experiments. For this task we have used the French *media* corpus: a tourism-specific corpus comprising 10k phrases summing up to a total of 64k words. The phrases are transcriptions of spontaneous dialogues between the tourism agent and various people asking for hotel bookings. 300 of these phrases were removed from this corpus, were manually translated to Romanian and were recorded by three speakers. This is how the CS_03 continuous speech database was created. The resulted speech data (900 single-phrase audio clips) sums up to about 1 hour of speech. The CS_03 database is used for evaluation in all the following speech recognition experiments. The rest of the *media* corpus is used for language model adaptation, just as specified in the previous section.

The domain adaptation methodology also requires a general Romanian text corpus. For this purpose we have concatenated the three general Romanian text corpora described in Section 4.3.2: europarl, 9am and hotnews. The resulted corpus comprises different types of news and discussions in the European Parliament. It consists of 9.8M phrases summing up to a total of about 169M words.

The speech recognition experiments presented in the next section employ the best CMU Sphinx-based acoustic model developed in Section 5.4.2: AM-11. This acoustic model uses 5-states HMMs and a total of 4000 senones to contextually model the 36 phonemes in Romanian. Each senone uses a Gaussian model with 16 mixtures as output probability distribution function. The acoustic model was trained using the WORDS and the CS_01 training speech database (approximately 54 hours of speech recorded by 17 speakers).

All the language models used in the ASR experiments are trigram, closed-vocabulary language models employing the Good-Turing smoothing technique and have been developed using the SRI-LM toolkit. This toolkit was also used for the interpolation of the general language model with the various domain-specific language models. The interpolation was systematically done with the weights 0.1 for the general language model and 0.9 for the domain-specific language model because our goal was to create a domain-specific ASR (the domain-specific LM should prevail). The interpolation weights tuning has not been considered for the moment. For the general language model, the number of unigrams had to be limited to the most frequent 64k due to the ASR decoder (Sphinx3) limitation.

The phonetic dictionary for all the ASR experiments was created using the technique presented in Section 4.1.3.4. Only 45k words out of the 64k had phonetic transcriptions within the initial phonetic dictionary. For the other 20k words we have utilized the graphemes-to-phonemes tool to produce phonetic transcriptions. Consequently the vocabulary for the speech recognition experiments consists of 64k words.

6.2.4 Experimental results

The evaluation of all the language models was done in terms of perplexity (PPL), out-of-vocabulary (OOV) rate, trigram hits and speech recognition word error rate (WER), all calculated on the test set. Among these four performance figures, the most important is the WER because, in fact, it is the only ASR performance figure. Nevertheless, the other three metrics also lead to important conclusions.

6.2.4.1 The Unsupervised Domain Adaptation Method

The unsupervised domain adaptation method is evaluated first and the results are presented in Table 6.8. We see that the unsupervised adaptation method produces a domain-specific language model (Exp 0) which is significantly better than the general language model. In other words, the Google-translated domain-specific corpus is much better than the general Romanian corpus on the domain-specific task. The interpolation of these two language models issues an even better language model (Exp 100).

Table 6.8 Unsupervised domain adaptation method results

Exp	LM	PPL	OOV	3gram hits	WER
-	general LM	164.7	4.27%	51.0%	29.7%
0	domain-specific LM	40.8	3.15%	31.1%	18.7%
100	domain-specific LM interpolated with out-of-domain LM	42.5	0.80%	55.4%	16.2%

Note that the general language model and the domain-specific language model have relatively high out-of-vocabulary rates (4.27%, respectively 3.15%). On the other hand, the interpolated language model has a very small out-of-vocabulary rate (0.80%). We can conclude that the general Romanian corpus lacks some domain-specific words, while the domain-specific translated-corpus lacks some general Romanian words. Nevertheless, the two corpora complement each other and the interpolated language model benefits from both the vocabularies.

Another interesting discussion regards the perplexity and the trigram hits for the three language models. The general language model has a large perplexity on the test set (it *is not* suitable to predict tourism-specific phrases), although 51.0% of the words are full trigram hits. This means that more than half of the 3-word sequences in the test corpus appeared in the training corpus. On the contrary, the domain-specific language model exhibits a better perplexity on the test set (it *is* suitable to predict tourism-specific phrases), although only 31.1% of the 3-word sequences in the test corpus appeared in the training corpus. Its better perplexity comes from higher prediction probabilities assigned to domain-specific unigrams and bigrams.

The interpolated language model benefits from the low perplexity of the domain-specific language model and the high trigram hits of the general language model. Thanks to this, its speech recognition results are the best. In conclusion the machine translated corpus plays a significant role in the improvement of the general language model and consequently the improvement of ASR system for a domain-specific task.

6.2.4.2 The Semi-supervised Domain Adaptation Methods

The results obtained for the semi-supervised adaptation methods are presented in the next tables. Table 6.9 shows the results for the domain-specific language models before interpolation with the general LM, and Table 6.10 shows the results after interpolation with the general LM.

The left part of the tables (experiments 1 – 5 and 101 – 105) evaluates the language models created using the first semi-supervised method (as described in Section 6.2.1.2). This means that the Google SMT system has been used to translate the whole French corpus and *xx%* of the translated corpus has been post-processed. The resulted Romanian corpus has been used to create the language models evaluated in these experiments. We will further refer to these systems as “first-method systems”.

The right part of the tables (experiments 6 – 10 and 106 – 110) evaluates the language models created using the second semi-supervised method (as described in Section 6.2.1.2). The Google SMT system has been used to translate only *xx%* of the French corpus. This part was afterwards post-processed and used to train the domain-specific SMT system. The latter was needed to translate the rest of the French corpus. The resulted Romanian corpus was used to create the language models evaluated in these experiments. We will further refer to these systems as “second-method systems”.

Please note that the experimental results for the unsupervised domain adaptation method (Exp 0 and Exp 100) were also listed in these two tables on both the left and the right side because they represent the baseline for all the other experiments.

Several conclusions can be drawn given the results in Table 6.9. First, we observe that even when a small amount of data (such as 5% of the Google translated text) was post-processed, all the performance figures are significantly better for the first-method system (Exp 1 compared to Exp 0). On the other hand, the second-method system that uses these 5% displays a significantly higher WER (Exp 6 compared to Exp 0). Even if the trigram hits and perplexity are better, the out-of-vocabulary rate is much worse and it causes the higher WER. This happens because these 5% (500 phrases) are not enough to train a robust SMT system; many words cannot be translated by this system, resulting in a pseudo-Romanian domain-specific corpus, which is clearly not suited for language modeling.

Table 6.9 Domain-specific language models results (before interpolation with general LM)

		+ rest% GMT							+ rest% dsMT			
Exp	**xx%GMTpp**	**PPL**	**OOV**	**3gram hits**	**WER**		**Exp**	**xx%GMTpp**	**PPL**	**OOV**	**3gram hits**	**WER**
0	00%	40.8	3.15%	31.1%	18.7%		0	00%	40.8	3.15%	31.1%	18.7%
1	05%	34.8	2.08%	34.0%	15.1%		6	05%	31.8	6.68%	35.3%	22.0%
2	10%	32.5	1.76%	35.2%	14.6%		7	10%	28.4	3.95%	38.4%	17.4%
3	20%	28.7	1.50%	37.9%	13.0%		8	20%	25.3	2.88%	41.2%	15.4%
4	30%	26.3	1.39%	39.4%	12.7%		9	30%	23.6	2.30%	42.1%	14.2%
5	40%	24.8	1.39%	41.3%	12.5%		10	40%	23.5	1.98%	42.7%	13.6%

Table 6.10 Improved domain-specific language models results (after interpolation with general LM)

		+ rest% GMT							+ rest% dsMT			
Exp	**xx%GMTpp**	**PPL**	**OOV**	**3gram hits**	**WER**		**Exp**	**xx%GMTpp**	**PPL**	**OOV**	**3gram hits**	**WER**
100	00%	42.5	0.80%	55.4%	16.2%		100	00%	42.5	0.80%	55.4%	16.2%
101	05%	34.4	0.80%	56.0%	14.6%		106	05%	36.3	0.80%	58.8%	14.2%
102	10%	32.4	0.53%	56.8%	13.9%		107	10%	30.1	0.53%	58.6%	12.7%
103	20%	29.0	0.48%	57.7%	13.1%		108	20%	26.7	0.48%	59.5%	12.6%
104	30%	26.6	0.48%	58.2%	12.4%		109	30%	24.3	0.48%	59.9%	11.6%
105	40%	25.2	0.48%	59.1%	12.2%		110	40%	23.8	0.48%	60.2%	11.5%

A second important conclusion is that both semi-supervised methodologies issue better and better ASR systems as more machine translated phrases are being post-processed (the only exception is the one presented and explained above). The growth in performance saturates as more and more data is being post-processed.

Comparing the left and the right parts of Table 6.9, we see that, when the same amount of data is post-processed, the second-method systems systematically display better trigram hits. This means that the newly developed SMT systems produced translations which include some new and useful trigrams. Nevertheless, the WERs for the second-method systems are higher due to the higher OOV rates. The OOV rate problem was already explained: the domain-specific SMT systems can only translate the words found in the small training corpus, leaving the other words in their “French version”. On the other hand, the Google MT system is able to adequately translate all the phrases to Romanian.

In conclusion, Table 6.9 states that the first-methodology systems are more robust than the second-methodology systems (when the domain-specific language models are not interpolated with the general language).

Let’s take a look now at the results in Table 6.10. First of all, for both semi-supervised methodologies we observe the same trend of lower WERs as more and more phrases are being post-processed. We also see that after the interpolation (Table 6.10), the OOV rates are equal for the two methodologies (when the same amount of data is post-processed). This means that the lack of coverage which characterized the second-method systems before interpolation (Table 6.9) has been overcome. Consequently, the second-methodology systems continue to be better in terms of perplexity and trigrams hits, but now outperform the first-method system in terms of WER (thanks to the fewer OOV words).

Comparing the corresponding lines in the two tables, we conclude that, after interpolation, the OOV rates and the trigram hits are much better. This is why the WERs are also lower for these ASR systems (the ones which benefit from the large coverage of the general language model).

To conclude this section: when the domain-specific language model, created using the second semi-supervised methodology, is interpolated with a general language model, the relative improvement in WER (for the corresponding ASR system) varies between 12% and 29% depending on the amount of machine translated text that was manually corrected (post-processed). Instead, if the first semi-supervised methodology is used, the relative improvement in WER is generally smaller (10% to 25%).

6.2.5 In-depth n-gram analysis

As shown in the previous section, the improved domain-specific language models have a good ability to predict (55% to 60% trigram hits) both domain-specific word sequences and out-of-domain word sequences (thanks to the interpolation with a general language model). In this study, the general language model was the same for all experiments, so, if we want to answer the question *why and how the proposed methodologies bring improvements in ASR?*, we have to analyze the various domain-specific language models before interpolation. Table 6.9 showed the results for all these language models. Some of them (Exp 0, 5 and 10) were selected and analyzed in Table 6.11 from the point of view of their ability to predict specific words (trigrams example). The selected language models have been created with corpus obtained using the unsupervised methodology (Exp 0), the first semi-supervised methodology (Exp 5) and the second semi-supervised methodology (Exp 10).

Table 6.11 shows seven trigram examples and analyses the way the language models manage to predict the bolded word in the given context. *3-gram* means the language model was able to predict the bolded word in the given trigram context and *2-gram* means the language model needed to back-off to bigrams to predict the bolded word. *1-gram* means the language model needed to back-off to unigrams to predict the bolded word and *OOV* asserts the LM cannot predict the bolded word (it is out-of-vocabulary).

Note that there are trigrams which can be very well predicted by all the analyzed language models (type a), but also trigrams that can only be predicted by the domain-specific language models (type b). The importance of the interpolation with the broader, general language model is motivated by its higher trigram hits (51%) and by trigrams which can only be well predicted by it (type c). The plus brought by the semi-supervised methods is revealed by examples of type d. Only a few trigrams can be better predicted by the second-method systems, when compared to the first-method systems (see the small difference in trigram hits and examples of type e). And, of course, there are examples of trigrams which cannot be well predicted by any of the analyzed language models (type f). The frequency of occurrence for these six types is difficult to estimate, but the big picture is illustrated by the trigram hits column.

Table 6.11 N-gram hits for the general and domain-specific language models (examples)

			Trigram examples							
			a	**b**	**b, d**	**c, d**	**c**	**c, d, e**	**f**	**Type**
			o cameră **single**	care acceptă **animale**	într-o locație **liniștită**	puteți să-mi **dați**	și acum **pentru**	prea scumpă **într-un**	nopți pentru **Belfort**	**Ro text**
Exp	**Language model**	**3-gram hits**	a **single** room	which accepts **animals**	in a **quiet** place	can you **give** me	and now **for**	too expensive **in a**	nights at **Belfort**	**En text**
-	general LM	51.0%	3-gram	1-gram	1-gram	3-gram	3-gram	2-gram	OOV	
0	0%GMTpp + 100%GMT	31.1%	3-gram	3-gram	2-gram	1-gram	1-gram	1-gram	1-gram	
5	40%GMTpp + 60%GMT	41.3%	3-gram	3-gram	3-gram	3-gram	1-gram	2-gram	1-gram	
10	40%GMTpp + 60%dsMT	42.7%	3-gram	3-gram	3-gram	3-gram	1-gram	3-gram	1-gram	

6.2.6 Conclusion

This study proposed several language portability methodologies to address the absence of domain-specific text resources for a particular language, given domain-specific data in a different language. We have particularly investigated the possibility of porting a tourism-specific French corpus to Romanian with the final goal of creating a tourism-specific ASR system for Romanian. Several SMT-based methods were proposed to create domain-specific language models. These methods were in the end evaluated in the context of ASR. The first method used Google's French-to-Romanian MT system in an unsupervised fashion. Two other semi-supervised methods, which benefit from manually corrected data, were introduced and compared with the unsupervised method. The relative improvement in WER brought by the semi-supervised methods varies from 10% to 29%, depending on the amount of machine translated text that was manually corrected (post-processed).

A more in-depth analysis, explaining the reasons why the proposed methods bring improvements in ASR, was also made and several examples of trigrams were given to illustrate the various language prediction scenarios.

Pragmatically, this study summarized the needed resources and proposed a SMT-based methodology, which could be used to develop a domain-specific ASR system for any under-resourced language, given that specific resources are available for a high-resourced language.

On the short term, the results presented in this study could be improved by tuning the domain-specific SMT system and the LM interpolation weights. A possible improvement could also be obtained by combining the semi-supervised methods.

On the long term, we plan to further validate the adaptation methodology by applying it for other specific domains and also for other pairs of source-target languages. Another interesting perspective would be the usage of the proposed methodology when domain-specific data is available in more than one high-resourced (source) languages.

6.3 SUMMARY

This chapter has presented our various attempts to create a general language model for Romanian and some innovative and original language model adaptation methodologies.

The first section of this chapter dealt with the task of constructing a general language model. The first initiative aimed to create a knowledge-based grammar for Romanian. A larger team has worked on this project and developed the required resources. The author of this thesis created a graphical-interface tool to help the development of grammar rules. In the end, the whole initiative has proved to be a failure due to the high complexity of the resulted language model. The second initiative aimed to create a statistical language model for Romanian. Large amounts of textual data were needed for this task. The acquisition, processing and analysis of the text corpora were described in Section 4.3. The resulted n-gram language models proved to be very successful for the speech recognition task. We have obtained a relative word error rate improvement of 50% over the ASR system that employed a basic word-loop language model.

The second section of this chapter introduced several SMT-based domain adaptation methods. A foreign domain-specific corpus was translated to Romanian and used for language modeling. The translation is done in an unsupervised and a semi-supervised fashion. The unsupervised method manages to create a domain-specific language model which is evaluated and compared with the general language model. For the domain-specific task, we have obtained a relative word error rate improvement of 45%. The semi-supervised methods output even better domain-specific language models. When compared to the general language model, the relative word error rate improvement varies between 51% and 61% depending on the amount of machine translated text that was manually corrected (post-processed).

CHAPTER 7

SPEAKER-INDEPENDENT, LARGE-VOCABULARY CONTINUOUS SPEECH RECOGNITION SYSTEM

7.1 THE INTEGRATED METHODOLOGY

The previous three chapters presented the various resources, tools and components we have created in our effort towards the final goal: the speaker-independent, large-vocabulary continuous speech recognition (LV-CSR) system. It is time to assemble them all together and evaluate the LV-CSR system.

Chapter 4 focused on the review, acquisition and analysis of phonetic, speech and text resources required in ASR. The resource acquisition process was one of the most important steps in this work, because these resources are the key components in phonetic, acoustic and language modeling. For example, continuous speech modeling was solely achieved thanks to the isolated words database (WORDS) and the general continuous speech database (CS_01). The general language model for Romanian was solely created using the extensive news (9am and hotnews) and parliament minutes (europarl) text corpora. Contrary to the speech material, the text material had to be processed before it could be used for language modeling. Among other processing tasks, the diacritics restoration task was a real challenge and had to be employed by a specially created tool.

Chapter 5 aimed to create the speaker-independent, continuous speech acoustic model. The speaker-independency is still questionable because the training speech database contains speech materials only from 17 speakers. The speaker-independency desiderate will be discussed further on in this chapter. This acoustic model was created using the CMU Sphinx toolkit and the default training strategy it provides. The acoustic model was evaluated in Chapter 5, but only in conjunction with a basic, small-vocabulary word-loop language model.

Finally, the large-vocabulary attribute of our CSR system was achieved by creating a general language model for Romanian using all the text corpora available (europarl + 9am + hotnews). In Chapter 6 we present the steps we took towards creating this general language model and we evaluate it both separately and in the context of ASR. The experiments presented in Section 6.1.2 use a 45k words vocabulary instead of a 64k words vocabulary (which is the general-accepted large-vocabulary size) due to the incompleteness of the phonetic dictionary.

The phonetic dictionary, which was available in our research group prior to this work, is obviously incomplete. It does not contain any proper names and it even lacks some essential, frequently used Romanian words. Nevertheless it served as a very good basis for creating a graphemes-to-phonemes (G2P) automatic conversion tool. This tool now enables us to create the phonetic vocabulary starting from the words vocabulary of any ASR task. Consequently, all the experimental results which will be reported in this chapter will benefit from a full 64k words vocabulary automatically created using the G2P conversion tool.

7.2 LV-CSR EXPERIMENTAL SETUP

The acoustic model used in the LV-CSR system is the one presented in Section 5.4.2 and denoted AM-11. This acoustic model was created using the CMU Sphinx Toolkit default training strategy. We employed 5-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 4000. Every senone was modeled with a Gaussian Mixture Model (GMM) with 16 Gaussian components.

The acoustic model was trained with the training part (90%) of the WORDS database and the training part (90%) of the CS_01 database. The total amount of training speech data summed up to about 54 hours of speech from 17 different speakers (7 males and 10 females).

The language model (LM) used in the LV-CSR system is the one presented in Section 6.1.2 and denoted europarl+9am+hotnews. This LM is a trigram language model and was constructed using the SRI-LM Toolkit. As a smoothing method we have utilized the Good-Turing discount method (the default in SRI-LM). The number of unigrams for the language model was limited to the most frequent 64k words due to a limitation of the CMU Sphinx speech decoder. Consequently, the size of the vocabulary for the LV-CSR system is 64k words (it can only output words from this subset).

The language model was trained with a Romanian text corpus obtained by concatenating the europarl corpus (a small size corpus comprising parliament minutes), the 9am corpus and the hotnews corpus (two large size corpora comprising news from a wide range of domains). The resulted corpus has about 169 million words.

Phonetic transcriptions are needed for all the 64k words in the language model. The 64k phonetic dictionary for the LV-CSR system was obtained as follows:

a) all the words (within the 64k words vocabulary) which were found in the 600k words phonetic dictionary were transcribed using this dictionary,
b) all the other words were transcribed using the graphemes-to-phonemes tool.

7.2.1 Evaluation setup

The purpose of this evaluation is to estimate the performance of the LV-CSR system for known and unknown speakers. By known speakers we denote the speakers which were also part of the training process. By unknown speakers we denote speakers to which the system has never been exposed before. This experiment aims to evaluate the speaker-independency attribute of the LV-CSR system.

The evaluation for known speakers was made using the testing part of the CS_01 database. This evaluation part of the database has about 2 hours of continuous speech recorded by 11 speakers.

The evaluation for unknown speakers was made using the CS_04 database. This database has about 2 hours of continuous speech recorded by 8 speakers.

The testing part of the CS_01 database and the CS_04 database comprise the same 100 audio clips for every speaker. This ensures that the evaluation is done in an identical manner for both known and unknown speakers. More details about the continuous speech databases are given in Section 4.2.2.

7.3 LV-CSR Experimental Results

The results for the speaker-independency experiments are summarized in Table 7.1. The word error rate for known speakers is 19.2%, while the word error rate for unknown speakers is 22.2%. In our opinion the decrease for unknown speakers (15.6% relative) is insignificant. The system was expected to have more difficulties to recognize the speech uttered by unknown speakers. Detailed results, regarding the word error rate for every speaker involved in the experiment are given in Table 7.2 (for known speakers) and in Table 7.3 (for unknown speakers).

Table 7.1 Speaker-independency experiment - results summary

Exp	**Acoustic model**	**Language model**	**Evaluation test set**	**WER [%]**
1	ASRS-11	europarl + 9am + hotnews (64k words)	CS_01	19.2
2			CS_04	22.2

Table 7.2 Speaker-independency experiment - results for known speakers

WER [%]		**Speaker**											
		01	02	03	04	06	10	16	17	18	19	20	all
Test set	CS_01	29.8	20.3	20.3	16.2	13.4	15.5	15.0	31.1	16.0	17.4	16.0	19.2

Table 7.3 Speaker-independency experiment - results for unknown speakers

WER [%]		**Speaker**								
		21	22	23	24	25	26	27	28	all
Test set	CS_04	18.5	20.4	17.4	37.5	23.6	18.9	23.8	17.2	22.2

The detailed, per-speaker results presented in Table 7.2 for known speakers and in Table 7.3 for unknown speakers reveal some interesting aspects regarding the LV-CSR system. First of all we observe that the results have a relatively large dispersion: the standard deviation for known speakers WER is 5.95 and the standard deviation for unknown speakers WER is 6.70. This is due to some special cases (speaker 01, 17 and 24) which are poorly recognized by the LV-CSR system. If we ignore these values, the standard deviation has more acceptable (lower) values: 2.31, and respectively 2.76. Nevertheless the conclusion is certain: there are some types of speakers for which the system's performance (29.7%, 31.1%, respectively 37.5% WER) is not acceptable. These special cases have to be further analyzed in order to adapt the system to all kind of voices. A first analysis shows that all the three speakers are males.

A second conclusion regards the various per-speaker WERs values. Even if the average WER for known speakers is lower than the average WER for unknown speakers, there are some special cases of unknown speakers (speakers 21, 23, 26 and 28) for which we obtain very good recognition results. These results are even better than some of the results obtained for known speakers! It is clear that some “unknown” voices are similar to some of the voices on which the system was trained and this explains the good recognition results. These are encouraging results, showing that the development methodology is right. The further improvements should regard collecting more speech data from speaker types which are poorly recognized.

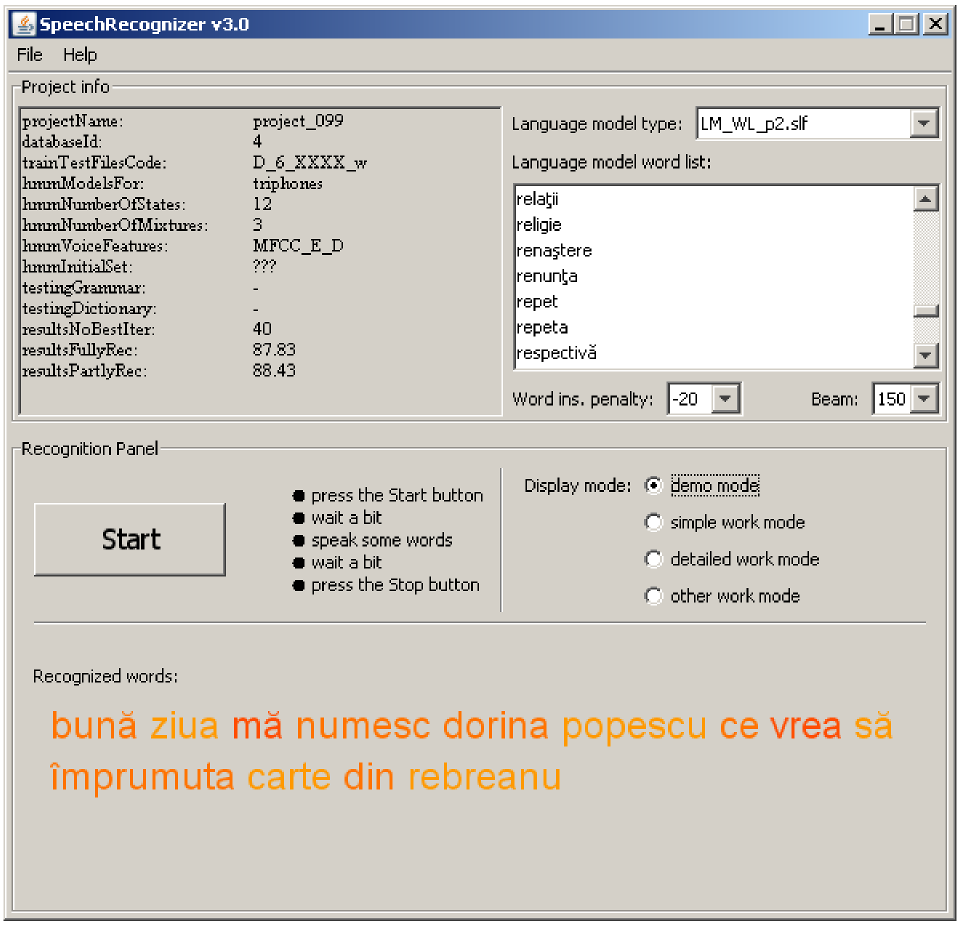
In conclusion, the speaker-independency experiment showed that the LV-CSR system suffers from an average of 15.6% relative performance drop (expressed as WER increase) when it is required to recognize speech uttered by unknown speakers. This performance drop is, in our opinion, very small and can be corrected by a further enlargement of the continuous speech database (by collecting speech data from more speakers).

7.4 LV-CSR GUI LIVE-DEMO

All the experiments presented so far in this thesis were done in an offline, experimental setup. For demonstration purposes a live-demo application was created. The application is developed in the Java programming language and benefits from a graphical user interface (GUI) that allows the user to load specific acoustic models and specific language models at his own choice. Moreover, the live-demo application provides the user with an easy way to configure some basic decoding parameters such as the word insertion penalty and the beam width value. Several information about the acoustic model, such as the basic speech units, the speech features and the HMM topology, are listed whenever a new speech recognition project is loaded. Also, the list of words composing the currently loaded language model is displayed in the graphical user interface. A screenshot illustrating the graphical user interface of the live-demo application is presented in Figure 7.1.

The live-demo application is very easy to use: first, a speech recognition project, developed with HTK, must be loaded. Afterwards, the application allows the user to select a language model for the specific ASR task. The valid words for the specific task are then listed in the *Language model word list* field. In the end, the user can press the *Start* button, speak a sentence and press the *Stop* button. Obviously, a microphone has to be connected to the computer so the application can record the spoken phrase. After processing, the application displays the hypothesis phrase. In the example presented in Figure 7.1 the user has spoken the following phrase: *Bună ziua! Mă numesc Dorina Popescu* ***şi aş*** *vrea să* ***împrumut o*** *carte* ***de*** *Rebreanu*. The words which were wrongly recognized were written in bold in the previous sentence. The application does not have the original phrase (does not prompt the user to speak a specific phrase). Consequently, it cannot compute the instantaneous word error rate.

This speech recognition application can be also used for development. It provides several display modes for investigating the recognition likelihood for each specific word in the hypothesis phrase. In demo mode the various words in the hypothesis phrase are colored differently based on their recognition probability. Light green means a very high recognition probability, while intense red means a very low recognition probability. In the example illustrated in Figure 7.1 all the words, except for the words *mă* and *vrea*, are moderately well recognized. In other display modes (simple work more, detailed work mode, etc.) the application lists various other information helping the user to investigate the recognition errors.



projectName:	project_099
databaseId:	4
trainTestFilesCode:	D_6_XXXX_w
hmmModelsFor:	triphones
hmmNumberOfStates:	12
hmmNumberOfMixtures:	3
hmmVoiceFeatures:	MFCC_E_D
hmmInitialSet:	???
testingGrammar:	-
testingDictionary:	-
resultsNoBestIter:	40
resultsFullyRec:	87.83
resultsPartlyRec:	88.43

Language model type: LM_WL_p2.slf

Language model word list:

relaţii
religie
renaştere
renunţa
repet
repeta
respectivă

Word ins. penalty: -20 Beam: 150

- press the Start button
- wait a bit
- speak some words
- wait a bit
- press the Stop button

Display mode: [✓ demo mode] simple work mode detailed work mode other work mode

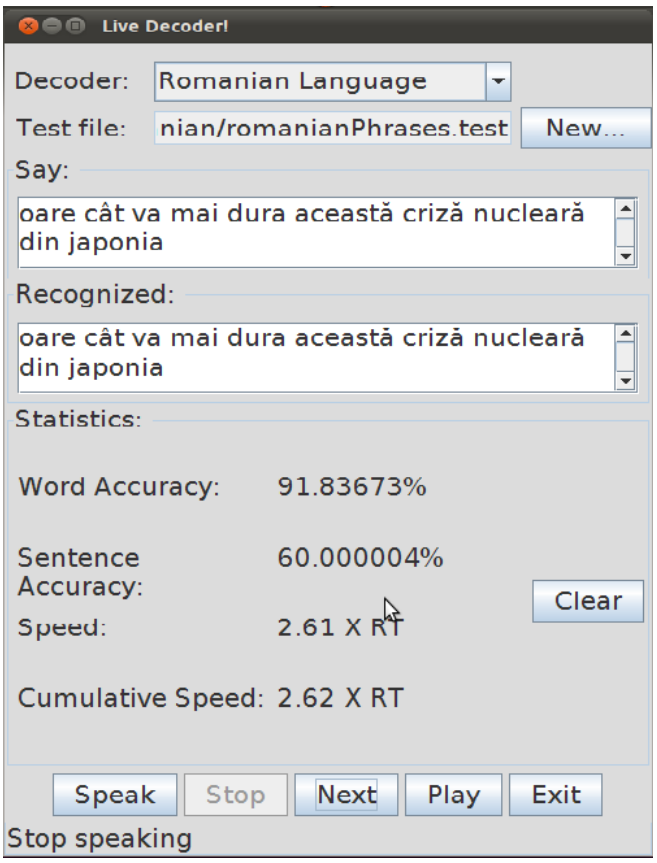
Recognized words:

bună ziua mă numesc dorina popescu ce vrea să împrumuta carte din rebreanu

Figure 7.1 The HTK-based speech recognition live-demo application

The speech recognition live-demo application uses the HTK Toolkit for the actual speech decoding. The first version of the application was very slow due to the loose interconnectivity between the Java live-demo application and the C modules in HTK. For the second version an extensive effort was employed to overcome this drawback. The HTK recognition module was modified and optimized to run the recognition process faster and to load the acoustic models and the language models only at start time. Moreover the recognition module was transformed into a dynamic linked library which could be interconnected more tightly with the Java live-demo application resulting in a real-time speech recognition application.

The switch from HTK Toolkit to CMU Sphinx Toolkit, as background speech recognition engine, made this speech recognition live application useless. As it was developed to tightly interconnect with the HTK Toolkit, the application had to be in-depth modified to support CMU Sphinx. Luckily the Sphinx 4 toolkit comes with a live-demo itself. This live-demo application can be configured to use specific acoustic models and language models designed with the CMU Sphinx Toolkit. Consequently, in order to use the Romanian LV-CSR system we were only required to configure this default live-demo application to use the models developed so far for Romanian. A screenshot illustrating the graphical user interface of the CMU Sphinx live-demo application is presented in Figure 7.2.


Live Decoder!

Decoder: Romanian Language

Test file: nian/romanianPhrases.test

Say:
oare cât va mai dura această criză nucleară din japonia

Recognized:
oare cât va mai dura această criză nucleară din japonia

Statistics:

Word Accuracy:	91.83673%
Sentence Accuracy:	60.000004%
Speed:	2.61 X RT
Cumulative Speed:	2.62 X RT

Stop speaking

Figure 7.2 The CMU Sphinx speech recognition live-demo application

The CMU Sphinx live-demo presents the hypotheses phrases in a slightly different manner than the live-demo we have developed for HTK. First of all the application allows the user to select a so-called decoder which is composed of the acoustic model, the language model, the phonetic dictionary and a large amount of configuration parameters. In the example presented in Figure 7.2 we have selected the speech decoder for the Romanian language (the LV-CSR system presented in the previous sections). The user is then able to select a test file comprising example phrases. The first example phrase is displayed in the *Say* text field (the application prompts the user to speak a specific phrase). The user can press the *Speak* button, speak the phrase and then click the *Stop* button. The spoken phrase is recorded and decoded and the hypothesis phrase is displayed in the *Recognized* text field. Moreover, this demo-application computes some statistics regarding the speech recognition process and the recognition results: the word accuracy, the sentence accuracy and the recognition speed. In the example illustrated in Figure 7.2, the instantaneous word accuracy is 100% (all words were correctly recognized), the cumulative word accuracy is about 91.8% and the cumulative recognition speed is about 2.62 x RT (real-time). 2.62 x real-time can be understood as follows: the recognition process took 2.62 more time than the duration of the spoken phrase.

CHAPTER 8

CONCLUSIONS

8.1 GENERAL CONCLUSIONS

The main objective of this thesis was to create a speaker-independent, large-vocabulary continuous speech recognition system for the Romanian language. This main target can be decomposed as a tree-part effort: the development of a continuous speech recognition system, the large-vocabulary attribute of the speech recognition system and the speaker-independency attribute of the speech recognition system. The first and second parts were successfully accomplished by the work described in this thesis, while the third part is still not entirely solved.

This thesis presents the successive steps which were employed by the author in order to create a high-performance LV-CSR system. After describing the state-of-the-art in speech recognition in Chapter 1 and the theoretical aspects regarding speech recognition and statistical machine translation in Chapter 2 and 3, Chapter 4 presents the resources and processing tools required to create a LV-CSR system. Phonemes were chosen as basic speech units for the HMM-based speech recognition system, therefore a phonetic dictionary that maps words to their phonetic form is indispensable. Section 4.1 discusses various aspects regarding Romanian phonetics and draws some important conclusions regarding the most frequent phonemes used in this language. Going further, this section describes the phonetic dictionary which was available within our research group prior to this work. The inherent limitation of this phonetic dictionary is illustrated and an automatic solution of creating phonetic transcriptions (a graphemes-to-phonemes tool) is proposed. The SMT-based graphemes-to-phonemes tool is in-depth described and compared to other similar tools found in the literature and proves to be the most effective one.

The speech database (a database of speech audio clips with associated textual transcriptions) is the most important resource required to create an ASR system. Section 4.2 makes a review of the speech databases available for Romanian and complains about the inexistence of a sufficiently large database for speaker-independent acoustic modeling. Going further, this section presents and analyses various speech database acquisition methodologies, selects the most suitable one and describes the speech databases acquired using it. The acquired speech databases are in the end analyzed and compared from a phonetic point of view.

Text corpora are an indispensable resource for statistical language modeling and consequently an indispensable resource in our LV-CSR system. This thesis makes a review of the text corpora available for Romanian in Section 4.3. The review serves as a motivation for acquiring larger text corpora in order to suitably train n-gram language models for Romanian. The acquisition and processing steps are detailed in the same section. Several NLP tools, among which a diacritics restoration tool, are designed, implemented and evaluated. The diacritics restoration system turns out to be one of the most efficient systems developed for Romanian. Section 4.3 ends with an analysis of the acquired text corpora. Important conclusions regarding the most frequently used words in Romanian and their language coverage are drawn in this section.

After the required resources are described and analyzed in Chapter 4, their usage is highlighted in Chapter 5, which focuses on acoustic modeling specific issues, and Chapter 6, which focuses on language modeling specific issues. Chapter 5 describes the design of the HMM-based acoustic model and introduces a hierarchical training strategy to optimize the training process. A large amount of experiments have been done to find the best design setup for isolated words acoustic models. Their results are summarized and some conclusions are drawn in Section 5.2. The following section introduces an innovative three-step isolated words recognition method that manages to optimize the speed of the decoding process. The real-time desiderate is satisfied for this recognition method, as opposed to the usual recognition method, which was less time-efficient. Chapter 5 ends with some continuous speech recognition experiments. All these experiments were made using a basic word-loop language model because at this point more complex language models were not available.

More efficient language models and adaptation techniques are described in Chapter 6. For large-vocabulary ASR it is mandatory to use more efficient language models (word-loop language models don not offer a satisfactory performance). Two types of language models were tried out for the Romanian language: finite state grammars (FSGs) and n-gram statistical language models. The first approach was a failure, because FSGs are only suitable for specific tasks, but the second one (which is the state-of-the-art in language modeling) was a success. Using a general n-gram language model we were able to create and evaluate the first LV-CSR system for Romanian. Its 19.0% word error rate is satisfactory for the moment for this very general ASR task.

An innovative ASR domain adaptation methodology is also introduced in Chapter 6. Domain adaptation is an important step when a general-domain ASR system is available, but a domain-specific ASR system with improved performance is required. An unsupervised and two semi-supervised adaptation methods are proposed and evaluated in Section 6.2. The results for the domain-specific ASR system are significantly better than the results for the general ASR system.

In Chapter 7 the speaker-independency attribute of the LV-CSR system was evaluated. For this task we have collected a new speech database with speech data from speakers which were not part of the training process. The experiment concluded that the relative performance drop for these unknown speakers is only 15.6% (22.2% WER for unknown speakers compared to 19.2% WER for known speakers). These results, and the large dispersion of the per-speaker results, show that the system is not yet speaker-independent. However, the average performance drop is not significant and we expect to reduce it by collecting more speech data from various speakers and use it in the training process.

Chapter 7 also presents two GUI live-demo applications. The first one was developed by the author of this thesis to integrate the HTK speech recognition toolkit, while the second one comes by default with the CMU Sphinx toolkit. These two applications can be used to demonstrate the capabilities of the LV-CSR system we have developed.

8.2 PERSONAL CONTRIBUTIONS

The personal contributions of the author of this thesis are organized in Chapters 4, 5, 6 and 7 and can be summarized as follows:

a) A classification of the most frequently used phonemes in Romanian (Section 4.1.1).

b) The design and implementation of a graphemes-to-phonemes conversion tool (Section 4.1.3). This tool uses SMT principles to "translate" words into phonetic forms and it was created solely by the author of this thesis. This graphemes-to-phonemes (G2P) conversion tool was compared and evaluated as the best G2P tool available for Romanian. Design and implementation details were also given in [Cucu, 2011c].

c) The acquisition of several Romanian speech databases (Section 4.2): BOOKS, PHONES, WORDS, CS_01, CS_02, CS_03 and CS_04. The acquisition process, as of June 2010, is also presented in [Petrea, 2010] and [Burileanu, 2010b]. The acquisition was done within a research group, but the author of this thesis had a leading role in this process. The speech databases were acquired using a speech recording tool which was designed and implemented by the author of this thesis. Note that the continuous speech material within databases CS_01, CS_02, CS_03 and CS_04 is the largest for the Romanian language.

d) The acquisition of several Romanian text corpora (Section 4.3): europarl, 9am and hotnews. Together, these three corpora form the largest text material available for the Romanian language. The acquisition of these corpora was also presented in [Cucu, 2011b]. These corpora were acquired solely by the author of this thesis.

e) The design and implementation of an NLP tool which "cleans" the text corpora and prepares them for statistical language modeling (Section 4.3.3).

f) The design and implementation of a diacritics restoration system for Romanian (Section 4.3.4). Details about the diacritics restoration system and the comparison with the best system for Romanian were also published in [Cucu, 2011b]. The diacritics restoration tool was created solely by the author of this thesis. This tool was compared and evaluated in this thesis as the second best diacritics restoration tool available for Romanian.

g) A classification of the most frequently used words in Romanian (Section 4.3.5) based on the largest text material available for Romanian.

h) Acoustic models (for isolated words) optimizations following a hierarchical training strategy (Section 5.2). The training strategy is not a contribution of the author of this thesis, but most of the optimization experiments were performed by the author of this thesis. These experimental results were published in [Petrea, 2010] and [Burileanu, 2010b].

i) The design and implementation of a three-step isolated words recognition algorithm (Section 5.3). This algorithm ensures real-time recognition and it was firstly described in [Cucu, 2011a]. The general design of this algorithm is a product of our research group, but specific design characteristics and the implementation was solely contributed by the author of this thesis.

j) Several acoustic models for continuous speech (Section 5.4) based on the continuous speech databases. The best acoustic model is also used in [Cucu, 2011b] and [Cucu, 2011c]. The best acoustic model is the core of the LV-CSR system and was solely created by the author of this thesis.

k) The design and implementation of a GUI tool for creating finite state grammars (Section 6.1.1). This tool was used to build the FSG for the Romanian language, which turned out to be a failure, but it can be successfully used and it is very efficient for building smaller task FSGs. This tool is solely the contribution of the author of this thesis.

l) A general statistical language model for Romanian (Section 6.1.2) based on the largest text material available for Romanian. This language model is also described and used in [Cucu, 2011b] and [Cucu, 2011c]. This language model is a main component of the LV-CSR system and is solely the contribution of the author of this thesis.

m) An ASR domain-adaptation methodology based on SMT principles (Section 6.2). The methodology comprises an unsupervised adaptation method, which is also in-depth described in [Cucu, 2011b], and two semi-supervised adaptation methods, also presented in [Cucu, 2011c]. Several researchers contributed to the design of the adaptation methodology, while the implementation and evaluation of these methods is solely the contribution of the author of this thesis.

n) The design and implementation of a large-vocabulary continuous speech recognition live-demo application (Section 7.4).

o) The speaker-independent LV-CSR system for Romanian (Chapter 7). The design and the development methodology for this system are contributions of the entire research group, but the language modeling component, the continuous speech acoustic model and the actual implementation are solely the contributions of the author of this thesis.

8.3 Future Work

The large-vocabulary continuous speech recognition system, which is the main output of this thesis, was only evaluated in ideal conditions: clean speech (SNR ~40 dB), read speech (not conversational, spontaneous speech), small inter-speaker variability (all the speakers were healthy, highly-educated students with ages between 24 and 28 years), etc. Given this, one of the perspectives of this work is obvious: improving the system's robustness. From this point further we can only regard this LV-CSR system as a baseline. Improving ASR robustness is also regarded as a hot topic in ASR by the international scientific community. There are already numerous methods and techniques which deal with noise reduction, spontaneous speech and, more generally, with environment and speaker variability. We plan to study, evaluate and improve these methods for the Romanian LV-CSR system.

The inter-speaker variability must also be approached by creating even more general acoustic models. The only way to accomplish this is by collecting a larger continuous speech database. By "larger", we mean a more representative database, with at least 150 speakers of different ages, different education, different social environments, etc. We expect this process to be even more difficult than the acquisition of the previous continuous speech database.

A second perspective consists in developing and applying various domain adaptation or speaker adaptation techniques to improve the performance of the ASR system for a specific application. The 19.2% word error rate is decent for a general LV-CSR system, but is not acceptable for a domain-specific or a speaker-dependent application. Therefore we intend to evaluate the various existing speaker adaptation techniques and domain-adaptation techniques to create more powerful systems. A first domain adaptation methodology was already presented and evaluated in this thesis, but we also have other ideas that might turn out to be even more efficient.

Following in a chronological and logical order after these first two perspectives it would be interesting to transform the current LV-CSR system into a real application that could be finally used and exploited by human users. The closest applications would be real-time dictation systems, transcription systems for conferences, law trials, news, etc. or movie subtitling systems.

Finally, we intend to study the possibility of developing a spoken dialogue system consisting of both speech recognition and speech synthesis. We expect that we will be able to reuse many of the resources and tools developed for speech recognition for speech synthesis and along with our experience in speech processing this could be an important plus.

PUBLICATIONS LIST

1. [Burileanu, 2008] Corneliu Burileanu, Cristina-Sorina Petrea, Andi Buzo, **Horia Cucu**, Alina Paşca, “Report on building a tool for Romanian spontaneous speech recognition,” *The Phonetician*, ISSN 0741-6164, No. 97 / 2008-I-II, pp. 68-98, 2008.
2. [Burileanu, 2010a] Corneliu Burileanu, Andi Buzo, Cristina-Sorina Petre, Diana Ghelmez-Hanes, **Horia Cucu**, “Romanian Spoken Language Resources and Annotation for Speaker Independent Spontaneous Speech Recognition,” *ICDT 2010*, pp.7-10, Athens, Greece, 2010.
3. [Burileanu, 2010b] Corneliu Burileanu, Cristina-Sorina Petrea, Andi Buzo, **Horia Cucu**, “Speech Recognition Experiments Starting from Isolated Words for Spoken Romanian Language,” *Multilinguality and Interoperability in Language Processing with Emphasis on Romanian*, Romanian Academy Publishing House, Bucharest, ISBN: 978-973-27-1972-5, pp. 231-244, 2010.
4. [Petrea, 2010] Cristina-Sorina Petrea, Andi Buzo, **Horia Cucu**, Miruna Paşca, Corneliu Burileanu, “Speech Recognition Experimental Results for Romanian Language,” *ECIT 2010*, ISSN: 2069-038X, pp. 13, Iaşi, Romania, 2010.
5. [Buzo, 2011a] Andi Buzo, **Horia Cucu**, Corneliu Burileanu, Miruna Paşca, Vladimir Popescu, “Word error rate improvement and complexity reduction in automatic speech recognition by analyzing acoustic model uncertainty and confusion,” *SpeD 2011*, ISBN: 978-1-4577-0439-0, pp. 67-74, 2011.
6. [Buzo, 2011b] Andi Buzo, **Horia Cucu**, Corneliu Burileanu, “Improving automatic speech recognition robustness for the romanian language,” *EUSIPCO 2011*, ISSN: 2076-1465, pp 2119-2122, 2011.
7. [Cucu, 2011a] **Horia Cucu**, Andi Buzo, Corneliu Burileanu, “Optimization methods for large vocabulary, isolated words recognition in Romanian language”, *University Politehnica of Bucharest Scientific Bulletin*, Series C, no. 2, ISSN: 1454-234x, pp. 179-192, Bucharest, 2011.
8. [Cucu, 2011b] **Horia Cucu**, Laurent Besacier, Corneliu Burileanu, Andi Buzo, ”Enhancing Automatic Speech Recognition for Romanian by Using Machine Translated and Web-based Text Corpora,” *SPECOM 2011*, pp. 81-88, Kazan, Russia, 2011.
9. [Cucu, 2011c] **Horia Cucu**, Laurent Besacier, Corneliu Burileanu, Andi Buzo, ”Investigating the Role of Machine Translated Text in ASR Domain Adaptation: Unsupervised and Semi-supervised Methods,” *ASRU 2011*, Hawaii, United States of America, to appear 2011.

REFERENCES

[9am] 9am online newspaper (http://www.9am.ro)

[Bahl, 1991] Bahl, L.R., et al., "Multonic Markov Word Models for Large Vocabulary Continuous Speech Recognition," *IEEE Transactions on Speech and Audio Processing*, 1993, vol. 1, no. 3, pp. 334-344, 1991.

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

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

[Baum, 1972] Baum, L.E., "An inequality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes," *Inequalities*, vol. 3, no. 1, pp. 1-8, 1972.

[Bertoldi, 2009] Bertoldi, N., Haddow, B., Fouet, J.-B., "Improved Minimum Error Rate Training in Moses," *The Prague Bulletin of Mathematical Linguistics*, pp. 1-11, February 2009.

[Beulen, 1997] Beulen, K., Bransch, E., Ney, H., "State tying for context dependent phoneme models," *Eurospeech 1997*, pp. 1179-1182, 1997.

[Bick, 2010] Bick, E., Greavu, A., "A Grammatically Annotated Corpus of Romanian Business Texts," *Multilinguality and Interoperability in Language Processing with Emphasis on Romanian*, Romanian Academy Publishing House, Bucharest, ISBN: 978-973-27-1972-5, pp. 169-182, 2010.

[Billa, 2002] Billa, J., Noamany, M., Srivastava, A., Liu, D., Stone, R., Xu, J., et al., "Audio indexing of Arabic broadcast news," *ICASSP 2002*, pp. 5-8, 2002.

[Bisani, 2003] Bisani, M., Ney, H., "Multigram-based grapheme-to-phoneme conversion for LVCSR," *Eurospeech 2003*, pp. 933-936, Geneva, Switzerland, 2003.

[Bisani, 2008] Bisani, M., Ney, H., "Joint-sequence models for grapheme-to-phoneme conversion," *Speech Communications*, vol. 50, no. 5, pp. 434–451, 2008.

[Bonaventura, 1998] Bonaventura, P., Giuliani, F., Garrido, J.M., Ortin, I., "Grapheme-to-phoneme transcription rules for Spanish, with application to automatic speech recognition and synthesis," *ACL 1998*, 1998.

[Burileanu, 1983] Burileanu, C., Teodorescu, V., Stolojanu, G., Radu, C., "Sistem cu logică programată pentru recunoaşterea cuvintelor izolate, independent de vorbitor," *Artificial intelligence and robotics*, Romanian Academy Publishing House, Bucharest, vol. 1, pp.266-274, 1983.

[Burileanu, 1998] Burileanu, D., Sima, M., Burileanu, C., Croitoru, V., "A Neuronal Network-Based Speaker-Independent System for Word Recognition in Romanian Language," *TSD 1998*, pp. 177-182, 1998.

[Burileanu, 1999] Burileanu, D., Sima, M., Neagu, A., "A phonetic converter for speech synthesis in Romanian," *ICPhS 1999*, vol. 1, pp. 503-506, San Francisco, 1999.

[Burileanu, 2003] Burileanu, D., Dervis, A., "An Efficient Keyword Identification Method for Continuous Speech," *Modern Technologies in the XXI Century*, Bucharest, pp. 195-201, Nov. 2003.

[Burileanu, 2004] Burileanu, C., Popescu, V., "An Efficient Distributed Speech Recognition Front-End Implementation Using a Motorola Star Core 140-Based Platform," *Politehnica University of Timisoara Scientific Bulletin - Transactions on Electronics and Communications*, vol. 49(63-1), pp 305-310, 2004.

[Burileanu, 2008] Burileanu, C., Petrea, C.-S., Buzo, A., Cucu H., Paşca, A., "Report on building a tool for Romanian spontaneous speech recognition," *The Phonetician*, No. 97 / 2008-I-II, pp. 68-98, 2008.

[Burileanu, 2010a] Burileanu, C., Buzo, A., Petre, C.S., Ghelmez-Hanes, D., Cucu, H., "Romanian Spoken Language Resources and Annotation for Speaker Independent Spontaneous Speech Recognition," *ICDT 2010*, pp.7-10, Athens, Greece, 2010.

[Burileanu, 2010b] Burileanu, C., Petrea, C.-S., Buzo, A., Cucu H., "Speech Recognition Experiments Starting from Isolated Words for Spoken Romanian Language," *Multilinguality and Interoperability in Language Processing with Emphasis on Romanian*, Romanian Academy Publishing House, Bucharest, pp. 231-244, 2010.

[Buzo, 2011a] Buzo, A., Cucu, H., Burileanu, C., Paşca, M., Popescu, V., "Word error rate improvement and complexity reduction in automatic speech recognition by analyzing acoustic model uncertainty and confusion," *SpeD 2011*, pp. 67-74, 2011.

[Buzo, 2011b] Buzo, A., Cucu, H., Burileanu, C., "Improving automatic speech recognition robustness for the romanian language," *EUSIPCO 2011*, pp 2119-2122, 2011.

[Camara, 2007] Camara, M., *La reconnaissance automatique de la parole continue pour la langue roumaine*, Rapport de diplôme, Université Joseph Fourier de Grenoble et Université Polytechnique de Bucarest, 2007.

[Chen, 1998] Chen, S.F., Goodman, J., "An empirical study of smoothing techniques for language modeling," *Technical Report TR-10-98*, Computer Science Group, Harvard University, 1998.

[Ciucă, 2010] Ciucă, Ş., Vlad, A., Mitrea, A., "A Comparison Between Two Literary Printed Romanian Corpora Based on the Statistical Letter Structure with Orthography and Punctuation Marks," *Communications 2010*, pp. 119-122, 2010.

[Cristea, 2006] Cristea, D., Forăscu, C., "Linguistic Resources and Technologies for Romanian Language", *Journal of Computer Science of Moldova*, vol. 14, no. 1, pp. 33-73, 2006.

[Cucu, 2011a] Cucu, H., Buzo, A., Burileanu, C., "Optimization methods for large vocabulary, isolated words recognition in Romanian language", *University Politehnica of Bucharest Scientific Bulletin*, Series C, no. 2, pp. 179-192, 2011.

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

[Cucu, 2011c] Cucu, H., Besacier, L., Burileanu, C., Buzo, A., "Investigating the Role of Machine Translated Text in ASR Domain Adaptation: Unsupervised and Semi-supervised Methods," *ASRU 2011*, to appear 2011.

[Davis, 1980] Davis, S., Mermelstein, P., "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 28, no. 4, pp. 357–366, 1980.

[Dempster, 1977] Dempster, A., Laird, N., Rubin, D., "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society*, Series B, vol. 39, pp. 1-38, 1977.

[Diac] Diac+ Diacritics restoration system (http://www.racai.ro/diac).

[Domokoş, 2009] Domokoş, J., "Contributions on Continuous Speech Recognition and Natural Language Processing," PhD Thesis, Technical University of Cluj-Napoca, 2009.

[Domokoş, 2011] Domokoş, J., "Automated Grapheme-to-Phoneme Conversion System for Romanian", *SPED 2011*, pp. 1-6, Braşov, Romania, 2011.

[Drăgănescu, 1986] Drăgănescu, M., Burileanu, C. "Analiza şi sinteza semnalului vocal", Romanian Academy Publishing House, Bucharest, 1986.

[Dumitru, 2008] Dumitru, C.-O., Gavăt, I., "Progress in Speech Recognition for Romanian Language," *Advances in Robotics, Automation and Control*, pp 472, Vienna, Austria, Oct. 2008.

[Europarl] European Parliament Parallel Corpus (http://www.statmt.org/europarl).

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

[Gavăt, 2007] Gavăt, I., Dumitru, C.O., "ASR for Romanian Language," *IWSSIP 2007*, pp. 300-303, Maribor, 2007.

[Giurgiu, 2011] Giurgiu, M., Kabir, A., "Comparison of Vocal Tract Length Normalization Technique Applied for Clean and Noisy Speech", *TSP 2011*, Budapest, 2011.

[Grigore, 1998] Grigore, O., Gavăt, I. and Zirra, M., “Neural network vowel recognition in Romanian language,” *CONTI 1998*, pp. 165-172, Timişoara, Romania.

[Hermansky, 1990] Hermansky, H., “Perceptual Linear Predictive (PLP) Analysis of Speech,” *Journal of the Acoustical Society of America*, vol. 87, no. 4, pp. 1738-1752, 1990.

[Hon, 1991] Hon, H.W., Lee, K.F., “CMU Robust Vocabulary-Independent Speech Recognition System,” *ICASSP 1991*, Toronto, pp. 889-892, 1991.

[Hotnews] Hotnews online newspaper (http://www.hotnews.ro).

[Huang, 2001] Huang, X., Acero, A., Wuen-Hon, H., *Spoken Language Processing – A Guide to Theory, Algorithm, and System Development*, Prentice Hall, 2001.

[Hwang, 1991] Hwang, M.Y., X.D. Huang, “Acoustic Classification of Phonetic Hidden Markov Models,” *Eurospeech 1991*, 1991.

[Hwang, 1993] Hwang, M.Y., Huang, X., Alleva, F., “Predicting Unseen Triphones with Senones,” *ICASSP 1993*, Minneapolis, pp. 311-314, 1993.

[Iordan, 2005] Iordan, I., *Romanian Language Orthographic, Orthoepic and Morphological Dictionary*, 2nd Edition, Romanian Academy Publishing House, 2005.

[Java] Java Development Kit (http://www.oracle.com/technetwork/java/javase/overview/index.html).

[Jelinek, 1998] Jelinek, F., *Statistical Methods for Speech Recognition*, The MIT Press, Cambridge, MA, 1998.

[JGraph] JGraph API (http://www.jgraph.com/jgraph.html)

[Jitcă, 2003] Jitcă D., Teodorescu H. N., Apopei V., Grigoraş F., “An ANN-based method to improve the phonetic transcription and prosody modules of a TTS system for the Romanian language,” *SpeD 2003*, pp. 43-50, 2003.

[Jurafsky, 2009] Jurafsky, D., Martin, J., “Automatic Speech Recognition,” *Speech and Language Processing: An introduction to natural language processing, computational linguistics, and speech recognition (2nd Ed.)*, Pearson Education, 2009.

[Kabir, 2011] Kabir, A., Giurgiu, M., “A Romanian Corpus for Speech Perception and Automatic Speech Recognition,” *The 10th International Conference on Signal Processing, Robotics and Automation*, Cambridge, UK, pp. 323-327, 2011.

[Karanasou, 2010] Karanasou P., Lamel, L., “Comparing SMT Methods for Automatic Generation of Pronunciation Variants,” *IceTAL 2010*, pp. 167, 2010.

[Kneser, 1995] Kneser, R., Ney, H., “Improved backing-off for m-gram language modeling,” *ICASSP 1995*, vol. 1, pp. 181-184, 1995.

[Knight, 1999] Knight, K., “Decoding complexity in word-replacement translation models,” *Computational Linguistics*, vol. 25, no. 4, pp. 607-615, 1999.

[Koehn, 2003] Koehn, P., Och, F.J., Marcu, D., “Statistical phrase based translation,” *HLT-NAACL 2003*, 2003.

[Koehn, 2005] Koehn P., “Europarl: A Parallel Corpus for Statistical Machine Translation,” *MT Summit 2005*, pp. 79-86, Phuket, Thailand, 2005.

[Koehn, 2007] Koehn, P., et al., “Moses: Open Source Toolkit for Statistical Machine Translation,” *ACL 2007*, Prague, Czech Republic, 2007.

[Koehn, 2010] Koehn, P., *Statistical Machine Translation*, Cambridge University Press, 2010.

[Laurent, 2009] Laurent, A., Deléglise, P., Meignier, S., “Grapheme to phoneme conversion using an SMT system,” *Interspeech 2009*, pp. 708, 2009

[Lee, 1988] Lee, K.F., *Large-Vocabulary Speaker-Independent Continuous Speech Recognition: The SPHINX System*, PhD Thesis, Computer Science Dept., Carnegie Mellon University, Pittsburgh, 1988.

[Liță, 2003] Liță, L., Ittycheriah, A., Roukos, S., Kambhatla, N., tRuEcasIng, *ACL 2003*, pp.152-159, Sapporo, Japan, 2003.

[Liu, 1999] Liu, Y., Fung, P.N., “Decision tree-based triphones are robust and practical for Mandarian speech recognition,” *Eurospeech 1999*, pp. 895-898, 1999.

[Lynx] Lynx browser (http://lynx.browser.org/).

[Macoveiciuc, 2010] Macoveiciuc, M., Kilgarriff, A., “The RoWaC Corpus and Romanian Word Sketches,” *Multilinguality and Interoperability in Language Processing with Emphasis on Romanian*, Romanian Academy Publishing House, Bucharest, ISBN: 978-973-27-1972-5, pp. 151-168, 2010.

[Mareuil, 1999] de Mareuil, P.B., Corredor-Ardoy, C., Adda-Decker, M., “Multi-Lingual Automatic Phoneme Clustering,” *ICPhS 1999*, pp. 1209-1212, San Francisco, USA, 1999.

[Mihalcea, 2002] Mihalcea, R., Nastase, V., “Letter Level Learning for Language Independent Diacritics Restoration,” *CoNLL 2002*, Taipei, Taiwan, pp. 105-111, 2002.

[Militaru, 2009] Militaru, D., Gavăt, I., Dumitru, O., Zaharia, T., Segărceanu, S., “Protologos, System for Romanian Language Automatic Speech Recognition and Understanding,” *SpeD 2009*, pp. 21 – 32, Constanța, Romania, 2009.

[Munteanu, 2006] Munteanu, D.-P., *Contribuții la elaborarea metodelor de recunoaştere a vorbirii în limba română*, PhD Thesis, Academia Tehnică Militară, Bucharest, 2006.

[Munteanu, 2008] Munteanu, D.-P., Vizitiu, C.I., “Robust Romanian Language Automatic Speech Recognizer Based on Multistyle Training,” *WSEAS Transactions on Computer Research*, vol. 3, no. 2, 2008.

[NIST, 2005] NIST Speech recognition scoring toolkit (sctk) version 2.1 (http://www.nist.gov/speech/tools/).

[Oancea, 2004] Oancea, E., Gavăt, I., Dumitru, O., Munteanu, D., “Continuous Speech Recognition for Romanian Language Based on Context Dependent Modeling,” *Communications 2004*, Bucharest, pp. 221-224, 2004.

[Och, 2003] Och, F.J., Ney, H., “A systematic comparison of various statistical alignment models,” *Computational Linguistics*, vol. 29, no. 1, pp. 19–52, 2003.

[Ordean, 2009] Ordean, M.A., Saupe, A., Ordean, M., Duma, M., Silaghi, G.C., “Enhanced rule-based phonetic transcription for the Romanian language,” *SYNASC 2009*, pp. 401-406, Timişoara, Romania.

[Papineni, 2002] Papineni, K., Roukos, S., Ward, T., Zhu, W.-J., “BLEU: a method for automatic evaluation of machine translation,” *ACL 2002*, pp. 311-318, Philadelphia, USA, 2002.

[Petrea, 2008] Petrea, C., Ghelmez-Hanes, D., Popescu, V., Buzo, A., Burileanu,, C., “Spontaneus Speech Database for Romanian Language,” *ECIT 2008*, Iaşi, Romania, 2008.

[Petrea, 2010] Petrea, C.S.. Buzo, A., Cucu, H., Paşca, M., Burileanu, C., “Speech Recognition Experimental Results for Romanian Language,” *ECIT 2010*, pp. 13, Iaşi, Romania, 2010.

[Poritz, 1988] Poritz, A., “Hidden Markov models: a guided tour,” *ICASSP 1988*, pp. 7–13.

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

[Renalds, 2010] Renals S., Hain, T., “Speech Recognition,” *Handbook of Computational Linguistics and Natural Language Processing*, 2010.

[Sabac, 1998] Sabac, B., Valsan, Z., Gavăt, I., “Isolated Word Recognition Using the DTW Algorithm,” *Communications 1998*, pp. 245 – 251, 1998.

[Sam, 2010] Sam, S., Castelli, E., Besacier. L., “Unsupervised Acoustic Model Adaptation for Multi-Origin Non Native ASR,” *Interspeech 2010*, pp. 254-257, Tokyo, Japan, 2010.

[Samudravijaya, 2003] Samudravijaya, K., Barot, M., “A comparison of public-domain software tools for speech recognition,” *WSLP 2003*, pp. 125-131, Mumbai, India, 2003.

[Sketch] The Sketch Engine (http://www.sketchengine.co.uk).

[Steward, 2002] Stewart, D., Ji, M., Ming, H., Philip, S., Jack, F., “A state-tying approach to building syllable HMMs,” *ICSLP 2002*, pp. 2649-2652, 2002.

[Stolcke, 2002] Stolcke, A., “SRILM - an extensible language modeling toolkit,” *ICSLP 2002*, pp. 901-904, Colorado, USA, 2002

[Suenderman, 2009] Suenderman, K., Liscombe, J., “Localization of speech recognition in spoken dialog systems: How machine translation can make our lives,” *Interspeech 2009*, pp. 1475-1478, 2009.

[Tan, 2007] Tan, T.-P., Besacier. L., “Acoustic Model Interpolation for Non-Native Speech Recognition,” *ICASSP 2007*, pp. 1009-1012, Honolulu, USA, 2007.

[Tan, 2008] Tan, T.-P., *Automatic Speech Recognition for Non-Native Speakers*. PhD Thesis, University Joseph Fourier, Grenoble, France, 2008.

[Teodorescu, 2009] Teodorescu, H.-N., Pistol, L., Feraru, M., Zbancioc, M., Trandabăț, D., *Sounds of the Romanian Language Corpus*, http://www.etc.tuiasi.ro/sibm/romanian_spoken_language/index.htm, 2009.

[Toma, 2009] Toma Ş.-A., Munteanu, D.-P., "Rule-Based Automatic Phonetic Transcription for the Romanian Language", *COMPUTATIONWORLD 2009*, pp. 682-686, 2009.

[Tufiş, 1999] Tufiş, D., Chițu, A., "Automatic Insertion of Diacritics in Romanian Texts," *International Workshop on Computational Lexicography 1999*, pp. 185-194, Pecs, Hungary, 1999.

[Tufiş, 2006] Tufiş, D., Irimia, E., "RoCo News – A hand validated journalistic corpus of Romanian," *LREC 2006*, pp. 869-872, Genoa, Italy, 2006.

[Tufiş, 2008] Tufiş, D., Ceauşu. D., "DIAC+: A Professional Diacritics Recovering System," *LREC 2008*, Marrakech, Morocco, 2008.

[Ungurean, 2008] Ungurean, C., Burileanu, D., Popescu, V., Negrescu, C., Dervis, A., "Automatic Diacritics Restoration for a TTS-based E-mail Reader Application," *University Politehnica of Bucharest Scientific Bulletin*, Series C, vol. 70, no. 4, Bucharest, Romania, 2008.

[Ungurean, 2011] Ungurean, C., Burileanu, D., "An advanced NLP framework for high-quality Text-to-Speech synthesis," *SpeD 2011*, Braşov, Romania, 2011.

[Valsan, 1998a] Valsan, Z., Sabac, B., Gavăt, I., Zamfirescu, D., "Combining self-organizing map and multilayer perceptron in a neural system for improved isolated word recognition," *Communications 1998*, pp. 245-251, Bucharest, Romania, 1998.

[Valsan, 1998b] Valsan, Z., Sabac, B., Gavăt, I., "Combining self organizing feature map and multilayer perceptron in a neural system for fast key-word spotting," *SPECOM 1998*, pp. 303-308, St. Petersburg, Russia, 1998.

[Vlad, 2007] Vlad, A., Mitrea, A., Mitrea, M., "Printed Romanian Modeling: A Corpus Linguistic Based Study With Orthography And Punctuation Marks Included," *Computational Science and It's Applications – ICCSA 2007*, Lecture Notes in Computer Science, vol. 4705, Springer Verlag, Berlin Heidelberg, pp. 409-423, 2007.

[Vlad, 2010] Vlad, A., Mitrea, A., Mitrea, M., Ciucă, Ş., "Enriching printed Romanian statistical description", *Multilinguality and Interoperability in Language Processing with Emphasis on Romanian*, Romanian Academy Publishing House, Bucharest, ISBN: 978-973-27-1972-5, pp. 245-271, 2010.

[Witten, 1991] Witten, I.H., Bell, T.C., "The zero-frequency problem: estimating the probabilities of novelevents in adaptive text compression," *IEEE Transactions on Information Theory*, vol. 37, no. 4, pp. 1085–1094, 1991.

[Young, 1989] Young, S.J., Russell, N.H., Thornton, J.H.S., "Token Passing: a Simple Conceptual Model for Connected Speech Recognition Systems," 1989.

[Young, 1993] Young, S.J., Woodland, P.C., "The Use of State Tying in Continuous Speech Recognition," *Eurospeech 1993*, pp. 2203-2206, Berlin, 1993.

[Young, 2005] Young, S., Evermann, G., Kershaw, D., Moore, G., Odell, J., Ollason, D., Povey, D., Valtchev, V., Woodland P., *The HTK Book*, Cambridge University, United Kingdom, 2005.

[Zens, 2002] Zens, R., Och, F.J., Ney, H., "Phrase-based statistical machine translation," *German Conference on Artificial Intelligence (KI 2002)*, 2002