Universität Karlsruhe

Carnegie-Mellon University

Bulgarian Speech Recognition and Multilingual Language Modeling

von Aneliya Mircheva

Studienarbeit

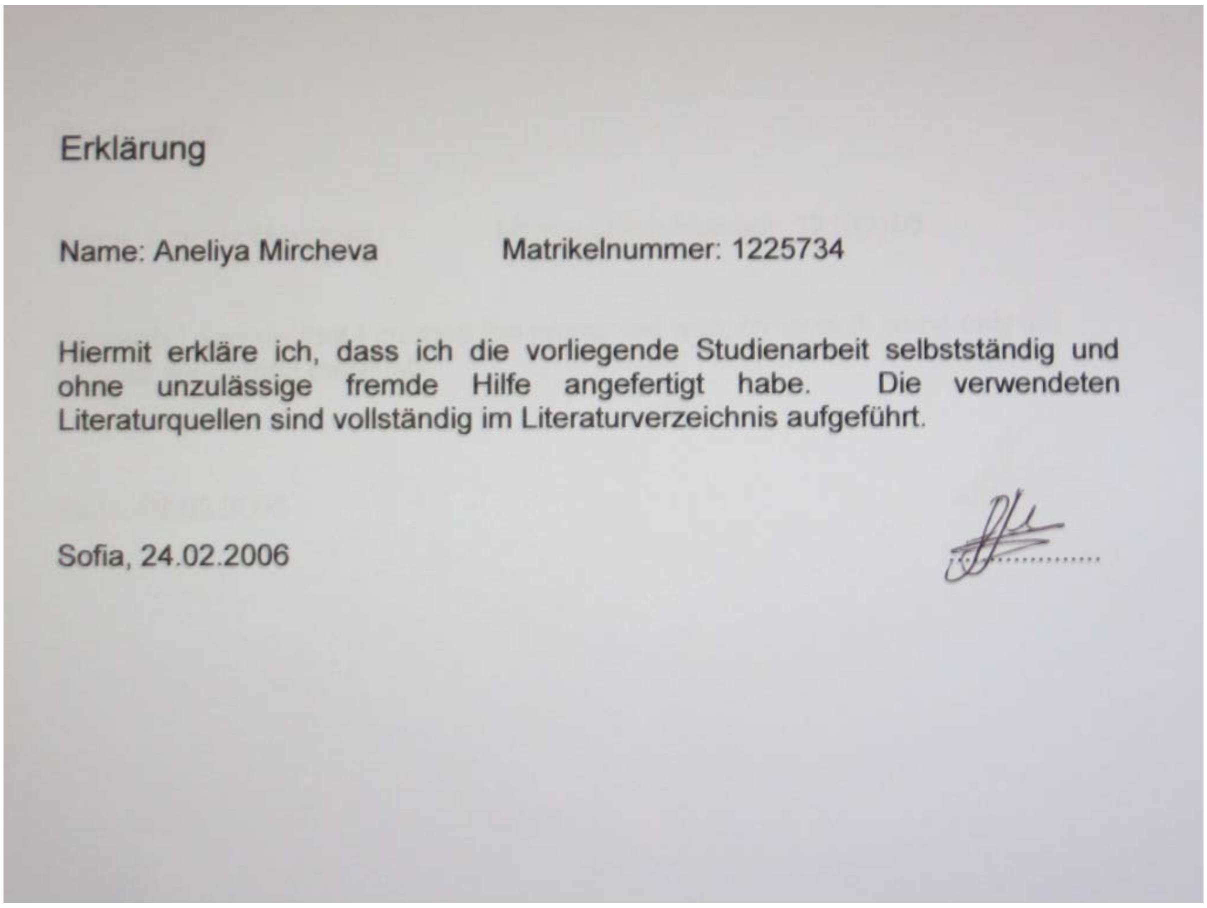
am Institut für Theoretische Informatik
Prof. Dr. Alex Waibel

Fakultät für Informatik, Universität Karlsruhe

Betreuer:

Dr. Tanja Schultz
Dipl.-Inf. Matthias Paulik

Tag der Anmeldung:	23.12.2005
Tag der Abgabe:	21.03.2006



Erklärung

Name: Aneliya Mircheva Matrikelnummer: 1225734

Hiermit erkläre ich, dass ich die vorliegende Studienarbeit selbstständig und ohne unzulässige fremde Hilfe angefertigt habe. Die verwendeten Literaturquellen sind vollständig im Literaturverzeichnis aufgeführt.

Sofia, 24.02.2006

[illegible]

CONTENT

List of Figures

List of Tables

1. Introduction

The following report describes the work at the interAct on Bulgarian speech recognition, including the collection of data, training a Bulgarian speech recognizer and experimenting with Russian text data to improve the recognition. It also gives an overview of the unique traits of Bulgarian language, introduces the main issues of speech recognition, and presents the results.

1.1 Tasks and Results – an Overview

The field of speech recognition aims to convert speech signals into written word sequences. There are many applications where speech recognition cannot only be helpful but is also necessary. Different implementations of speech recognizers are used increasingly around the world. Examples are dictation systems, translation systems, information retrieval systems, telephone services, control devices, identification systems, learning programs and many more. In short, recognizers are used whenever communication is involved.

The large number of applications indicates even larger variability in the definition of the speech recognition task. We may want a system to work well not only with command words, but also with continuous speech; not only with read speech sequences but also with spontaneous speech. Another example is that we may not be happy with a system that works only with vocabulary for a specific task or only in a quiet environment, but we may want it to work with large vocabulary or in a noisy environment. One of the major challenges of speech recognition is the variability of speech: two identical utterances, spoken by the same person may result in two different speech signals. An important issue is the ability of the system to recognize accurately even if there are some bigger differences in the pronunciation like dialects or disabilities.

Different chapters of this work describe the characteristics of Bulgarian language, how text and speech data have been collected and preprocessed and how a Bulgarian recognizer for continuous speech has been initialized and trained. Language modeling for the created recognizer is reviewed in detail and its performance is evaluated. About 21.38 hours (1283.12 minutes) of speech data is collected from 77 native speakers in Bulgaria. The speakers are reading text, which has been collected from national and international economical and political articles from online newspapers. The data is later divided into training, development, and evaluation data for different purposes. The training set is used to build and train the acoustic model of the recognizer, the development set to adjust the parameters for the decoding phase, and the evaluation set is used to test the performance of the final system. Word Error Rate (WER) measures how many percent of the words in a sentence are falsely recognized, and is used as a measure for the performance of the recognizer. The best average WER for case sensitive Bulgarian recognition on the development data is 24.84% and on the evaluation data is 26.57%.

A studied topic is how can text written in a language close to Bulgarian improve the language modeling for Bulgarian and thus improve the speech recognition, too. This research addresses mostly languages that have very few available data. For such languages, it is difficult to build a stable recognizer and perform good recognition. This report presents results of experiments, where language models are created via interpolation between Bulgarian and Russian models.

1.2 Janus

Janus [WA-W$^+$94] is developed at the Interactive System Laboratories at Carnegie Mellon University and the University of Karlsruhe since the late '80s and can be described as a programmable system, which consists of several modules – the main modules are speech recognition, parsing and discourse analysis. The programming language of Janus [WA-W$^+$94] is C and its shell is programmed in Tcl/Tk. The shell is programmed to use also some additional specific object classes (for example dictionary) and their methods.

To build the Bulgarian recognizer, the speech recognition module of the speech-to-speech translation system Janus [WA-W$^+$94] is used. It is based on the Janus Recognition Toolkit (JRTk) [FGH$^+$97]. The speech recognition module is language independent and allows loading and working with texts and speech in different languages. This makes it easy to adapt the system to new domains. Recognition with Janus [WA-W$^+$94] can already be performed for many languages (English, Japanese, German, Spanish, Korean, Chinese, Russian, Mandarin, and Thai). Translation with Janus [WLL97] is performed into German, English, Spanish, Japanese, Korean, Chinese, Arabic, Thai, Egyptian, Hindi, and French.

2. The Bulgarian Language

In this chapter, we introduce into the Bulgarian language - from its distribution over its phonetic, up to its morphology and lexis. We will also explain which characteristics of the language affect Bulgarian speech recognition. Most of the linguistic information in this chapter is citation of the available linguistic information in Wikipedia [Wiki06], the free encyclopedia.

2.1 Distribution, Affiliations, and General Characteristics

The Bulgarian language is a member of the Indo-European family of languages. This family includes more than 443 estimated languages [CC93], and contains the Slavic languages, the Romance languages (French, Italian, Spanish, Portuguese, and Romanian), the Germanic languages (German, English, Swedish, Norwegian, Dutch, Danish), and others, such as Greek, Albanian, Armenian, Sanskrit, Persian and so on. The group of the Slavic languages is distributed over more than 250 million people in Eastern and Central Europe, most of the Balkan Peninsula, and in northern Asia (see its distribution in Figure 1) and it is divided into three branches as shown in the following list.

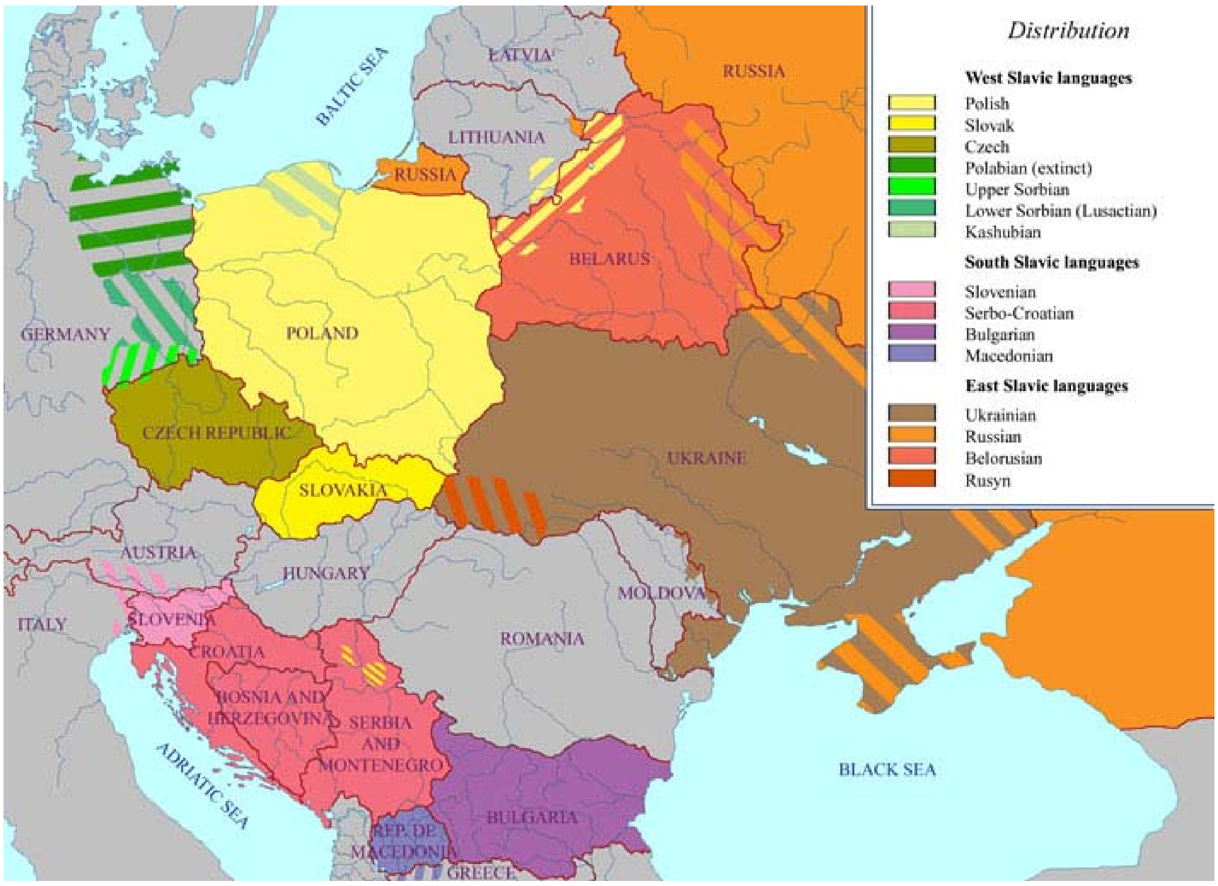
The group of Slavic languages [CC93]:

- **East Slavic:** Russian, Ukrainian, Belarusian and Rusyn;

- **South Slavic** is subdivided into:
  - eastern group - Bulgarian and Macedonian;
  - western group - Slovenian, Serbian, Croatian, and Bosnian (the last three languages in this group are often combined by Slavicists together into the Serbo-Croatian language);

- **West Slavic** is subdivided into:
  - Czech and Slovak;
  - Upper and Lower Sorbian;
  - Polish, Pomeranian/ Kashubian and extinct Polabian.

Figure 1 Distribution of all Slavic languages in Europe and in northern Asia [Wiki06]



As displayed in the list and in Figure 1, Bulgarian is a Southern Slavic language. Its distribution is estimated to be about 12 millions speakers mainly in Bulgaria, but also in Ukraine, Macedonia, Serbia, Turkey, Greece, Romania, Canada, USA, Australia, Germany, and Spain [Omn05].

Bulgarian is part of the Balkan linguistic union [Wiki06], which includes the mutually intelligible Macedonian language and the closely related Serbian, Romanian, and Albanian. Most of these languages share some characteristics, like definite article (the form of a suffix joined to the noun or to its adjective), the lack of a verb infinitive and a complicated verb system as result of further development of the proto-Slavic verb system (there are various verb forms to express non-witnessed, retold, and doubtful action).

These characteristics also set the language apart from other Slavic languages like Slovenian, Ukrainian, and Russian. Another difference between Bulgarian and most other Slavic languages is that Bulgarian has almost completely dropped the numerous case forms of the noun. It uses position and prepositions (like English) to indicate grammatical relationships in a sentence instead of cases (like Russian). The Bulgarian language lacks definite rules for stress (just like all other Slavic languages except the West Slavic languages and Macedonian). Therefore, the accent of every word must be learned individually. Despite these differences, Bulgarian closely resembles the other Slavic languages, especially with regard to grammar [Wiki06].

2.2 History

The development of the Bulgarian language can be divided into several historical periods [CC93]. The prehistoric period (essentially proto-Slavic) occurred between the Slavonic invasion of the eastern Balkans and the mission of St. Cyril and St. Methodius to Great Moravia in the 860s. Old Bulgarian (9th to 11th century, also referred to as Old Church Slavonic) was the language used by St. Cyril, St. Methodius and their disciples to translate the Bible and other liturgical literature from Greek.

Middle Bulgarian (12th to 15th century) was a language of rich literary activity and major innovations. During the Middle Bulgarian period, the language underwent dramatic changes, losing the Old Slavonic case system, but preserving the rich verb system (while the development was exactly the opposite in most other Slavic languages) and developing a definite article.

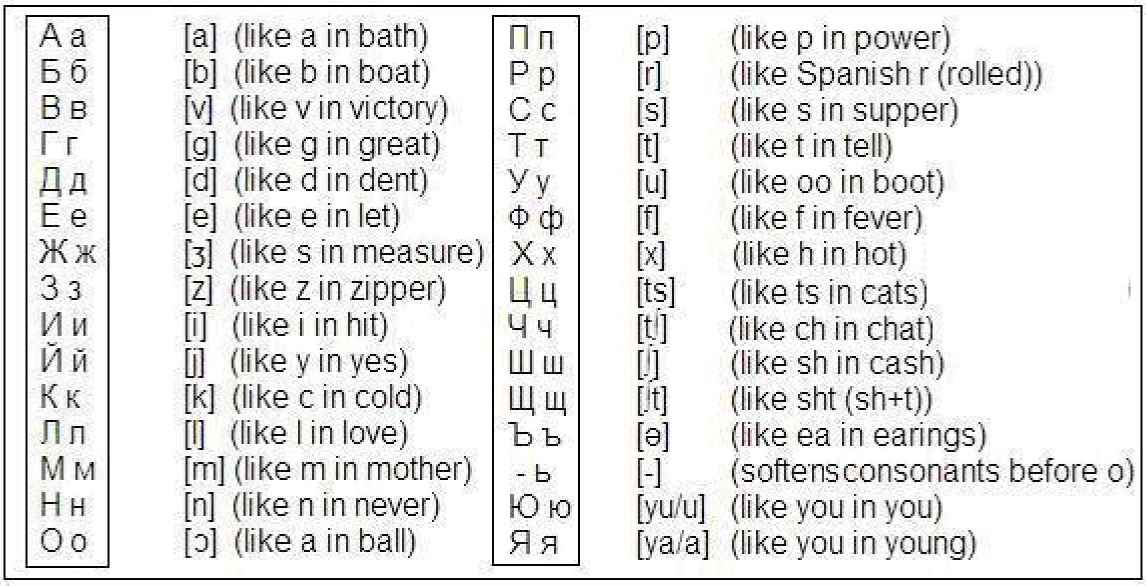
The Modern Bulgarian Period started in the 15th century, but the modern literary language, which is quite different from Old Bulgarian, formed only during the 19th century. One of the main changes Bulgarian underwent in its sound system and in the number of letters in its alphabet

[Wiki06]. You can find information about the Bulgarian alphabet and script in the next section of this chapter.

2.3 Alphabet

The first alphabet used was the 'Glagolitic' alphabet [Wiki06]. The brothers' disciples created the 'Cyrillic' alphabet, from the name of Cyril. The Cyrillic alphabet is mainly based on the Greek alphabet, supplemented by new graphemes to render special Slavonic phonemes. The original Cyrillic alphabet has contained 44 letters for 44 sounds but after the dramatically change by the 19th century of the Bulgarian sounds system, which started using fewer sounds, the number of letters used has been reduced from 44 to 32. The alphabet used after the reform in 1945 contained the same 32 letters as the previous one with the exception of ѣ (called "double e") and ѫ (called "yus"). Thus, the modern Bulgarian alphabet has 30 letters [Wiki06]. Figure 2 shows each grapheme of the Bulgarian language, its pronunciation, and an example (for English speakers).

Figure 2 Alphabet and pronunciation of the letters with English examples [FAQ06]



А а	[a]	(like a in bath)	П п	[p]	(like p in power)
Б б	[b]	(like b in boat)	Р р	[r]	(like Spanish r (rolled))
В в	[v]	(like v in victory)	С с	[s]	(like s in supper)
Г г	[g]	(like g in great)	Т т	[t]	(like t in tell)
Д д	[d]	(like d in dent)	У у	[u]	(like oo in boot)
Е е	[e]	(like e in let)	Ф ф	[f]	(like f in fever)
Ж ж	[ʒ]	(like s in measure)	Х х	[x]	(like h in hot)
З з	[z]	(like z in zipper)	Ц ц	[ts]	(like ts in cats)
И и	[i]	(like i in hit)	Ч ч	[tʃ]	(like ch in chat)
Й й	[j]	(like y in yes)	Ш ш	[ʃ]	(like sh in cash)
К к	[k]	(like c in cold)	Щ щ	[ʃt]	(like sht (sh+t))
Л л	[l]	(like l in love)	Ъ ъ	[ə]	(like ea in earings)
М м	[m]	(like m in mother)	- ь	[-]	(softens consonants before o)
Н н	[n]	(like n in never)	Ю ю	[yu/u]	(like you in you)
О о	[ɔ]	(like a in ball)	Я я	[ya/a]	(like you in young)

Notes to Figure 2:

ю (yu) = [u] after a palatalized consonant and я (ya) = [a] after a palatalized consonant (see section 2.4 of this chapter).

2.3.1 Script Encoding

There are many possible encodings for representing the Bulgarian alphabet in computers. Maybe the most widely used are the UTF-8 and the Unicode encoding systems. In the presented work, we will make use of the UTF-8 encoding system, which fortunately is well supported by the Janus Speech Recognition Toolkit [FGH$^+$97].

2.4 Phonetics and IPA

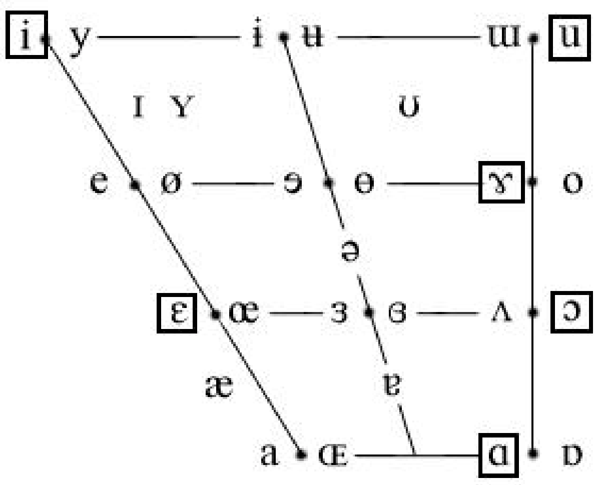
Most letters in the Bulgarian alphabet stand for one specific sound and that sound only [Mad84]. Three letters stand for the single expression of combinations of sounds, namely щ [St], ю [yu] and я [ya]. Two sounds do not have separate letters assigned to them and are expressed by the combination of two letters, namely дж [dZ] and дз [dz] (in this chapter, anything written in [and] is a phoneme according to the grapheme-to-phoneme mapping table in Figure 5 in subsection 2.4.4).

In the Bulgarian spoken language, many of the consonants can be softened depending on the next letter in the word. The resulting palatalized consonants are considered as separate phonemes and will be described later in this section. The phoneme set we were using for building the Bulgarian speech recognizer consists of 45 phonemes - 6 vowels, 1 semivowel, 19 hard consonants, 15 palatalized consonants, 2 diphthongs, and 2 additional consonants. In this paper and in our work on building the recognizer, we will use a letter or a pair of letters to notate each of these phonemes. Detailed descriptions of the phonemes and their corresponding notation with Roman letters can be found in subsection 2.4.4.

2.4.1 Vowels

Figure 3 illustrates the set of IPA phonemes used to represent the six vowels in Bulgarian. All vowels are relatively lax, as in most other Slavic languages, and unlike the tense vowels as, for example, in the Germanic languages.

Figure 3 IPA table for vowels [Mad84]



<u>Notes to Figure 3:</u>

You can find IPA sets for Bulgarian, where instead of the open back ‘ɑ’, the open central sound ‘a’ is used for the phoneme [a].

When the vowels are unstressed, they tend to be shorter, and weaker compared to when they are stressed, the corresponding pairs of open and closed vowels approaching each other with a tendency to merge, although the coalescence is not always complete [Wiki06]. The variation of the norm seems to be socially conditioned: on the one hand, the relative absence of reduction is intuitively associated with certain types of low-status (provincial, especially West Bulgarian, or Romani-influenced) speech. On the other hand, the awareness of the distinctions is naturally perceived as a sign of literacy and education. The merger is, at least in non-dialectal pronunciation, totally accomplished for [a] and [Y] in all positions (except, occasionally and for some speakers, in a syllable immediately preceding another [a]). Unstressed [o] also tends to be pronounced like [u] (the difference is either minimal or nonexistent in pre-stress position and totally absent after stress), but the status of that coalescence is less clear, perhaps because post-stress [u] is not very common in the first place. The considerable reduction of [e] notwithstanding, similar coalescence of [e] and [i] is not allowed in formal speech and is definitely regarded as a provincial (East Bulgarian) feature; rather, unstressed and above all post-stress [e] might occasionally approach a more front form of [Y] [Wiki06].

2.4.2 Semivowels and Diphthongs

Bulgarian possesses one semivowel: [j], equivalent to the English <y> in yes. The [j] always immediately precedes or follows a vowel. The semivowel is most usually expressed graphically by the letter й, as, for example, in най [n a j] ("most"). There are two diphthongs [yu] and [ya] which represent the letters ю and я in the phoneme set and if one of them follows a consonant in the word, it palatalizes the consonant, and transforms itself respectively into the phoneme [u] or [a], for example бял (b ya l) [bj a l] ("white") [Wiki06]. In different papers, instead as [yu] and [ya], the diphthongs are notated as [ju] and [ja]. In this work, we will use the notation [yu] for ю and [ya] for я.

2.4.3 Consonants

In this project, we use 19 hard and 15 softened consonants, and 2 affricates. You can see all 36 of them and the 1 semivowel, in the IPA table in Figure 4. We only use the affricates [dz] and [dZ], which can be seen in our text data, and ignore other two affricates ([dzj] and [xj]), which occur very rarely, mostly in foreign names.

The Bulgarian consonants may be divided into pairs (voiced<>voiceless). The contrast 'voiced vs. voiceless' is neutralized in word-final position, where often the consonants are pronounced as voiceless (as in most Slavic languages, German, etc.); this neutralization is, however, not reflected in the spelling [Wiki06].

Hard and Palatalized Consonants

Some of the Bulgarian consonants ([b], [v], [g], [d], [z], [k], [l], [m], [n], [p], [r], [s], [t], [f], [ts]) can have both a normal, "hard" pronunciation, as well as a "soft", palatalized one. For the rest of the consonants ([S], [tS], [x] and [Z] no palatalizing is applied.

The softness of the palatalized consonants is indicated always in writing in Bulgarian. A consonant is palatalized if:

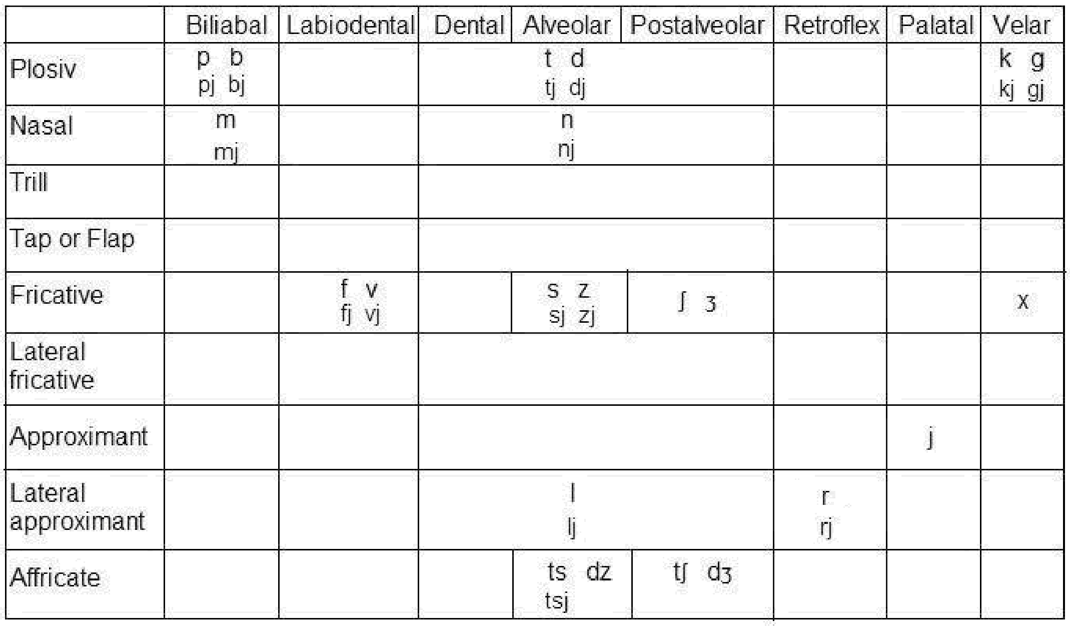
* it is followed by ь;

* it is followed by the letters я (ya) or ю (yu);

(the phonemes [ya] and [yu] are used in all other cases) [Wiki06]

The IPA table in Figure 4 shows all of the 19 hard consonants, the 15 palatalized consonants, the 2 affricates, and the 1 semivowel, used to represent the Bulgarian acoustic.

Figure 4 Consonants, affricates and semivowel [Mad84]



	Biliabal	Labiodental	Dental	Alveolar	Postalveolar	Retroflex	Palatal	Velar
Plosiv	p b pj bj			t d tj dj				k g kj gj
Nasal	m mj			n nj				
Trill								
Tap or Flap								
Fricative		f v fj vj		s z sj zj	ʃ ʒ			x
Lateral fricative								
Approximant							j	
Lateral approximant				l lj		r rj		
Affricate				ts dz tsj	tʃ dʒ			

2.4.4 Grapheme-to-phoneme Mapping

We perform grapheme-to-phoneme mapping for all of the 45 Bulgarian sounds. We define explicit rules, which:

- allow us to write down any Bulgarian (Cyrillic) word with Roman letters and convert it back into the same word written in Cyrillic any time we want to;

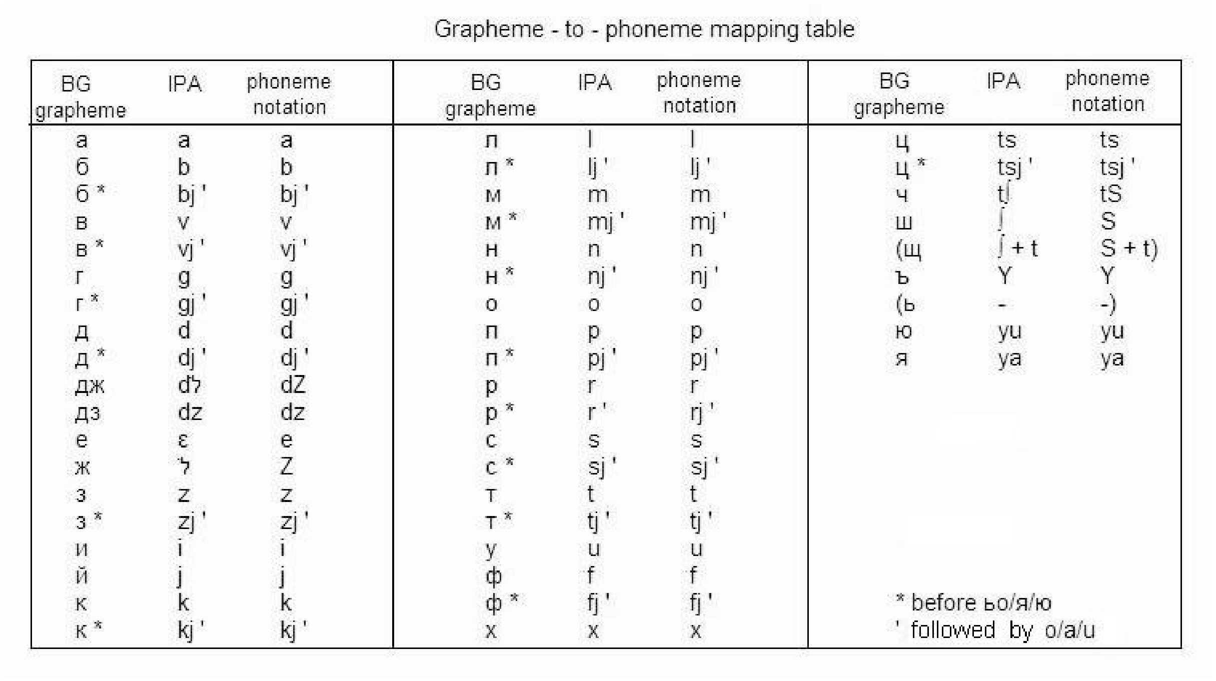
- map a specific IPA sound to each of the 45 phonemes we want to use for acoustic modeling;

- make it easy to find the corresponding match for a Bulgarian phoneme in the phoneme set of other languages, and even generate its pronunciation for further training of the Bulgarian acoustic model [Sch04];

- give us an easy notation form of the phonemes.

In Figure 5, you can find the letters and combinations of letters which we use to indicate the different phonemes. All phonemes are written with lower case letters except for 5 phonemes, which are or contain upper case letters ([Z], [dZ], [S], [tS] and [Y]). This guarantees consistency when converting to and from Roman letters. The notation will be used to represent phonemes in this report too. Any phonemes sequence in this report will be written in brackets [and].

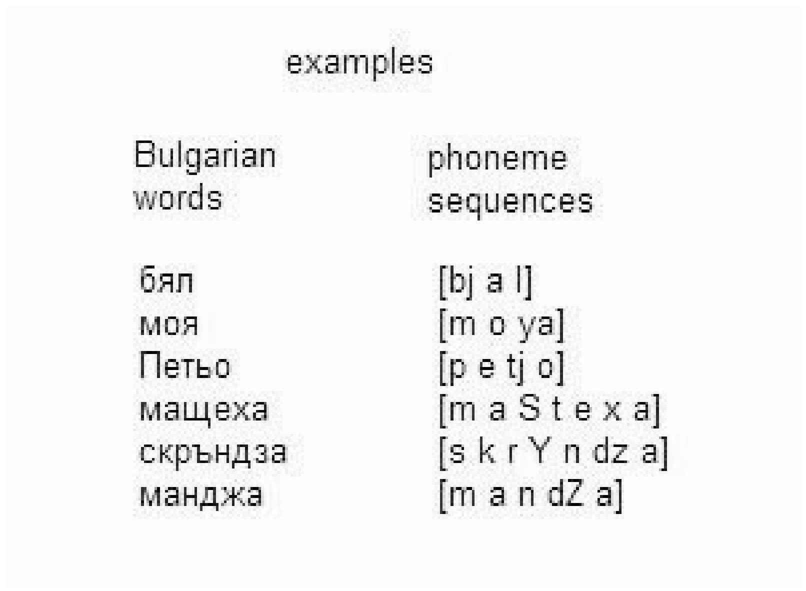
Figure 5 Grapheme-to-phoneme mapping (idea from [Wiki06])


Grapheme - to - phoneme mapping table

BG grapheme	IPA	phoneme notation	BG grapheme	IPA	phoneme notation	BG grapheme	IPA	phoneme notation
а	a	a	л	l	l	ц	ts	ts
б	b	b	л *	lj '	lj '	ц *	tsj '	tsj '
б *	bj '	bj '	м	m	m	ч	tʃ	tS
в	v	v	м *	mj '	mj '	ш	ʃ	S
в *	vj '	vj '	н	n	n	(щ	ʃ + t	S + t)
г	g	g	н *	nj '	nj '	ъ	Y	Y
г *	gj '	gj '	о	o	o	(ь	-	-)
д	d	d	п	p	p	ю	yu	yu
д *	dj '	dj '	п *	pj '	pj '	я	ya	ya
дж	dʒ	dZ	р	r	r			
дз	dz	dz	р *	r '	rj '			
е	ɛ	e	с	s	s			
ж	ʒ	Z	с *	sj '	sj '			
з	z	z	т	t	t			
з *	zj '	zj '	т *	tj '	tj '			
и	i	i	у	u	u			
й	j	j	ф	f	f			
к	k	k	ф *	fj '	fj '	* before ьо/я/ю		
к *	kj '	kj '	х	x	x	' followed by o/a/u		

There are several additional rules, which we apply during mapping the words to phoneme sequences. The letters [yu] and [ya] are mapped to the phonemes [yu] and [ya] if they do not follow a consonant in the word. If they follow a consonant, this consonant is replaced by its corresponding softened consonant and [yu] and [ya] themself are substituted respectively by the phonemes [u] and [a]. The letter "er malyk" softens the consonant it follows, and just like [yu] and [ya] after a consonant, this consonant is replaced by its corresponding softened one but this time the letter "er malyk" will not be substituted by any phoneme in the phoneme set. The letter (St) is not mapped to a separate phoneme in our work, but is divided into the phonemes [S] and [t]. One single phoneme [dz] or [dZ] results from each of the sequences of letters (d z) and (d Z). In Figure 6, you can find examples for each of these rules.

Figure 6 Mapping rules – examples (see Figure 5)



examples

Bulgarian words	phoneme sequences
бял	[bj a l]
моя	[m o ya]
Петьо	[p e tj o]
мащеха	[m a S t e x a]
скръндза	[s k r Y n dz a]
манджа	[m a n dZ a]

2.5 Morphology and Grammar

The parts of speech in Bulgarian are divided in 10 different types, which are categorized in two broad classes: mutable and immutable [Wiki06]. The difference is that mutable parts of speech vary grammatically, whereas the immutable ones do not change, regardless of their use. The five classes of mutable are nouns, adjectives, numerals, pronouns, and verbs. Syntactically, the first four of these form the group of the noun or the nominal group. The immutable are adverbs, prepositions, conjunctions, particles, and interjections. Verbs and adverbs form the group of the verb or the verbal group.

2.5.1 Nominal Morphology

Nouns, adjectives, and pronouns are inflected for grammatical gender, number, case (to a very limited extent), and definiteness in Bulgarian. Adjectives and adjectival pronouns agree with nouns in number and gender. See Figure 7 for examples.

Nominal Inflection

Gender

There are three grammatical genders in Bulgarian: masculine, feminine and neuter [Wiki06]. The gender of the noun can largely be determined according to its ending. The vast majority of Bulgarian nouns ending in a consonant (zero ending) are masculine (for example, град (city), син (son), мъж (man)). Feminine nouns include almost all nouns that have the endings (а/ я) (жена (woman), дъщеря (daughter)) and most nouns with zero ending expressing quality, degree or an abstraction, including all nouns ending on ост/ ест (мъдрост (wisdom), прелест (loveliness)). Another, much smaller group of feminine nouns is the one of irregular nouns with zero ending which define tangible objects or concepts (кръв (blood), кост (bone)). Nouns ending in е, о are almost exclusively neuter (дете (child), езеро (lake)). The same applies to a limited number of loan words ending in и, у, and ю (цунами (tsunami), меню (menu)).

Number

Two numbers are distinguished in Bulgarian - singular and plural [Wiki06]. The most typical plural ending for feminine nouns is и, which is appended to the word upon dropping the singular ending а/ я. Plural forms of neutral and masculine nouns use a variety of suffixes, the most typical of which are а/ я (both require dropping of the singular endings е/о) and та for neutral nouns and е, и and ове for masculine nouns. Exceptions, irregular declension and alternative plural forms are, however, very common for all three genders.

Case

Vestiges are well preserved only in the personal pronouns and the masculine personal interrogative pronoun кой (who), which have nominative, accusative and dative forms [Wiki06]. Vocative forms are still in use for masculine and feminine nouns (not for neuter), but endings in masculine nouns are determined solely according to the stem-final consonant of the noun. In other cases, the proto-Slavonic case system has been replaced by prepositional and other syntactic constructions.

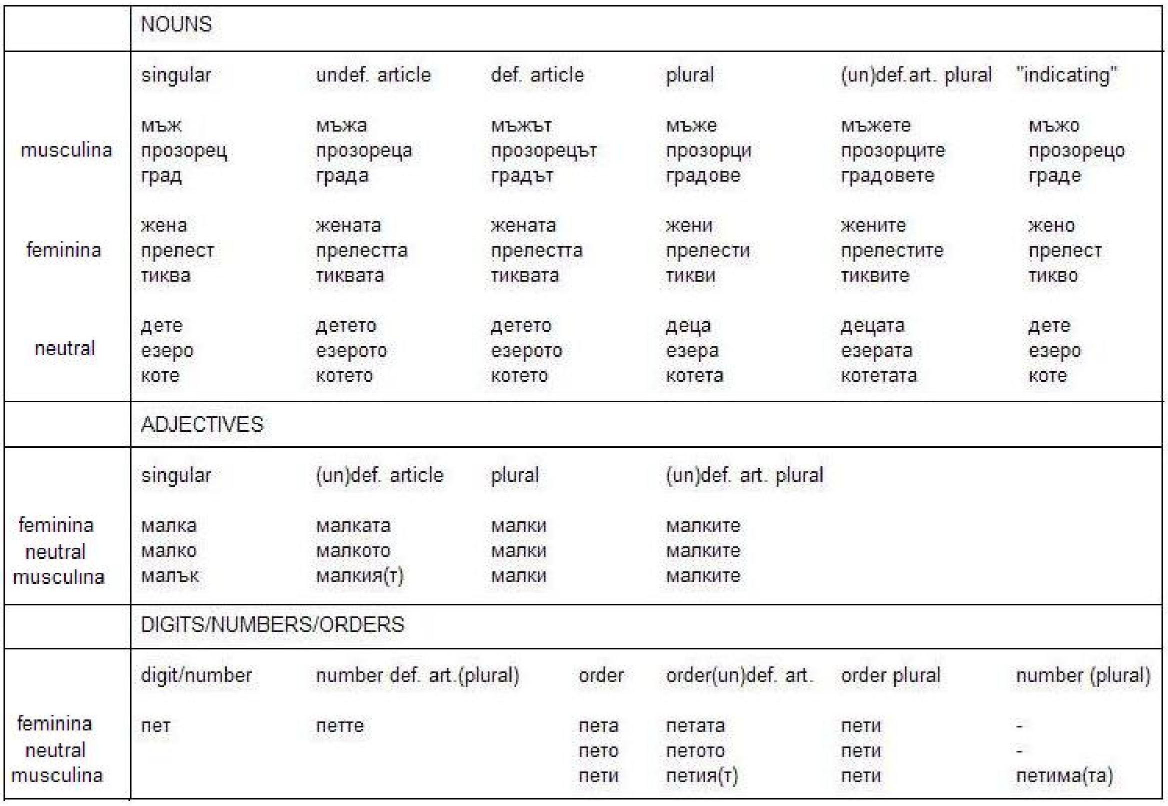
Definiteness

In modern Bulgarian, definiteness is expressed by a definite article which is postfixed to the noun (indefinite: човек, man; definite: човекът, the man) or the first nominal constituent of definite noun phrases (indefinite: добър човек, a good man; definite: добрият човек, the good man), much like in the Scandinavian languages or Romanian. There are four singular definite articles: ът/ ят for masculine nouns that are grammatical subjects, а/ я for masculine nouns that are grammatical objects, та for feminine nouns, and to for neuter nouns. The two masculine definite articles may also be considered as two grammatical forms of the same article. The plural definite articles are те for masculine and feminine nouns, and та for neuter nouns. When postfixed to adjectives the definite articles are ят/я for masculine, та for feminine, to for neuter and те for plural nouns [Wiki06].

2.5.2 Adjective and Numeral Inflection and Pronouns

Both adjective and numeral agree in gender and number with the noun they are appended to [Wiki06]. They may also take up the definite article as explained above. Pronouns may vary in gender, number, definiteness and are the only parts of the speech that have retained case inflections. Some groups of pronouns exhibit three cases: nominative, accusative, and dative, although dative is often substituted by accusative constructions. The distinguishable types of pronouns include the following: personal, relative, reflexive, interrogative, negative, indefinitive, summative, and possessive.

Figure 7 Nouns – examples



	NOUNS					
	singular	undef. article	def. article	plural	(un)def.art. plural	"indicating"
musculina	мъж	мъжа	мъжът	мъже	мъжете	мъжо
	прозорец	прозореца	прозорецът	прозорци	прозорците	прозорецо
	град	града	градът	градове	градовете	граде
feminina	жена	жената	жената	жени	жените	жено
	прелест	прелестта	прелестта	прелести	прелестите	прелест
	тиква	тиквата	тиквата	тикви	тиквите	тикво
neutral	дете	детето	детето	деца	децата	дете
	езеро	езерото	езерото	езера	езерата	езеро
	коте	котето	котето	котета	котетата	коте
	ADJECTIVES					
	singular	(un)def. article	plural	(un)def. art. plural		
feminina	малка	малката	малки	малките		
neutral	малко	малкото	малки	малките		
musculina	малък	малкия(т)	малки	малките		
	DIGITS/NUMBERS/ORDERS					

	digit/number	number def. art.(plural)	order	order(un)def. art.	order plural	number (plural)
feminina	пет	петте	пета	петата	пети	-
neutral			пето	петото	пети	-
musculina			пети	петия(т)	пети	петима(та)

2.5.3 Verbal Morphology and Grammar

Bulgarian adverbs coincide with the neuter singular form of the corresponding adjectives and are only syntactically distinguishable from the latter [Wiki06]. Verb forms, however, vary in aspect, mood, tense, person, number and sometimes gender and voice.

Finite Verbal Forms

Finite verbal forms are simple or compound and agree with subjects in person (first, second and third) and number (singular, plural) in Bulgarian. In addition to that, past compound forms using participles vary in gender (masculine, feminine, neuter) and voice (active and passive) as well as aspect (perfective/aorist and imperfective) [Wiki06].

Aspect

Bulgarian verbs express lexical aspect: perfective verbs signify the completion of the action of the verb and form past aorist tenses [Wiki06]. In Bulgarian, there is also grammatical aspect. Three grammatical aspects are distinguishable: neutral, perfect and pluperfect. The neutral aspect comprises the three simple tenses and the future tense. The pluperfect aspect is manifest in tenses that use double or triple auxiliary "be" participles like the past pluperfect subjunctive. Perfect tenses use a single auxiliary "be".

Mood

In addition to the four moods shared by most other European languages - indicative, imperative, subjunctive, and conditional - in Bulgarian there is one more to describe past unwitnessed events - the renarrative mood.

Tense

There are three grammatically distinctive positions in time present, past and future, which combine with aspect and mood to produce a number of formations. There are more than 30 different tenses across Bulgarian's two aspects and five moods.

2.6 Lexis

The native lexical terms in Bulgarian (both from proto-Slavonic and from the Bulgar language), account for 70% to 75% of the word-stock of the language [Wiki06]. The remaining 25% to 30% are loanwords from a number of languages, as well as derivations of such words. The languages which have contributed most to Bulgarian are Latin and Greek (mostly international terminology), and to a lesser extent French and Russian. The numerous loanwords from Turkish (and, via Turkish, from Arabic and Persian) which were adopted into Bulgarian during the long period of Ottoman rule have been substituted largely with native terms or borrowings from other languages. As in much of the rest of the world, English has had the greatest influence on Bulgarian over recent decades [Wiki06].

2.7 Conclusions

We will find the defined rules for Romanization and grapheme-to-phoneme mapping very useful, since one of the important characteristics of initialization of the recognizer is the use of multilingual phoneme set through the Spice Toolkit [Sch04] to generate initial codebooks and to write initial labels (see subsection 5.1.3 in chapter 5).

The fact that most letters in the Bulgarian alphabet stand for one specific sound and that sound only, makes our task much easier than if we had to deal with big differences in the pronunciation of the letters in different word contexts. There are still many words to be found, where a voiced consonant is spoken as voiceless in a word-end position or even somewhere in the middle of a word. We will not make use of this characteristic in the current version of the recognizer.

A challenge to deal with, are the many flexions of the words expressed by gender, number, gender specific suffixed definite articles and huge amount of verbal forms. This leads to a large number of different words in a given text and to many of them, which may have never been seen in other texts even when the text deals with the same topic.

3. Data Collection

To build a speech recognizer, requires a large amount of data to train the recognizer to understand continuous speech. Speakers read written sentences and both text and speech are recorded into the computer. In this section, you will find detailed description of the whole text and speech collection process, which is associated with the training of our speech recognizer. First, we are going to introduce what kind of text data was collected, preprocessed into suitable formats, and other relevant details about our recording process.

3.1 Text Data

Text data from national and international political and economic articles from Bulgarian newspapers was the primary source for this project. One of the reasons to choose this kind of articles is the present availability of text and speech data in the same domain in other languages, which are collected by students during the work at the Global Phone project [SW01], [Sch02]. Thus, the possibility of occurrences of analogical words in the other language was high and a it presented a good opportunity for comparison at different levels. Besides this, an important characteristic of this domain is that it guarantees the use of a large range of words and allows us to collect more text data because such sources are released every day.

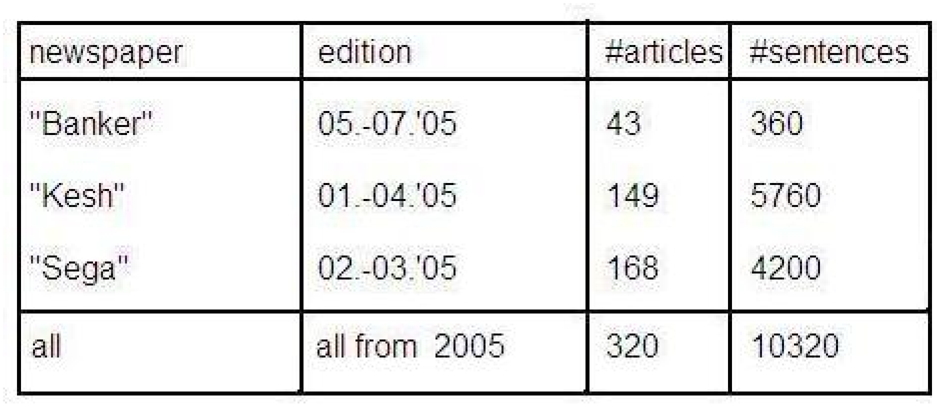
We collect speech data from 77 native speakers and each one of them read up to 100 - 120 sentences. The author downloaded and preprocessed over 10,000 sentences from 350 articles, which are chosen from the online editions of three national Bulgarian newspapers.

These are the newspapers:

- "Banker" (http://www.banker.bg/)
- "Kesh" (http://www.cash.bg)
- "Sega" (http://www.segabg.com/)

Each of the articles has been stored in a separate text file with a name containing the newspaper's name, its edition, and main part of the article's name. Table 1 illustrates basic information about the collected text data. No detailed information is given for the number of words and vocabulary of the collected text at that time, but will be provided later in this chapter.

Table 1 Basic information about the collected for the recordings text data



newspaper	edition	#articles	#sentences
"Banker"	05.-07.'05	43	360
"Kesh"	01.-04.'05	149	5760
"Sega"	02.-03.'05	168	4200
all	all from 2005	320	10320

3.2 Data Processing

The first processing of the text data aimed to guarantee a consistent reading of the sentences by all speakers. The domain of the articles causes many occurrences of abbreviations, acronyms, foreign words (written in Roman) and digits. In the unprocessed text data many of the abbreviations do not always end by a dot as expected, acronyms are used mostly for parties and institutions, and some of them are written in Roman. Sometimes the same acronym is written in two different ways. Most of the foreign words are names of institutions or specific day-to-day

English words. Digits describe not only an amount or a year, but also a specific amount (money, percentage, etc).

To make the reading simple for everyone, many of the abbreviations are decomposed into the origin words which they present, but due to their number, variability, and lack of time to prepare, many of them are written the way they should be spoken letter by letter (for example ABC -> ABeCe). It is important here that all words with the same meaning to be read in the same way. To guarantee the reading of foreign words (written in Roman) by all speakers consistently, they are substituted by an equivalent words written by Cyrillic letters. We do not want our text data to contain any digits so all the numbers in the data are substituted by their corresponding words. Thus, consistency in reading is assured by replacing words and representing the meaning of the digits. The use of digits in our database and dictionaries is excluded.

All the text data is collected into one text file and few scripts are written in Python and Tcl to make the transformation faster and more consistent. A large number of changes are made manually, because of the differences in text units, used only once or twice in the entire text data or text units which have to be mapped to the same words written in different ways.

In order to make the text data more suitable for reading by the speakers, many text files are created, each of them contains 120 sentences written one sentence per line. These files are loaded in the Data Collection Toolkit (DCT), designed by Tanja Schultz, and described in the subsection 3.3 of this chapter. Difficulty with the separation of the sentences on a single line was the occurrences of dots not only to remark the end of a sentence, but also to remark the end of an abbreviation in the text (not all the abbreviations).

Unfortunately, the author was not aware of all requirements for the further use of the text data in Janus [WA-W+94] at the time of data collection, and many of the changes ended up conflicting with these requirements. In the end, there was not enough time for more processing before the recording of speakers. A second processing of the data is performed after the recording of speakers in order to clean the text data and make it more appropriate and useful for building the Bulgarian speech recognizer with the Janus Speech Recognition Toolkit (JRTk) [FGH+97].

During the second processing, many changes are made to correct the lower and upper case writing of words with the goal that only the names in our text data should begin with an upper letter and some abbreviation should contain only upper letters too. Most of the difficulties here came because all of the words in the beginning of sentences are written with in upper case but not all of them are names. Some of the changes are also caused by previously made changes by the author; for example mapping of an abbreviation to words, or by the occurrences of titles written all in upper case letters. The process of making corrections in the case is expedited by creating a list of all words, which contain at least one upper case letter. All of the first words in the sentences are excluded and for the rest we assume that they are all names. We create another list, which contains only the first words of all sentences and if any of them do not occur in the first list, we rewrite it with lower case letters. After this, the remaining words with an upper case are corrected one by one and the most errors are removed.

At that time, all of the abbreviations, which are rewritten for better readability (for example ABC -> ABeCe) on one hand contain both upper and lower case, and on the other hand are not correctly written words - not even correctly written abbreviations. This is why many of them are rewritten into their origin, which requires special consideration when we later build a phoneme dictionary with them.

All of the punctuation is removed with exception of some of the "-" characters, which are part of a word. The corresponding difficulty here is the occurrence of "-", which denotes a break in the thought but not a part of a word. Difficulty by cleaning the text data is also the occurrence of some type-errors in the articles, which are probably made by the authors of the newspaper articles.

3.3 Recording Tools

The speakers are recorded directly into a laptop using the Data Collection Toolkit (DCT), created by Tanja Schultz. This toolkit loads text files, records sentence after sentence and allows different settings. The first three menus deal with the collecting of data and rest of the menus are optional. The Toolkit is made to prepare the preliminary files, required by the Janus Speech Recognition Toolkit [FGH+97].

The first menu - TEXT - simply loads text files. For each of the speakers multiple files can be loaded. For our goal, the sentences in the file should be written on a single line with one empty line between them. Thus, the Toolkit will load sentence after sentence in the order as provided by the texts.

The second menu - SPEAKER - enrolls new or load known speaker. For each new speaker a datasheet should be filled, which contains demographics of the speaker (name, age, gender, job, dialect and others) and of the recording environment. A Consent Form is included and the speakers have to agree with it in order to participate.

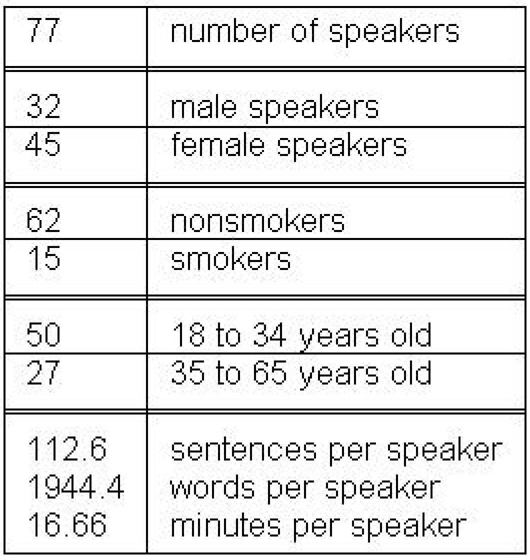
The third menu - RECORDING - performs the recording sentence by sentence. After recording of a sentence, it can be listened and cancelled if a mistake occurs or something happens. The sentence can be skipped too. In the moment of agreeing with a spoken sentences and displaying of the next one, files are being written into a chosen in the Toolkit SETTING menu directories. These directories contain a file for each speaker named by a number with the ending .dat and folders for speakers containing other folders for each of the spoken sentences. The last folder contains two files with the same name but with different extension: .adc for the recorded speech and .trl for the transcription of the spoken sentence. A file 'index' is also stored with the number of recordings per speaker.

The optional menus in SETTINGS allow us to choose the number of sentences per speaker (default is 200). In our case, this number is set to 120. Another setting enables us to switch on and off the checkADC ability (if ON - each of the sentences is checked for quality and there is an opportunity first to listen to the spoken sentence before agreeing with it and go to the next one). In our case, this option is switched to ON. Further, we can change the audio format - sampling rate, recording channels, encoding, and byte order. We are working with a sample rate of 16 kHZ, "mono" recording channel, "Lin16" encoding and little Endian byte order. In this menu, we can also choose the recording directory, which is set for us by default.

3.4 The Speakers

Seventy-seven speakers are recorded for twelve days in the cities Sofia and Pazardzhik. Almost all of them come from the west or the central part of Bulgaria. Table 2 shows statistic, containing the number of males, females, and speakers over and under 35 years, smokers, and nonsmokers). The table contains also the average number of spoken sentences, words, and minutes for every speaker.

Table 2 Speakers information



77	number of speakers
32	male speakers
45	female speakers
62	nonsmokers
15	smokers
50	18 to 34 years old
27	35 to 65 years old
112.6 1944.4 16.66	sentences per speaker words per speaker minutes per speaker

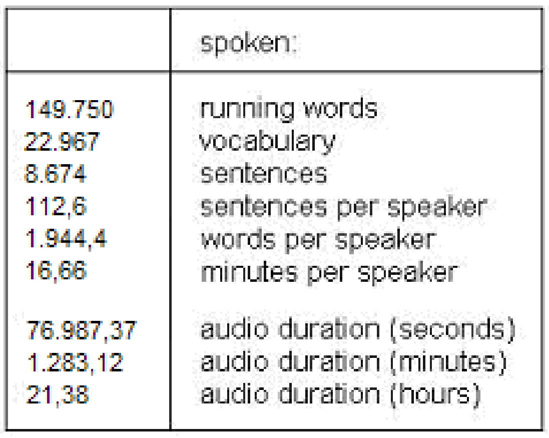
Almost all of them are speakers of true Bulgarian and do not have a dialect. The majority owns a degree and consists of well educated participants. The speakers are mainly graduated students, construction engineers, and teachers. During the recordings, information about each of them was collected and recorded into special “speaker” files. This kind of information is known that it might influence the speech. The speakers were allowed to remind anonymous if they want to and not to give their real names, but only few of them have chosen that option.

3.5 Recording Action

Almost 22 hours of speech data are collected in Bulgaria during the twelve days of recording action. Almost all of the recordings are performed in middle-sized rooms with a middle level of noise. Interesting point to note is that the speed and the loudness of reading vary widely for each of the speakers.

During the reading, most of the speakers get somehow nervous just because of the fact that they are recorded and thus they make many mistakes. Another reason for why they make mistakes is that many of the sentences happened to be long and with difficult vocabulary, which the people do not use every day in conversations. Further reason for the mistakes is the duration of recording. Even with breaks available, most of the speakers are not used to reading for such long durations. This mistakes are made mostly by adding or changing an entire word in the text, reading part of the word not correctly or just interrupting the reading because a feeling of making a mistake. The average duration of reading by one speaker was about 35-40 minutes. A best reading is performed by a lawyer, and by all of the teachers, but even they make mistakes. Table 3 shows the main information about the recordings.

Table 3 Information about recorded data



	spoken:
149.750	running words
22.967	vocabulary
8.674	sentences
112,6	sentences per speaker
1.944,4	words per speaker
16,66	minutes per speaker
76.987,37	audio duration (seconds)
1.283,12	audio duration (minutes)
21,38	audio duration (hours)

The above-described data is further divided into three groups. The proportion of the number of speakers in the different sets is approximately 8:1:1. The selection of the speakers for each of the groups also tries to guarantee the same proportion in the number of male and female speakers, and in the number of nonsmokers and smokers. Initial, proportion based on the age was desired, but unfortunately, it was not achieved because of wrong assumption for the age of some speakers. The biggest speaker set contains the data, which is to be used to build and train the speech recognizer, and includes 63 speakers. The second one contains the development data - data, which will be used to tune the parameters of the recognizer, after we already have one. The third group is used for the evaluation of the performance of the speech recognizer. Each of the smaller groups contains seven speakers. You can find detailed information about the speakers and the recordings in each of these groups in Table 4.

Table 4 Recordings

	number of speakers	male	female	age		smoker	non-smoker	number of sentences	running words	vocabulary	duration in min.	Speaker IDs
				<35	>34							
Training set	63	27	36	40	23	13	50	7.012	120.719	20.233	1.026,35	018,020,021,023,025, 026,027,032,035,039, 041,042,043,045,046, 047,048,049,050,052, 053,054,056,060,062, 064,065,066,067,069, 070,071,072,073,075, 077,078,079,080,082, 083,085,087,088,089, 091,092,093,094,096, 097,098,099,101,102, 103,104,105,107,111, 112,113,114
Development set	7	3	4	4	3	1	6	830	15.127	4.748	138,79	051,055,058,084,090, 100,106
Evaluation set	7	2	5	6	1	1	6	832	13.904	4.796	117,98	040,059,063,068,095, 109,110

3.6 Additional Data Collection

Later, additional text data is needed to build language models for the recognizer (see section 4.3). This additional data should not contain any of the text data that has been already collected. Very appropriate is the text data provided by the BulTreeBank project [SOK+02]. Thus, our additional text data is taken from the database of this project, after kind approval by Kiril Simov, a maintainer. The BulTreeBank project is based at the Linguistic Modeling Laboratory (LML), Institute for Parallel

Processing, Bulgarian Academy of Sciences and is funded by the Volkswagen Stiftung, Federal Republic of Germany under the program "Cooperation with Natural and Engineering Scientists in Central and Eastern Europe". The main objective of BulTreeBank project [SOK$^+$02] is to create a high quality set of syntactic structures of Bulgarian sentences within the framework of HPSG (Head-Driven Phrase Structure Grammar). Ideally, the tree bank should contain samples of all the syntactic structures of the language. These sentences should serve as templates for future corporate development, could become the basis for the development of a more comprehensive test suite for NLP applications, and can be used as a source for grammar extraction and for linguistic research.

What we use from the BulTreeBank database are e-newspaper articles in the same topic as the previously collected text data. We extract these articles from the following e-newspapers, provided by the BulTreeBank project:

- "Sega" (http://www.segabg.com/), editions: from January to December 2002.
- “Standart” (http://www.standartnews.com/), editions: from January to December 2002

The additional text data contains about 8.5 millions of running words and 248.993 vocabulary words. This data is not clean in the meaning of the previously described cleaning of text data by removing digits, symbols, punctuation, abbreviations and acronyms. In addition, the upper case words in the beginning of the sentences are rarely names. Some of the scripts used to clean the previous data are applied to the additional text data too. Due to the big amount and variability of the additional data, there are more changes in the text to be made in order to have fully clean text data. Unfortunately, the time-costly nature of cleaning the text does not allow us to make these changes. In addition, only vocabulary is extracted from xml dictionary files in the BulTreeBank database. The so extracted vocabulary contains 13.924 words.

The additional text data is used not only to create language models, but also to extend the training dictionary built on the previously collected training data. Thus, new and much bigger training dictionaries are created, using both training and additional text data and the BulTreeBank dictionary files. These extended dictionaries contain the pronunciations of the words and are used in the decoding process (see section 4.4). The use of extended dictionaries simply means that when looking for a word in the dictionary, the probability to find it will be greater. Since many words from both development and evaluation sets are not seen in the training set, a search in the training dictionary built only on this training set will perform worse than a search in an extended dictionary.

The use of extended dictionaries for decoding with Janus[WA-W$^+$94] requires their vocabularies not to contain the specific for Janus symbols. Section 5.1 gives more details on the extended vocabularies and dictionaries built using the additional data.

4. Basics of Speech Recognition

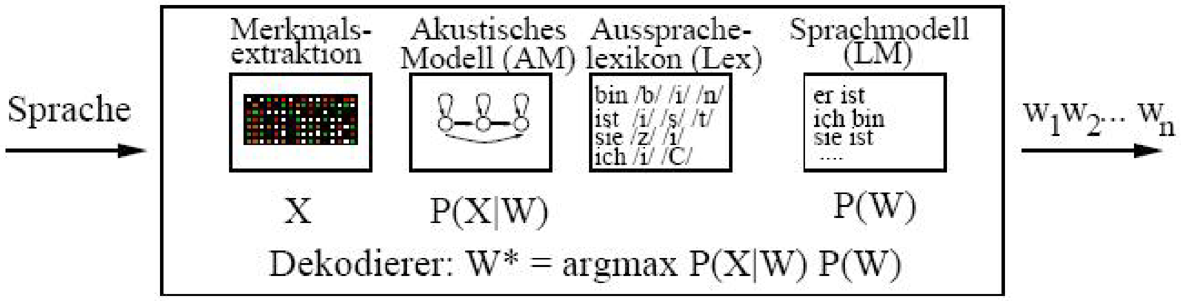
Speech recognition has the aim to produce an accurate written output from a human speech input. To find the best word sequence for a given utterance, a set of the n most likely written word sequences is computed, and one with the highest probability is chosen to be a representation of what was spoken. The probabilistic theory has been found to be most suitable to give a definition of this problem in the practice. The fundamental equation of speech recognition is:

$$\begin{aligned} W^* &= \underset{W}{\operatorname{argmax}}\ P(W|\mathbf{X}) \\ &= \underset{W}{\operatorname{argmax}}\ \frac{P(\mathbf{X}|W)\cdot P(W)}{P(\mathbf{X})} \\ &= \underset{W}{\operatorname{argmax}}\ P(\mathbf{X}|W)\cdot P(W) \end{aligned} \qquad (3.1)$$

What this equation says, is that to find the most likely word sequence W* for an observed acoustic event (given utterance X has been spoken), we need to maximize the product of P(X|W) and P(W). P(X|W) is the probability to observe the signal X if it's known that the written word sequence W was spoken and is called acoustic model. P(W) is the a-priory probability for the written word sequence W to be spoken and is called “language model”. To maximize this product we need an efficient search for a word sequence W* and this is an assignment for a so-called decoder. Solving of these three problems is the main challenge of automatic speech recognition. Their basics are explained in this chapter [Sch00].

Figure 8 shows a simple structure of the components of a speech recognizer. Besides an acoustic and a language model, we need a component for extraction of the features from the speech signal. The basics of this process, called signal preprocessing, are explained in first section of this chapter. A pronunciation dictionary contains all the words that can be recognized, and is the word source for producing a hypothesis of what has been said. Once all these components are available, the decoder finds the best hypothesis, containing a word sequence $w_1w_2...w_n$.

Figure 8 Simple speech recognizer [Sch00]



4.1 Signal Preprocessing and Feature Extraction

Some preprocessing is required before we use the speech signal for recognizing. Its goal is to extract specific information from the acoustic signal, which is crucial for speech recognition. We are looking for a compact but also complete description of the signal, which will allow us to make a proper classification of the speech data for further analysis.

First, discretization of the signal is performed. There are several benefits of working with a set of time discrete classification vectors. This allows us to represent the signal in a digital way, to work with less data and with less space capacity. When we have a discrete signal, we easily can transform it or extract the relevant information that characterizes the speaker, the utterances, and the environment effects.

We achieve an accurate representation of the speech sounds used for speech recognition through a 16-bit A/D conversion with a sample rate of 16 kHz. This sample rate is chosen considering the Shannon theorem [Sha49], which says that the sampling frequency must be greater than at least twice the input signal bandwidth in order to be able to reconstruct the original perfectly from the sampled version. Any human sound can be reconstructed with a sampling rate of 20 kHz, but to represent the human speech we do not need a sampling rate greater than 16 kHz.

Next, we do a spectral analysis of the quantized signal, which has the main goal to reduce the data used. After we assume that for minor interval (5ms to 30 ms), the signal is stationary, at every 10ms starting from the beginning of the speech signal we take out 16ms of it. To take an interval out we multiply the whole signal with a special function with a finite bandwidth. We will use the Hamming function but there are also other functions that can be used such as the Hanning or the Gaussian function. For each of the extracted intervals we use the Fast Fourier Transformation (FFT) approach to compute its spectrum. Computing the FFT for an interval of the speech longer than the used offset assures a better use of each of our intervals' borders.

After we use the FFT onto each of the 256 sample coefficients, we end up with 129 spectral coefficients. In our computation, we do not use the phase spectrum, because it has turned out not to be important for the recognition since the information it gives us is not crucial for the understanding of the speech in real situations. Next step toward the reduction of the data is done based on applying Mel-Scaling which is filtering that imitate the different sensibility of the ear to high and low frequencies. This filtering reduces the number of coefficients in each of the vectors from 129 to 30 Mel-Scale coefficients. We transform these coefficients again with a Fourier transformation into 30 Cepstral coefficients. In speech recognition, we are usually interested in the first 13 cepstral coefficients with the first often being omitted in favor or a more robust energy measure. Mel Frequency Cepstral Coefficient (MFCC) features have more or less been adopted by the speech processing society as standard. MFCCs models the basilar membrane by a mel scaled frequency axis, and turns the convolution with the vocal tract into a sum by using the cepstrum instead of the spectrum. The first and second temporal derivatives of the cepstrum, called the delta and delta-delta coefficients respectively help give an estimate of the temporal variations in the signal. To improve the speech recognition rate vectors representing the delta and delta-delta normally augment the feature vectors [Fur86].

Once we extract the named features in the preprocessing phase, we can use methods for reducing the feature dimension even more. One of these methods is called Linear Discriminate Analysis (LDA) and it transforms the extracted feature vectors by the multiplication with an estimated matrix called LDA-matrix. This method reduces the dimensionality and at the same time keeps crucial features for maximally discriminating between the classes, built by similar features. The variability of the vectors as well as their class affiliation is considered, the average variance inside the classes is minimized, and the variance between the classes is maximized [Sch00].

4.2 Acoustic-phonetic modeling

Subject of the acoustic modeling is to find the probability to observe the signal X if it is known that the word sequence W was spoken.

Different units of speech can be used to model it [Sch00]. When it comes to recognizing continuous speech, usually phonemes and subphonemes are used. The phonemes are very flexible units of speech. In most of the languages, only 30 to 50 phonemes are enough to represent the words in this language. Sounds as entities in a linguistic system (as the smallest units that distinguish a minimal word pair, e.g. *pin* vs. *bin*) are termed phonemes [Sch05]. It is a small enough unit, so it is supposed to occur in a lot of the speech data used to build the recognizer. Usually three subphonemes are used to represent the dynamic in a phoneme. In regard of the co-articulation, polyphone can be used instead of phoneme, which is the phoneme in a defined context of other phonemes next to it. In the same way, subpolyphones can be built from subphonemes. To overcome the huge number of the resulting subpolyphones and thus to use more speech data to train the acoustic models, generalized subpolyphones are built by grouping of context dependant models. The process is called agglomeration and there are different algorithms to perform it [YW94]. In this work, top down clustering with Entropy – criteria is used.

The Hidden Markov Model (HMM) [Rab89] has been found to be very suitable to model the dynamic and variability of the speech expressed by its acoustic units. A HMM is composed by a set of states, called HMM-states, a likelihood distribution, which indicates the probability for an HMM state to be the beginning state of a state sequence. HMM also contains a matrix with probabilities for the transition of each of the states into another, and of a matrix with emission probabilities, which is the probability for an output to be observed when a state is entered. It also contains a set of symbols, which can be emitted [Sch00]. A HMM is called discrete if the matrix with emission probabilities consists of probability tables, containing discrete values. A HMM is continuous if instead of probability tables there are probability densities. Usually an emission probability is modeled as a Gaussian mixture distribution [Sch00]:

$$b_j(\boldsymbol{x}) = \sum_{l=1}^{L_j} c_{jl} \cdot Gau\beta(\boldsymbol{x}|\mu_{jl}, \Sigma_{jl}) \quad , \quad \sum_{l=1}^{L_j} c_{jl} = 1 \tag{3.2}$$

,where L_j is the number of defined distributions in the state S_j and the Gaussian distribution *Gauß(x |μ, ∑)* with mean vector *μ* and a covariance matrix ∑ is defined as:

$$Gau\beta(\boldsymbol{x}|\mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^d |\Sigma|}} \cdot e^{\frac{1}{2}(\boldsymbol{x}-\mu)^T \sum_i^{-1} (\boldsymbol{x}-\mu)} \tag{3.3}$$

The mean vectors μ_{jl} and the variances $\sum_{jl}$ are called a codebook of a model and the weights coefficients are called mixture weights. If each of the states of the HMM has its own probability density with own mean vectors, variances and weights, then it is a fully continuous HMM. If a couple of states shares the same Gaussian distribution and still each of them has its own mixture weights, then the HMM is semi continuous.

There are three main problems to be solved in the acoustic modeling [Sch00]. The first of them is the evaluation problem. The answer it gives is what is the probability for a given HMM to produce a certain observation sequence and is the solution is achieved by a so called Forward algorithm. The second problem - the decoding problem, is to find the path of states, which is most likely to have produced the certain observation sequence. The Viterbi algorithm provides its solution. The last one is the optimization problem or how to find a parameter for the model of the word w, such that the probability of producing the certain sequence to be maximized. Here the Forward-Backward algorithm is applied.

4.3 Language Modeling

There are two main methods for estimating a language model - linguistically, by imitating the syntactic structure of a language, and statistically, by using big text corpora to evaluate a-priori probabilities for the occurring words sequences in this text. Mostly we refer to a statistic language modeling, which aims to estimate the probability P(W) for a word sequence, without to consider any acoustic information about the words [Sch00]. Sometimes combination between it and the lingual method are made, in order to overcome some limitations of the coded text corpora.

An approach called n-gram is widely used to model language. The a-priory probability is given by [Sch00]:

$$P(W) = \prod_{i=1}^{n} P(w_i|w_1 w_2 \ldots w_{i-1}) \tag{3.4}$$

Time and space, necessary to compute these probabilities, are reduced by different methods. In general, a function on w_1 to w_{i-1} has to be estimated, which tries to combine somehow different sequences into classes. Usually the length of the considered word sequences is limited to 3 words. This technique has been proven to provide as good performance as if there were no limitation. Thus, there are bigrams and trigrams (sequences of 2 or 3 words). This is how a trigram probability is estimated [Sch00]:

$$P(w_i|w_{i-1}, w_{i-2}) = \frac{\mathrm{Count}_{Trigramm}(w_{i-2}, w_{i-1}, w_i)}{\mathrm{Count}_{Bigramm}(w_{i-2}, w_{i-1})} \tag{3.5}$$

Even considering this limitation in the number of words in a sequence, there is usually not enough text data to compute stable and accurate probabilities for many of the occurring trigrams. There are two main techniques to overcome this problem - discounting and backing-off. The discounting approach decreases the probability of the most likely trigrams and distributes the discounted probability over the most unlikely trigrams. The backing-off method takes rather the probability of the corresponding bigram and multiplies it with a factor estimated on the excluded and on the middle word to guarantee the sum of the trigram probabilities is 1.

There is a measurement for how good a language model is, called perplexity. Perplexity measures how many different words are likely to come after a certain word according to the language model. Perplexity of the language model is the average perplexity, computed on all of the words in the text data. Usually the language model with the lower perplexity is the better one, when different language models for a given text are compared.

4.4 Decoding and Performance Measurement

Once both acoustic P(X|W) and language model P(W) are estimated, the product of them has to be maximized in a process called the search, or decoding (see next section). The probabilities of both models are combined with the aim to produce the most likely word sequence, which represents what has been said. The combining usually has to be parameterized to achieve a better-weighted end model. A parameter *z* is used to give a weight to the language model and a parameter *q*, called word penalty, normalizes the different lengths of the observed word sequences [Sch00]:

$$P(W|\mathbf{X}) = \frac{P(\mathbf{X}|W)\cdot P(W)}{P(\mathbf{X})} \xrightarrow{z,q} \frac{P(\mathbf{X}|W)\cdot P(W)^{z}\cdot q^{|W|}}{P(\mathbf{X})} \tag{3.6}$$

Both parameters *z* and *q* are adjusted manually.

An output of the decoding is an end hypothesis containing a word sequence and made by the speech recognizer in order to represent the speech with written words. There is an approach, called One-Stage-Dynamic-Time-Warping, which combines the Viterbi algorithm for decoding of a single word with an algorithm, which finds the best segmentation for single words in a sequence. This segmentation is performed by building a sequence on an optional coupled word model instead of a sentence model. A sequence path is built considering the acoustic likelihood of a word to descend into another word, and the text information about the context, given by the language model. To keep the complexity of the search for best sequence paths in normal time and space dimensions, there is a technique to examine only the most likely word transitions, called Pruning [Sch00]. Not only of the best path, but also of the best N paths can be estimated, which allows further processing to obtain even more accurate word sequence path.

Once a hypothesis, containing a word sequence is computed, there is need to estimate how good the recognition is. There is a measurement called Word Error Rate (WER), which considers how many words has been substituted (N_{sub}), inserted (N_{ins}), or deleted (N_{del}) in the process of matching of the hypothesis to the reference sequence (what has really been said).

$$WE = 100 \cdot \frac{N_{sub} + N_{ins} + N_{del}}{N} \tag{3.7}$$

N is the number of all words in the reference sequence, which is usually a sentence. This measure considers the whole words, but there are also other measurement methods, which estimate the performance of the recognition based for example on phonemes. We used WER for our experiments.

5. Experiments and Results

In the previous chapters, the Bulgarian language and the basics of speech recognition have been introduced. A set of 45 phonemes is chosen to represent the Bulgarian sounds in our recognizer (see chapter 2). Text and speech data is collected and preprocessed in order to meet specific requirements (see chapter 3). The introduced basics of speech recognition (see chapter 4) will help better understand how the Bulgarian speech recognizer is now trained, developed and evaluated. This chapter gives detailed description of different experiments regarding not only Bulgarian data, but also Russian data.

The first section describes the steps of creating and training the Bulgarian acoustic model. This includes building a pronunciation dictionary and database in Janus, initializing and training the acoustic model. The second section gives detailed information about creating language models using both Bulgarian and Russian data. Russian data is added to determine how the text data from a similar language enhances the Bulgarian language models, and thus, the performance of the Bulgarian speech recognizer. To understand the influence of Russian, the overlap of vocabularies and coverage of texts between Russian and Bulgarian are estimated. The third section presents decoding and performance evaluation of the recognizer using different language models. Both context independent and context dependent systems are built. Recognition experiments are made later using only the context dependent system.

5.1 The Acoustic Model

The basics of acoustic modeling are briefly introduced in section 4.2. The created phoneme set (see Figure 5) contains all Bulgarian acoustic sounds, which are now trained using the collected training speech data and its corresponding training transcriptions. In order to do this, a training pronunciation dictionary based on the training vocabulary and a Janus database are created. The initialization and optimization process of acoustic codebooks is performed via a tool called SPICE [Sch04] and via the Janus Speech Recognition Toolkit [FGH$^+$97].

5.1.1 Pronunciation Dictionary

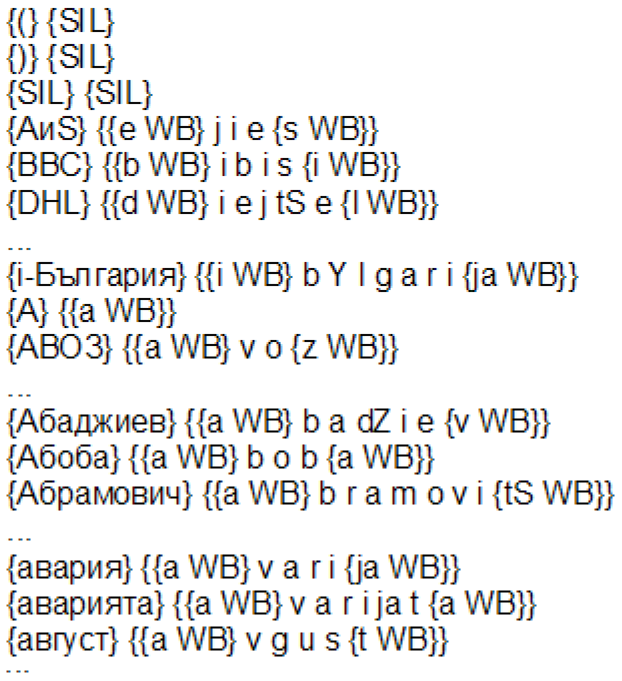
Once ADC and transcription files of the spoken utterances are collected via the Data Collection Toolkit (DCT), a training vocabulary and a training pronunciation dictionary are built. First, a Tcl script is created to extract a vocabulary from the training text set. Before extracting this training vocabulary, it might be very helpful to spend some time cleaning the training text again. Since the shell of Janus [WA-W$^+$94] uses Tcl commands, an appropriate approach is to assure that the vocabulary used to create a pronunciation dictionary does not contain any symbols (quotes, braces ...), which are specific to Tcl. Their absence helps avoiding problems with Janus. The only symbol left in the training vocabulary is the dash symbol. It might be part of words in some cases, namely when two words are combined into one. Another issue of cleaning the vocabulary is to avoid the occurrence of the same word written in different ways. Exceptions are those words, which can be correctly written in both upper and lower case notation.

A Perl script is adapted (see annex A) and used to build a training pronunciation dictionary (also called a “phoneme dictionary”). The script is executed on the previously extracted training vocabulary. The phoneme sequence is added to each of the words in the vocabulary. This dictionary is a user readable file with a specific structure (see Figure 9). The sequences contain only phonemes defined in the phoneme set for Bulgarian (see section 2.4). In summary, the phoneme set contains 45 phonemes – 6 vowels, 1 semivowel, 19 hard consonants, 15 palatalized consonants, 2 diphthongs and 2 additional consonants.

The phoneme sequences are created according to a set of mapping rules, which are defined in the Perl script. The number of the mapping rules is 190. These rules represent the grapheme-to-phoneme relationship model (see Figure 5 in subsection 2.4.4) and all its variations caused by the occurrences of case sensitive words. Since there are acronyms in the training text data, more rules are applied to denote their correct pronunciation in the training pronunciation dictionary. Thus, 16 more rules for 16 acronyms are defined. In the end, 206 rules (190 grapheme-to-phoneme mapping rules + 16 rules for acronyms) are used to create the training pronunciation dictionary.

As already mentioned, extended dictionaries are built later using the additional data. The same Perl script used to build the training dictionary is later used to build these extended dictionaries. To provide correct pronunciations for 52 more acronyms, 52 new acronym rules are added to the Perl script. Thus, the script that is used to create extended dictionaries using not only the training data but also the additional data, consists of 258 mapping rules (190 grapheme-to-phoneme mapping rules + 68 rules for acronyms).

Figure 9 The Janus-Dictionary



```
{(} {SIL}
{)} {SIL}
{SIL} {SIL}
{АиS} {{e WB} j i e {s WB}}
{BBC} {{b WB} i b i s {i WB}}
{DHL} {{d WB} i e j tS e {l WB}}
...
{i-България} {{i WB} b Y l g a r i {ja WB}}
{A} {{a WB}}
{АВОЗ} {{a WB} v o {z WB}}
...
{Абаджиев} {{a WB} b a dZ i e {v WB}}
{Абоба} {{a WB} b o b {a WB}}
{Абрамович} {{a WB} b r a m o v i {tS WB}}
...
{авария} {{a WB} v a r i {ja WB}}
{аварията} {{a WB} v a r i ja t {a WB}}
{август} {{a WB} v g u s {t WB}}
...
```

The string “WB” (word boundary) is called a tag and is used to indicate the first and the last phoneme in a word. There are also other tags, used to identify syllable boundaries, or stress for example, but they are not used in this dictionary. A tagged phoneme can contain not only the name of the phoneme but also the phoneme itself, followed by one or more tag names. Three more entries are added to the dictionary: a filler word SIL, which indicates silence and two specific symbols for Janus, “(” and “)”, which indicate the begin and the end of the utterance. In Janus [WA-W$^+$94] the whole structure is saved as a so-called Dictionary object.

The Perl script used to create the pronunciation dictionary can be easily adjusted to first “romanize" the Cyrillic words and then map them to phoneme sequences. Janus [WA-W$^+$94] now supports the UTF-8 format, used to represent Cyrillic. This is why Cyrillic letters can be used in the pronunciation dictionary without first converting them to Roman letters. Instead of “romanizing" the letters, the Perl script attaches “CZ_” to the phoneme, which has been created via the mapping rules. This is exactly where “romanizing” can be performed. The next step simply adds the phoneme sequences to their corresponding words in the dictionary.

The training pronunciation dictionary used to train the acoustic model is based on the training vocabulary, which contains 20,233 words. Thus, the training dictionary contains 20,236 (20,233 + 3) entries. The extended vocabularies and dictionaries used for language modeling and for decoding are presented later in this chapter (subsection 5.2.1).

5.1.2 Janus Database

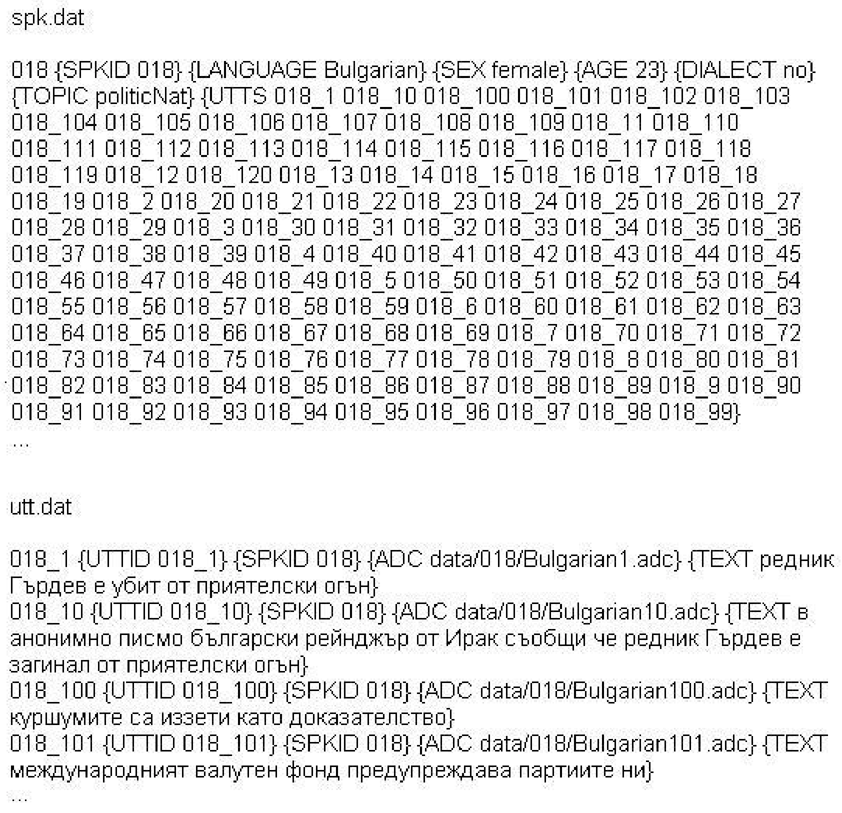
To simplify the use of the collected speech, text and speaker data, a Janus database is created. Its goal is to provide an easy way of accessing all the collected information. The database is created in Janus [WA-W$^+$94] by a Janus-script, which simply extracts and structures the information already provided by the Data Collection Toolkit (DCT). Thus, instead of dealing with the large amount of files created by the Data Collection Toolkit, only four files that represent the Janus database are used. Two of them - utt.idx and utt.dat – contain the information about the utterances (transcriptions, utterance ID, etc.) and the other two - spk.idx and spk.dat - describe the speakers. The .idx files store indices that increase the performance of database queries and the .dat files contain the database itself. In Janus [WA-W$^+$94], these files are represented by a so-called Database object. Figure 10 shows the content of the two .dat files.

5.1.3 Initialization of the Recognizer

To initialize a context independent Bulgarian recognizer, a bootstrapping technique is applied. The idea behind this is to use an already existing acoustic model and map it properly to the acoustic parts of speech chosen to represent the Bulgarian language. Since this is the first Bulgarian recognizer to be built at the ISL, initial weights (codebooks, distributions) from another language are needed. During the GlobalPhone project, a multilingual recognizer has been created for multiple languages. Its acoustic model contains phonemes that can be mapped to the

Bulgarian phonemes. Thus, each of the Bulgarian phonemes is manually mapped to a phoneme used in the multilingual recognizer. The initially created acoustic model is based on the HMM approach and provides weights for the set of 45 Bulgarian phonemes. In this model, each of the phoneme is divided into 3 subphonemes (begin, middle and end of the phoneme is modeled as a separated part of speech). A silence phoneme is also included in the phoneme set and is modeled as a middle subphoneme. The final set contains 136 subphonemes (135 subphonemes and the silent subphoneme).

Figure 10 Example from the database



```
spk.dat

018 {SPKID 018} {LANGUAGE Bulgarian} {SEX female} {AGE 23} {DIALECT no}
{TOPIC politicNat} {UTTS 018_1 018_10 018_100 018_101 018_102 018_103
018_104 018_105 018_106 018_107 018_108 018_109 018_11 018_110
018_111 018_112 018_113 018_114 018_115 018_116 018_117 018_118
018_119 018_12 018_120 018_13 018_14 018_15 018_16 018_17 018_18
018_19 018_2 018_20 018_21 018_22 018_23 018_24 018_25 018_26 018_27
018_28 018_29 018_3 018_30 018_31 018_32 018_33 018_34 018_35 018_36
018_37 018_38 018_39 018_4 018_40 018_41 018_42 018_43 018_44 018_45
018_46 018_47 018_48 018_49 018_5 018_50 018_51 018_52 018_53 018_54
018_55 018_56 018_57 018_58 018_59 018_6 018_60 018_61 018_62 018_63
018_64 018_65 018_66 018_67 018_68 018_69 018_7 018_70 018_71 018_72
018_73 018_74 018_75 018_76 018_77 018_78 018_79 018_8 018_80 018_81
018_82 018_83 018_84 018_85 018_86 018_87 018_88 018_89 018_9 018_90
018_91 018_92 018_93 018_94 018_95 018_96 018_97 018_98 018_99}
...

utt.dat

018_1 {UTTID 018_1} {SPKID 018} {ADC data/018/Bulgarian1.adc} {TEXT редник
Гърдев е убит от приятелски огън}
018_10 {UTTID 018_10} {SPKID 018} {ADC data/018/Bulgarian10.adc} {TEXT в
анонимно писмо български рейнджър от Ирак съобщи че редник Гърдев е
загинал от приятелски огън}
018_100 {UTTID 018_100} {SPKID 018} {ADC data/018/Bulgarian100.adc} {TEXT
куршумите са иззети като доказателство}
018_101 {UTTID 018_101} {SPKID 018} {ADC data/018/Bulgarian101.adc} {TEXT
международният валутен фонд предупреждава партиите ни}
...
```

A new powerful toolkit called SPICE (Speech Processing - Interactive Creation and Evaluation) [Sch04] introduces an easy technique to bootstrap a system for both experienced and non-experienced users. A user-friendly graphical interface contains a set of IPA tables that contain all the phonemes used to create the multilingual recognizer [SW01], [Sch02]. The toolkit provides an easy way to listen to each of the phonemes in order to find the most appropriate if several choices are available. The desired phonemes are then simply selected from the tables and given labels, which will be used as phoneme names in the system. If there are phonemes not included in the SPICE toolkit, they can be manually added without much effort.

The mapping of Bulgarian to multilingual phonemes is very good. The reason is that some phonemes used to build and train the multilingual recognizer come from languages acoustically close to Bulgarian, like Croatian, Russian, and Polish. The SPICE toolkit [Sch04] provides the initial weights for the Bulgarian speech recognizer. The 128 Gaussian mixture distributions, a 3-state HMM structure, and a fully continuous architecture characterize the initial Bulgarian system. A set of description files for the system is created as part of the initialization. One of these files contains a list of the 136 subphonemes in the phoneme set (PhoneSet), another one contains a detailed list of the 136 codebooks (codebookSet). The 136 mixture-weight distributions are also listed in file (distribSet). A simple context-querying decision tree is stored in file named distribTree. It contains questions about only the central phone, and one leaf node for each of the 136 distributions (distribTree).

Another issue when initializing the recognizer is the assignment of labels. Labels are mappings from frames to subphonemes and they show the recognizer what has been said at what time. Instead of using randomly defined labels, the initial labels for Bulgarian are computed based on the existing multilingual recognizer through a Viterbi algorithm [Sch00]. Thus, the initialization of the Bulgarian speech recognizer is finalized. The next step is training the recognizer. It is performed in several iterations, where computing new labels means optimization of the mapping of frames to subphonemes. The new labels are then used in the training iterations, which are known to improve the acoustic model of the speech recognizer.

5.1.4 Training a Context Independent System

The initialized Bulgarian speech recognizer now undergoes a training iteration. Multiple Janus-scripts are edited and executed in order to train the context independent system. The paths to all of the important files in the system are written into the so-called description file (**desc.tcl**). The desc.tcl file is used not only during the training of the recognizer, but also during the decoding. It is very useful, because it allows the initialization of all relevant files at once. Depending on the training step, the paths of the actual codebooks, distributions and labels are edited.

Using the **labels.tcl** script, labels of the speech are written into files. Since the given transcriptions of the utterances are used, the Viterbi algorithm does not perform real recognition.

The **mean.tcl** script initializes mean vectors, which will help to cluster the feature vectors into classes (see the samples' explanation and the k-means algorithms). The already described Linear Discriminative Analysis is applied (see section 4.1). Every feature vector is multiplied with the LDA matrix and is transformed into a 43-dimensional vector. The LDA matrix is dependent on the acoustic model and has to be recomputed if the model changes. The computation of the new feature vectors is done via the script **lda.tcl**.

Once the labels are available and dimension reduction via LDA is performed, a subphoneme is assigned to every frame of the training data. Extracting samples is a technique where for every frame of the training data, each of the feature vectors is assigned to exactly one subphoneme. The stored vectors will be further used by the k-means algorithm to initialize Gaussian mixture distributions, or the so-called codebooks. The number of collected sample vectors is previously defined and is stored in a separate file for each of the subphonemes. The extraction of the sample vectors is done via the script **samples.tcl**.

For each one of the subphonemes the initial 128 distributions are computed from the sample vectors via the k-means algorithms, executed in the script **kmeans.tcl**. Clustering into 128 classes is performed, whereas every sample vector is assigned to a class, new mean vectors are computed for each class, and sample vectors are assigned again. In the end, a codebook for every class is created, based on its mean vector and its covariance matrix. The system is context independent and the number of codebooks (136) is equal to the number of modeled subphonemes. Each phoneme is presented by three subphonemes, which indicate begin (-b), middle (-m) and end (-e) of the phoneme. The SIL-phoneme is the only exception, because it is represented only by one subphoneme (SIL-m).

The next step is performed by the execution of the script **train.tcl**. The codebooks and distributions are optimized with the Viterbi algorithm or with a kind of Forward-Backward algorithm. This step is performed 6 times one after another and every time new parameters replace the old ones. After the codebooks are optimized, new labels are written. Then, the same sequence of scripts is performed again (means.tcl, lda.tcl, samples.tcl, kmeans.tcl and train.tcl).

5.1.5 Train a Context Dependent System

The trained context independent acoustic model only takes into account the phonemes (or subphonemes), but not their context (their neighbor-phonemes). In practice, the same phoneme is pronounced differently in a different context. Fortunately, the big amount of collected Bulgarian speech assures obtaining a better and more precise acoustic model when considering the context of phonemes. Considering the whole word as context of a single phoneme is too costly. An acoustic model that takes into account all possible different phonemes as context of a given phoneme needs huge amount of data and time resources. There are restrictions to be made in order to gain optimal performance of the training steps for the given amount of Bulgarian training data. This is why a “bright” and a “weight” of the context is defined, and different contexts are grouped into classes.

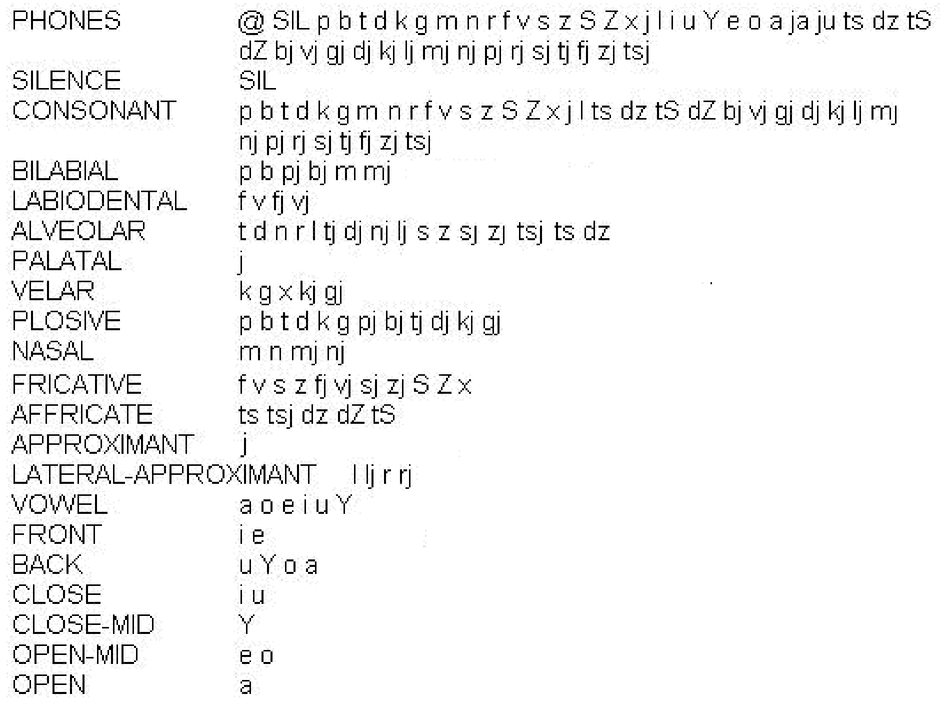
This grouping into classes is called clustering and is performed by answering questions about the phoneme and its context (respectively for a subphoneme). The system of questions can be seen in Figure 11. The questions are taken from an already existing system for another language. Further improvement of the acoustic model can be achieved by adding questions that reveal the specific characteristics of Bulgarian phonemes, for example, questions about palatalized consonants.

A phoneme, which is modeled depending on its context, is called a polyphone. In the Bulgarian recognizer, the maximum context bright is defined to be two, which means that the considered polyphone is called pentaphone. In Janus notation [WA-W$^+$94] it looks like this:

{A B C D E} -2 +2

This sequence describes the context of the considered phoneme C (called central phoneme), namely its left context containing the phonemes A and B, and its right context containing the phonemes D and E. In Janus [WA-W$^+$94] there is a restriction about the context. Any context of a phoneme can contain only one phoneme of the next word.

Figure 11 Questions about the phonemes



PHONES	@ SIL p b t d k g m n r f v s z S Z x j l i u Y e o a ja ju ts dz tS dZ bj vj gj dj kj lj mj nj pj rj sj tj fj zj tsj
SILENCE	SIL
CONSONANT	p b t d k g m n r f v s z S Z x j l ts dz tS dZ bj vj gj dj kj lj mj nj pj rj sj tj fj zj tsj
BILABIAL	p b pj bj m mj
LABIODENTAL	f v fj vj
ALVEOLAR	t d n r l tj dj nj lj s z sj zj tsj ts dz
PALATAL	j
VELAR	k g x kj gj
PLOSIVE	p b t d k g pj bj tj dj kj gj
NASAL	m n mj nj
FRICATIVE	f v s z fj vj sj zj S Z x
AFFRICATE	ts tsj dz dZ tS
APPROXIMANT	j
LATERAL-APPROXIMANT	l lj r rj
VOWEL	a o e i u Y
FRONT	i e
BACK	u Y o a
CLOSE	i u
CLOSE-MID	Y
OPEN-MID	e o
OPEN	a

The polyphones are collected into a tree structure. The polyphone tree grows with processing the whole training data, while Janus [WA-W+94] extracts polyphones from it. First, only questions about the central phoneme of the polyphones are answered and the number of leaves of the tree structure is equal to the number of phonemes. To each leaf, a basket that contains all polyphones that have the corresponding phoneme (the one stored in that leaf) as a central phoneme is attached. After building this tree, previously defined questions about different contexts of the phonemes are answered and the polyphones in the baskets attached to the leaves are further assigned – the tree grows. The splitting ends when there are no more good enough splits for our needs (in our system a maximum split count of 1000 was defined for the polyphones). The same is done respectively when dealing with subphonemes.

The script **Ptree.tcl** is executed in Janus [WA-W+94] to collect and count the subpolyphones. The script **train.tcl** is used to train their models and **cluster.tcl** builds the cluster tree structure. **Split.tcl** performs a divisive agglomeration to split the models into a set of classes. For the Bulgarian recognizer 2000 models are used. The training sequence of scripts is executed again (means.tcl, lda.tcl, samples.tcl, kmeans.tcl and train.tcl) but this time for the 2000 models of the new context dependent system. This last step is performed twice. The first time, the number of

Gaussian distributions is left unchanged and one 128-codebooks-final-system is created. For the second system, the number of Gaussian distributions is decreased to 64 and thus, the model complexity is reduced too, which decreases resource consumption.

5.2 The Language Model

In section 4.3, two different approaches in language modeling are introduced. To build language models for the Bulgarian speech recognizer, a statistical approach is used. Different n-gram language models are built based on both Bulgarian and Russian text data using different vocabularies. First, Bulgarian language models are created with the aim to evaluate the performance of the recognizer using only Bulgarian. Next, Russian data is added to build new mixed language models and the change of performance is studied. This section is divided into three subsections. The first one gives detailed information about the texts and vocabularies used to create Bulgarian language models. It also presents the values of important characteristics (OOV, Perplexity) of the Bulgarian language models, which are used to compare the different language models. The second subsection introduces the Russian text data and the estimated overlap and coverage between the Russian text and the Bulgarian text. This subsection also presents the created mixed language models (Bulgarian and Russian) and compares them. The last subsection describes the experiments performed using the acoustic model and different language models and the achieved improvements in the performance of the Bulgarian speech recognizer.

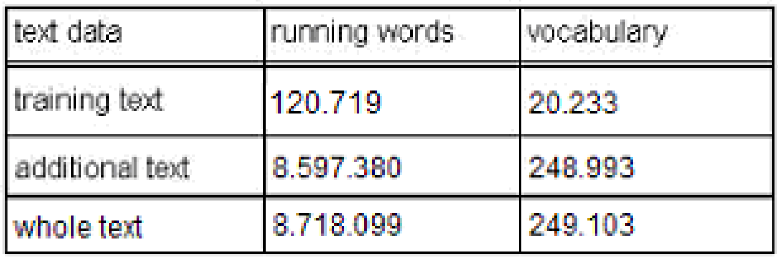
5.2.1 Bulgarian Language Models

The first half of the experiments to be performed requires the use of language models, which have been created based only on Bulgarian data. In this work, different n-gram language models differ from each other only in text and vocabulary used. These two factors affect the quality of the language model as well as the performance of the Bulgarian recognizer when using this language model. To help understand of how the language models are affected by the change of text and vocabulary, their detailed description is provided. In addition, unique notation for different texts and vocabularies is provided to simplify their use in this work.

5.2.1.1 Texts and vocabularies

There are two Bulgarian texts that are considered in language modeling. The first one is the text used to train the acoustic model. In this work, it is called the training text. The second is the text collected from the BulTreeBank project, called the additional text. Each of the Bulgarian language models is based on one of these texts or is created as interpolation of such language models. Table 5 displays both training and additional text with information about the contained running words and the size of the directly extracted vocabulary. In the training text, a word occurs about 6 times in average, and in the additional text, this number is 34.5. Of course, different words in the texts occur much more often than other. These are mostly prepositions and connectives. In order to estimate coverage between the Bulgarian and the Russian texts, both training and additional texts are combined into a text, which is called whole text. It is also shown in the table.

Table 5 The two Bulgarian texts used in language modeling and the whole text (both training and additional)



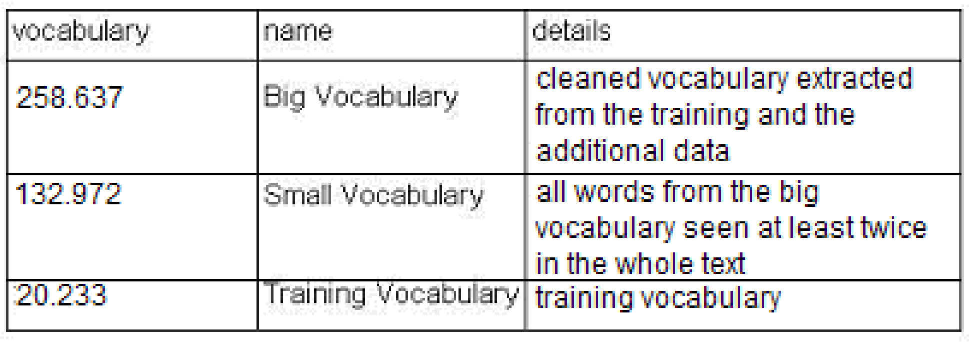
text data	running words	vocabulary
training text	120.719	20.233
additional text	8.597.380	248.993
whole text	8.718.099	249.103

The vocabulary of the training text (the training vocabulary) is clean and it is used as a vocabulary in both language modeling and decoding. The vocabulary of the additional text is not clean; it contains many digits, symbols, and foreign words, because of lack of time for manual processing of the huge text data. Since such words are not of interest and may cause problems while working with Janus, they are simply ignored. This is done by cleaning the additional vocabulary.

The cleaned additional vocabulary is now used to extend the training vocabulary. Extended vocabularies are needed in language modeling when using the additional text. They also improve the decoding (more words can be found and recognized). To create the first extended vocabulary, the BulTreeBank XML-vocabulary (13.924 words, see section 3.6) is added to the cleaned additional vocabulary. Then, the training vocabulary is added, and the result is the first extended vocabulary. The number of occurrences of each of the words in the extended vocabulary is estimated based on the Bulgarian whole text (training and additional). There are many words (about half of the extended vocabulary), which have been seen only once in the Bulgarian text. Those words are likely to be not relevant for the recognition performance. Thus, a new vocabulary is created that excludes those words. From the first extended vocabulary only those words are collected, which have occurred at least twice in the text data. Thus, the recognition during decoding is supposed to perform faster, without a noticeable difference in its precision. Special developed Tcl scripts do the extraction of vocabularies and the estimation of frequency lists of vocabulary.

Later, both extended vocabularies are used to produce their corresponding extended pronunciation dictionaries, which are needed for decoding. Both dictionaries are created via the Perl script that was previously used to create the training pronunciation dictionary. To the rules in this script, 52 more rules for acronyms are added (see subsection 5.1.1). Table 6 shows all three Bulgarian vocabularies, which are used to build Bulgarian language models, namely the training, the big (extended) and the small (extended) vocabulary. The table contains also a brief description of each vocabulary. The number of words in the small vocabulary is about half the size of the big one. As already mentioned, this means that almost half of the words in the big vocabulary have been seen only once in the whole text.

Table 6 Bulgarian vocabularies used in language modeling



vocabulary	name	details
258.637	Big Vocabulary	cleaned vocabulary extracted from the training and the additional data
132.972	Small Vocabulary	all words from the big vocabulary seen at least twice in the whole text
20.233	Training Vocabulary	training vocabulary

The same vocabularies are used for language modeling and for decoding. Decoding is performed on both development and evaluation texts and a so-called Out of Vocabulary rate (OOV) is estimated for both of them. For a given text and given vocabulary, it computes the percentage of the text (regarding its running words), that is not covered by the given vocabulary. An OOV rate of zero means that all words in the text can be found in the given vocabulary. Since the development and the evaluation text are to be decoded, it is interesting to know how both texts are covered by different vocabularies. A lower OOV rate means that the used vocabulary is better for the given text and the created language model is better. Table 7 shows computed OOV rates. The three presented vocabularies are the training, the small and the big vocabulary. OOV is estimated for both development and evaluation texts.

Table 7 OOV Rates estimated on the development and the evaluation texts using different Bulgarian vocabularies

used vocabulary:		OOV Rate (in %) for:	
name	#words	development set * running words: 15 127 * vocabulary: 4 748	evaluation set * running words: 13 904 * vocabulary: 4 796
Training	20 233	10.2	12.1
Small	132 972	2.0	2.3
Big	258 637	1.2	1.2

The results in Table 7 show that a big amount of words in both development and evaluation texts is not covered by the training vocabulary. Thus, these words are likely not to be recognized, if the training dictionary for decoding is used. Compared to the training vocabulary, the two extended vocabularies (small and big) have much lower OOV rates estimated on both texts. This simply means that a better recognition is expected, if the small or the big dictionary is used for decoding. The OOV rate is one of the main characteristics of every created language model, just like the perplexity (see section 4.3 of chapter 4). The decision which language model is better is based mainly on these two parameters – the perplexity and the OOV rate.

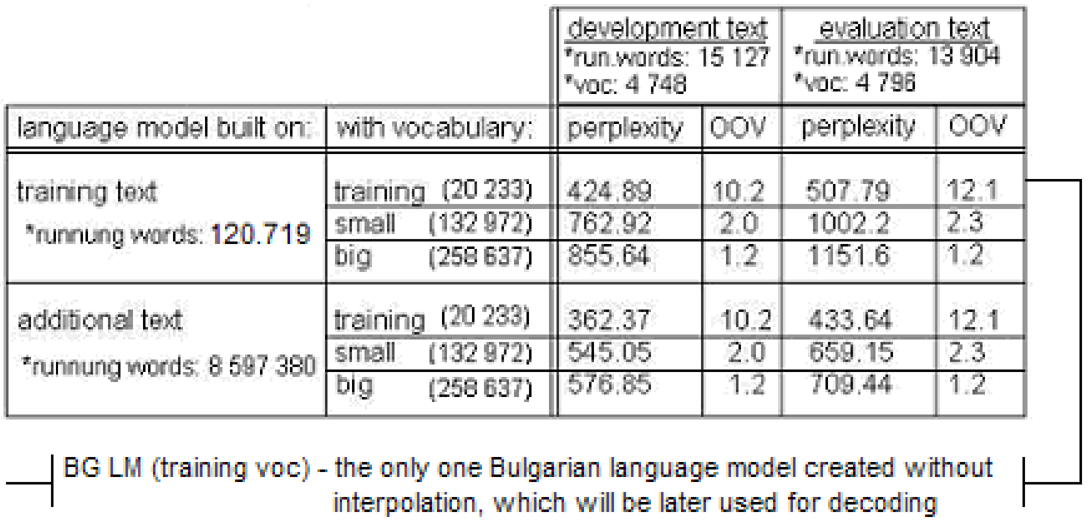
All three presented vocabularies – **training, small and big** – are relevant for further experiments. All of the three presented texts are relevant as well:

- The **additional text** is used only to create interpolated language models
- The **training text** is used to create different language models, to decode and to compute coverage and overlapping regarding Russian text
- The **whole text** is used only to compute coverage and overlapping regarding Russian text

5.2.1.2 Bulgarian language models built without interpolation

The Bulgarian n-gram language models are created via a special script, which allows the user to specify different parameters. Text, vocabulary, discounting parameter (see section 4.3 of chapter 4), and silence word are specified for the language models. Table 8 contains only Bulgarian models built without any interpolation between them. It shows the perplexity and the OOV rates of six different Bulgarian language models.

Table 8 Perplexities and OOV rates of 6 different language models estimated on both development and evaluation text



		development text *run.words: 15 127 *voc: 4 748		evaluation text *run.words: 13 904 *voc: 4 796	
language model built on:	with vocabulary:	perplexity	OOV	perplexity	OOV
training text *runnung words: 120.719	training (20 233)	424.89	10.2	507.79	12.1
	small (132 972)	762.92	2.0	1002.2	2.3
	big (258 637)	855.64	1.2	1151.6	1.2
additional text *runnung words: 8 597 380	training (20 233)	362.37	10.2	433.64	12.1
	small (132 972)	545.05	2.0	659.15	2.3
	big (258 637)	576.85	1.2	709.44	1.2

BG LM (training voc) - the only one Bulgarian language model created without interpolation, which will be later used for decoding

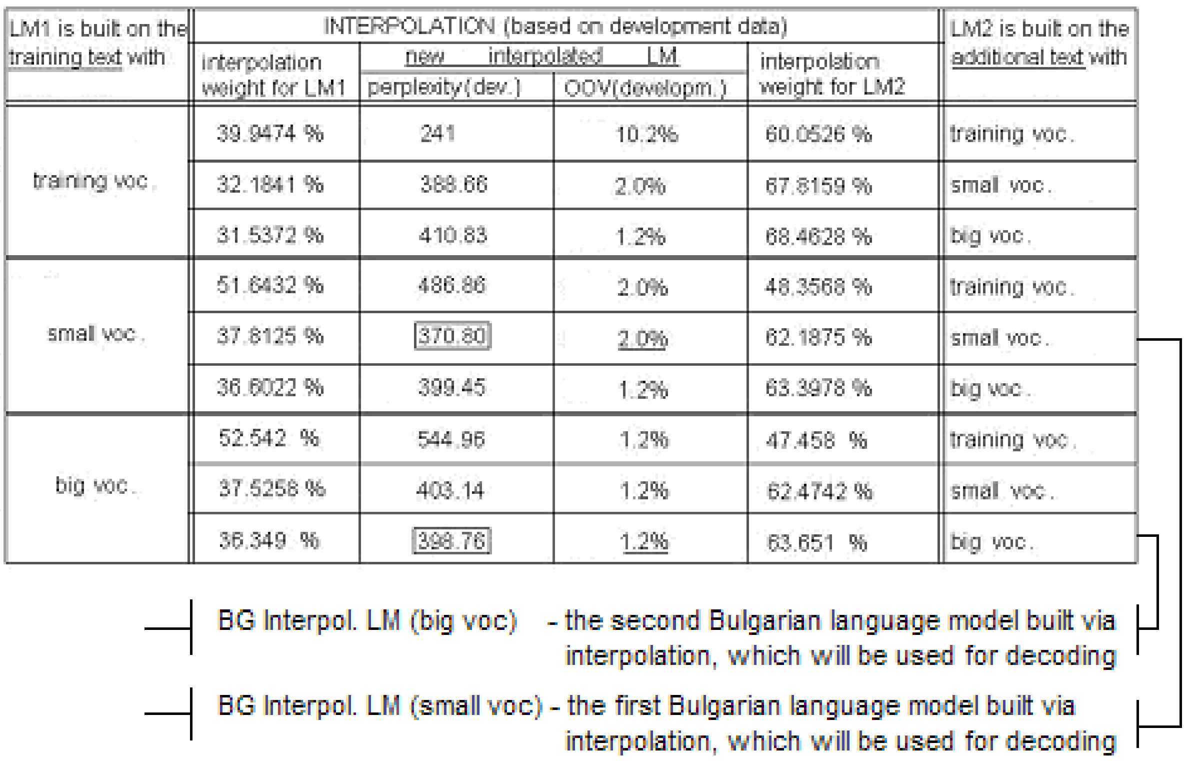
The first two columns of Table 8 specify the text and the vocabulary used to create the language model. As already mentioned, two texts (training and additional) and three vocabularies (training, small, big) are used. The number of running words in both texts and the size of the vocabularies are also given. Each line of the table represents one language model. The perplexity and the OOV rate of each six of them are given in the right half of the table. Both are estimated on the development and the evaluation text. The number of running words and the size of vocabulary for each of these texts is also given. The table reveals that language models with a lower OOV rate have a higher perplexity. In addition, the values estimated on the evaluation text are higher than the values based on the development text. Since the parameters of the recognizer are tuned regarding the development text, this will most likely result into a worse performance of recognition during evaluation. To try to improve the perplexity and the OOV rate, interpolated language models are created afterwards.

5.2.1.3 Bulgarian language models built via interpolation

Interpolation of language models is a technique that often produces new language models with lower perplexity and OOV values. First, a procedure is used to compute the best interpolation weights for the two models that are to be interpolated. These weights will create a language model with a perplexity as low as possible. The technique, which is used to estimate these weights, first assigns equal weights to both language models. Iteratively, the initial perplexity is computed and better weights are estimated. After each of the iteration steps, the perplexity of the current interpolated language model decreases. When there is no longer a significant difference between the last one and the newly estimated perplexity, the best weights have been found. Table 9 shows nine interpolated Bulgarian language models.

The first and the last columns show the two language models, which are about to be interpolated. Each of the language models described in the first column is estimated on the training text. The difference between these models is in the different vocabularies used to create them. Each of the language models described in the last column is estimated on the additional text. Again, the difference lays in the vocabularies used to create the language models. Each line in the first and the last column represents a different language model. Interpolation is performed between each two language models in a line. The optimal interpolation weights that have been estimated for each of the interpolations are given in the second and in the fifth column. The perplexity and the OOV rate are estimated based only on the development data and are shown in the third and the fourth column of the table.

Table 9 Perplexities and OOV rates of nine Bulgarian interpolated language models estimated on the development text



LM1 is built on the training text with	INTERPOLATION (based on development data)				LM2 is built on the additional text with
	interpolation weight for LM1	new interpolated LM		interpolation weight for LM2	
		perplexity(dev.)	OOV(developm.)		
training voc.	39.9474 %	241	10.2%	60.0526 %	training voc.
	32.1841 %	388.66	2.0%	67.8159 %	small voc.
	31.5372 %	410.83	1.2%	68.4628 %	big voc.
small voc.	51.6432 %	486.86	2.0%	48.3568 %	training voc.
	37.8125 %	$\boxed{370.80}$	2.0%	62.1875 %	small voc.
	36.6022 %	399.45	1.2%	63.3978 %	big voc.
big voc.	52.542 %	544.96	1.2%	47.458 %	training voc.
	37.5258 %	403.14	1.2%	62.4742 %	small voc.
	36.349 %	$\boxed{398.76}$	1.2%	63.651 %	big voc.

BG Interpol. LM (big voc) - the second Bulgarian language model built via interpolation, which will be used for decoding

BG Interpol. LM (small voc) - the first Bulgarian language model built via interpolation, which will be used for decoding

All the interpolated language models have better perplexity values and OOV rates compared to the original not interpolated language models. Because of the limited time, only few language models are used to evaluate the performance of the recognizer. Most suitable are the fifth and the ninth interpolated language models. They are further used because of the common vocabulary used by both original language models. The first of them is an interpolation between two language models, where both of them use the small vocabulary. The interpolation model is called Bulgarian interpolation using small vocabulary. The second is an interpolation between two models, where both of them use the big vocabulary. This interpolation model is called Bulgarian interpolation using big vocabulary. The perplexity values of both the interpolated language models are one of the best compared to the rest of the interpolated models. Their OOV rates are much lower than any model using only training vocabulary.

5.2.2 Russian Language Models and Mixed Models

In the previous subsection, different Bulgarian language models have been created. Some of them are now interpolated with Russian language models. More non-interpolated language models are also created using mixed vocabularies. This subsection is organized like the previous one. First, the Bulgarian and the Russian texts and vocabularies are introduced. Then, the created interpolated and not interpolated language models are presented. In addition, the overlapping of vocabulary and the coverage of text are estimated for both languages. The goal is to study the dimension of common words in both languages and to study how this affects the language modeling and the performance of the recognizer.

5.2.2.1 Texts and vocabularies

The Bulgarian training and additional texts, which have been described in subsection 5.2.1.1, are used to create some of the Russian language models and to interpolate them with Bulgarian models. The Bulgarian whole training text (both training and additional in one) is used only to estimate the coverage and overlapping between both languages. To see the details about all three Bulgarian texts, see Table 5 in subsection 5.2.1.1. The same subsection (see Table 6) contains information about the three Bulgarian vocabularies, which are now used to create Bulgarian-Russian language models. The development and the evaluation texts and vocabularies (see table 4) are used to estimate OOV rates of the language models and for decoding. In addition, they are used to estimate more common vocabularies between Bulgarian and Russian. During the Global Phone project [SW01], [Sch02] Russian text data has been collected from five e-newspapers. All these Russian articles are now added into one text file. The text has:

17.079.155 **running words**

and

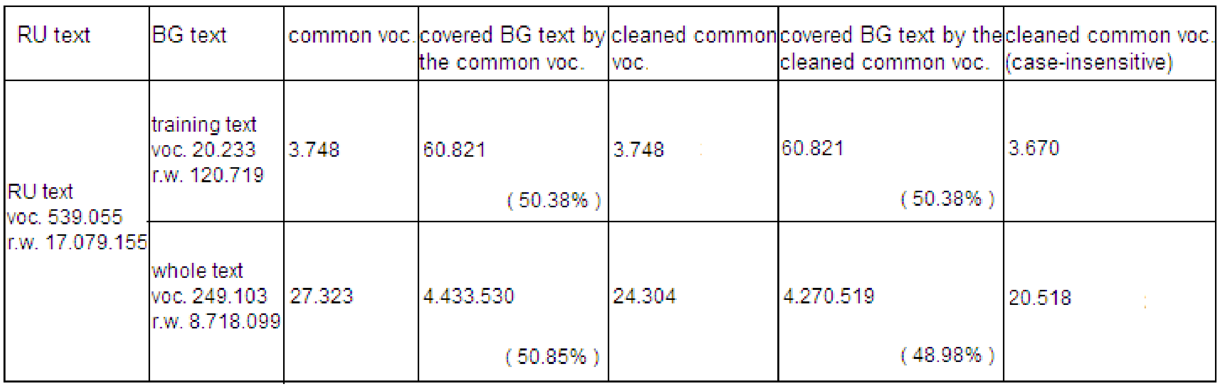
539.055 **vocabulary words.**

Few vocabularies have been created using Russian language. There is only one of them, which is later used to estimate the performance of the recognizer. As already said, the unclean Russian vocabulary contains 539.055 words. A new vocabulary is now created that includes both the cleaned Russian vocabulary and the Bulgarian training vocabulary. To suit the needs of Janus and to correspond to the Bulgarian set of phonemes, the Russian vocabulary is cleaned from words that contain any symbols, which are not part of the phoneme set of Bulgarian. The so created vocabulary is needed to find out if any improvement using Russian vocabulary in language modeling and decoding is achieved. It contains 377.787 words and is simply called “clean RU+training” vocabulary. Again, a pronunciation dictionary corresponding to the “clean RU+training” vocabulary is created. It is later used for decoding (following the principle of using the same vocabulary in the language model that will be used in the decoding too). Multiple common vocabularies are estimated using the Russian text and different Bulgarian texts and vocabularies. They are all presented in the next subsection but only one is used to create a language model later. This is the vocabulary presented in the last row of Table 11. It contains all the common words between the Russian text and the Bulgarian big vocabulary and its size is 25.717.

5.2.2.2 Overlapping and coverage

Different Bulgarian texts and vocabularies are now compared to the Russian text. The number of words in different common vocabularies as well as the coverage of Bulgarian texts by Russian (and visa versa) is estimated. Vocabularies are sometimes not clean. For example, they may contain false written words, which might be then included in the overlapping vocabulary or used to cover text. To find out if any false written words are relevant for overlapping or coverage, the type of the most frequently occurring common words is studied too.

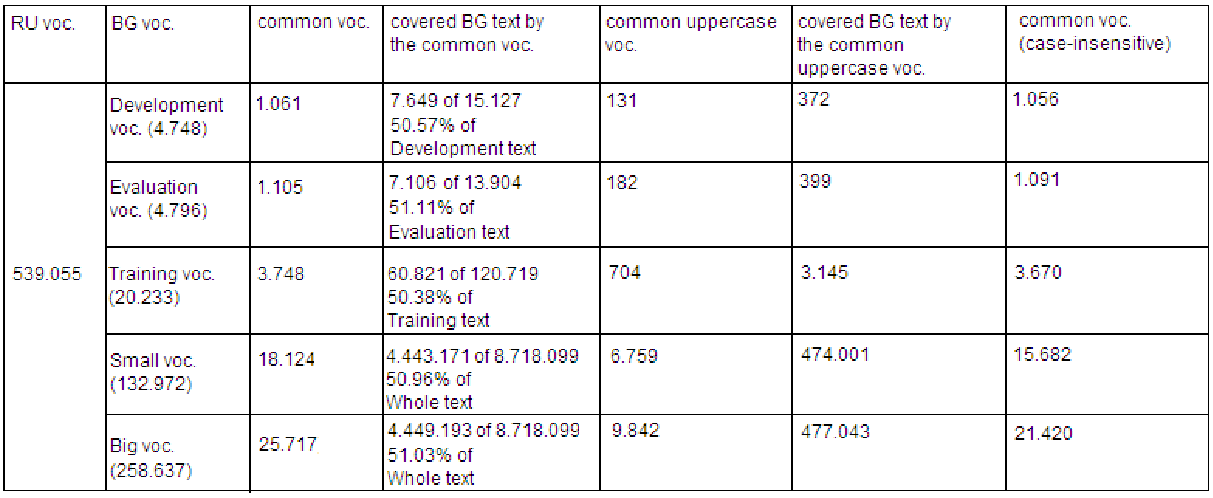
Table 10 Common vocabularies and coverage of Bulgarian data (estimated using the Russian text and two Bulgarian texts)



RU text	BG text	common voc.	covered BG text by the common voc.	cleaned common voc.	covered BG text by the cleaned common voc.	cleaned common voc. (case-insensitive)
RU text voc. 539.055 r.w. 17.079.155	training text voc. 20.233 r.w. 120.719	3.748	60.821 (50.38%)	3.748	60.821 (50.38%)	3.670
	whole text voc. 249.103 r.w. 8.718.099	27.323	4.433.530 (50.85%)	24.304	4.270.519 (48.98%)	20.518

Table 10 shows the results of comparing the Bulgarian and Russian text. More information about the Bulgarian texts can be found in the subsection 5.2.1.1. The third column shows the size of common vocabulary. The fourth column is the coverage of each of the Bulgarian texts using the corresponding common vocabulary. The fifth and the six columns represent the cleaned common vocabulary and the coverage of the Bulgarian texts when using this vocabulary. The last column simply gives information about the size of the cleaned common case-insensitive vocabularies. None of the sizes has decreased dramatically. The goal of this comparison is to show the high percentage of coverage of the Bulgarian data by Russian. The coverage by the manual cleaned common vocabulary (between the Russian and the Bulgarian whole texts) is only a little bit smaller than the coverage by the unclean common vocabulary. This means that the number of all unreal words among the covered words is so small that it does not affect the coverage statistics.

Table 11 Common vocabularies and coverage of Bulgarian data



RU voc.	BG voc.	common voc.	covered BG text by the common voc.	common uppercase voc.	covered BG text by the common uppercase voc.	common voc. (case-insensitive)
539.055	Development voc. (4.748)	1.061	7.649 of 15.127 50.57% of Development text	131	372	1.056
	Evaluation voc. (4.796)	1.105	7.106 of 13.904 51.11% of Evaluation text	182	399	1.091
	Training voc. (20.233)	3.748	60.821 of 120.719 50.38% of Training text	704	3.145	3.670
	Small voc. (132.972)	18.124	4.443.171 of 8.718.099 50.96% of Whole text	6.759	474.001	15.682
	Big voc. (258.637)	25.717	4.449.193 of 8.718.099 51.03% of Whole text	9.842	477.043	21.420

Table 11 represents the common vocabularies and coverage of different Bulgarian texts by the Russian vocabulary. Compared to the previous table, the used vocabularies are now no longer directly extracted from the text, but they are already extracted and cleaned vocabularies (see subsection 5.2.1.1). This table contains also information about the upper-case common vocabularies and how many words in the Bulgarian texts are covered by them. Again, the size of common case-insensitive vocabularies is given. The size of the covered Bulgarian text as documented in Table 11 is not significantly different from size of covered

Bulgarian text as shown in Table 10. This strengthens the conclusion that the coverage of Bulgarian texts by Russian is about 50 percents even if the unreal words have been excluded. The presented statistic about the common uppercase vocabularies tries to define a maximum for the number of names in the common vocabularies by simply assuming that this number is less than the number of uppercase vocabulary. Respectively, the coverage of Bulgarian texts by the corresponding vocabulary is also estimated.

Table 12 Common vocabularies and coverage of Russian text data (estimated using five different Bulgarian vocabularies and the Russian text)

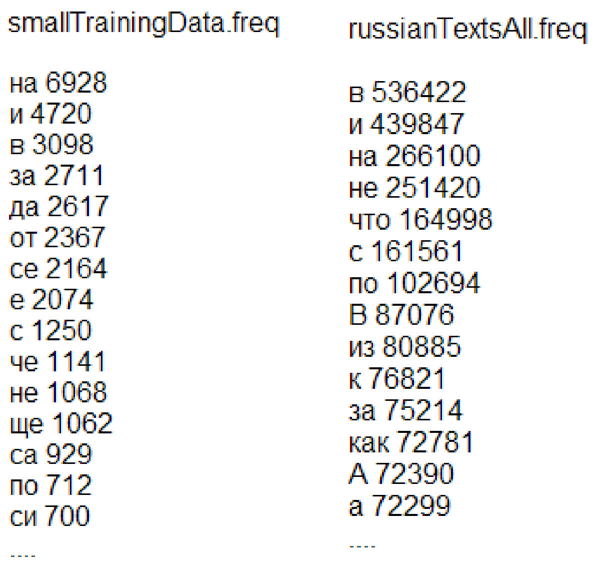
RU text	BG voc.	common voc.	RU text covered by the common voc.
vocabulary w. 539.055 running words 17.079.155	Training voc. (20.233)	3.748	3.934.246
	Small voc. (132.972)	18.124	6.097.026
	Big voc. (258.637)	25.717	6.748.325
	Whole text voc. (249.103)	27.323	7.297.075
		24.304 (cleaned)	6.848.578

Table 12 shows the coverage of Russian data by different Bulgarian vocabularies (see values from the last column). As can be observed, the amount of covered Russian text is very big, no matter which Bulgarian vocabulary is used. These results are not necessarily important for this work, but they are very interesting for further studies regarding the Bulgarian and the Russian language. The topic of such study might be the improvement of a Russian recognizer using Bulgarian data in the language modeling.

The values discussed in the previous tables indicate that a huge amount of Bulgarian texts is covered by the given Russian vocabulary. The following figures demonstrate the process of coverage of texts. To understand the values in the figures, the way they have been created has to be explained.

To understand the values in the figures, their estimation is now explained. A Perl script has been executed on both Bulgarian and Russian texts in order to compute the self-coverage of a text by its own vocabulary and the coverage of another text by the vocabulary of the first one. Four files have been created. Two of them have the ending .freq and represent two frequency lists. Figure 12 shows the frequency lists created during the estimation of the self-coverage of Russian and the coverage of the Bulgarian training text by the Russian vocabulary. The first list contains all the vocabulary words in the Bulgarian training text with the number of their occurrences in the same text. The second list shows all the words of the Russian training text with the number of their occurrences in the same text. Both lists start with the word with the highest number of occurrences. All words shown in the lists (all having a highest number of occurrences) are real words. In the Bulgarian list, all words differ from each other. In the Russian list, there are two pairs of words that are written twice – once with an uppercase and once with a lowercase letter. In both lists, the difference in number of occurrences of two words next to each other in the list is decreasing very fast. This means that if contained in the Russian vocabulary, the first words in the Bulgarian list will have the biggest responsibility for the high coverage of Bulgarian text by Russian. This is exactly what happening now, because many of the first words in both lists are the same words and have the same meaning.

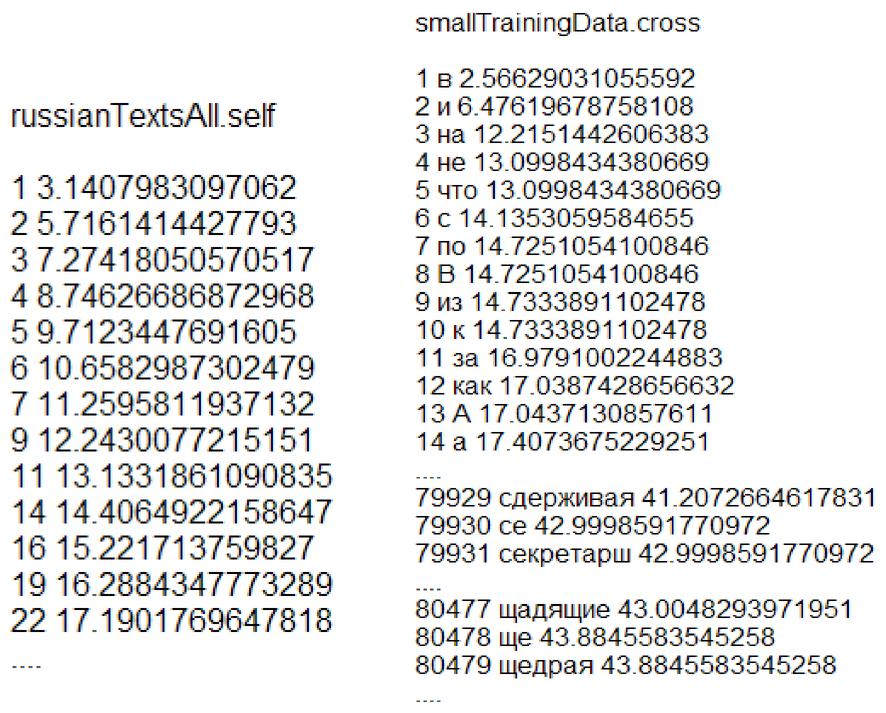
Figure 12 Frequency lists



smallTrainingData.freq	russianTextsAll.freq
на 6928	в 536422
и 4720	и 439847
в 3098	на 266100
за 2711	не 251420
да 2617	что 164998
от 2367	с 161561
се 2164	по 102694
е 2074	В 87076
с 1250	из 80885
че 1141	к 76821
не 1068	за 75214
ще 1062	как 72781
са 929	А 72390
по 712	а 72299
си 700	----

The next two files created with the execution of the coverage script have the extensions .self and .cross. The first one is the self-coverage of the Russian text by its own vocabulary. For the list of Russian vocabulary, shown in the last figure, the percent of coverage of Russian text by a single word is now added to the previous one. In the end, 100% of coverage of Russian text by its own vocabulary is achieved. These percentage values build the first line in the figure. The second file (.cross) consists of values that represent the coverage of Bulgarian training text by the Russian vocabulary. For each of the words in the Russian list, the percentage of coverage of Bulgarian text by a single Russian word is added to the previous one. The values build the second line in the figure. Figure 13 shows both lists containing the values of the self- and cross-coverage for the discussed Russian and the Bulgarian training text.

Figure 13 Self- and cross-coverage



```
russianTextsAll.self

1 3.1407983097062
2 5.7161414427793
3 7.27418050570517
4 8.74626686872968
5 9.7123447691605
6 10.6582987302479
7 11.2595811937132
9 12.2430077215151
11 13.1331861090835
14 14.4064922158647
16 15.221713759827
19 16.2884347773289
22 17.1901769647818
....
```

```
smallTrainingData.cross

1 в 2.56629031055592
2 и 6.47619678758108
3 на 12.2151442606383
4 не 13.0998434380669
5 что 13.0998434380669
6 с 14.1353059584655
7 по 14.7251054100846
8 В 14.7251054100846
9 из 14.7333891102478
10 к 14.7333891102478
11 за 16.9791002244883
12 как 17.0387428656632
13 А 17.0437130857611
14 а 17.4073675229251
....
79929 сдерживая 41.2072664617831
79930 се 42.9998591770972
79931 секретарш 42.9998591770972
....
80477 щадящие 43.0048293971951
80478 ще 43.8845583545258
80479 щедрая 43.8845583545258
....
```

With this background, the following four figures can be understood more easily. In Figure 14, the illustrated coverage of Bulgarian training text by the Russian vocabulary is 50.38%. Only about fifteen common words, which are most often seen in the Bulgarian training text, cover about 25% from it. All these words are real words - prepositions and connectives (on, and, for, to, etc).

Figure 14 Self-coverage of Russian text and coverage of Bulgarian training text by the Russian vocabulary

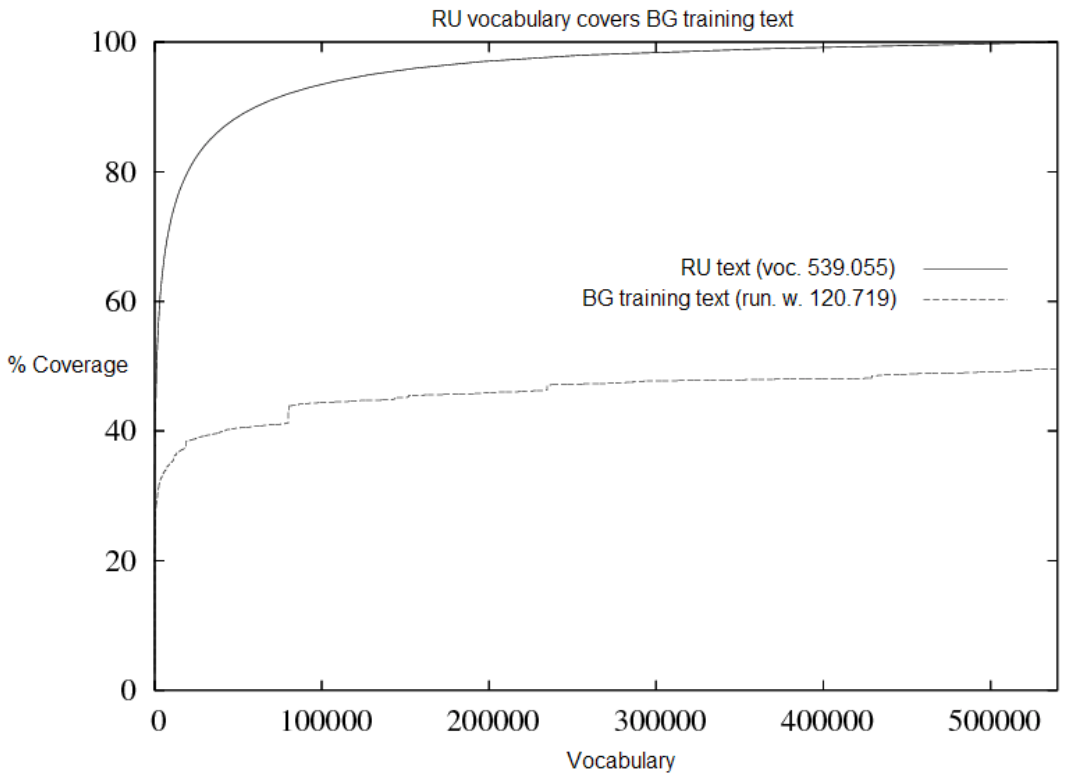
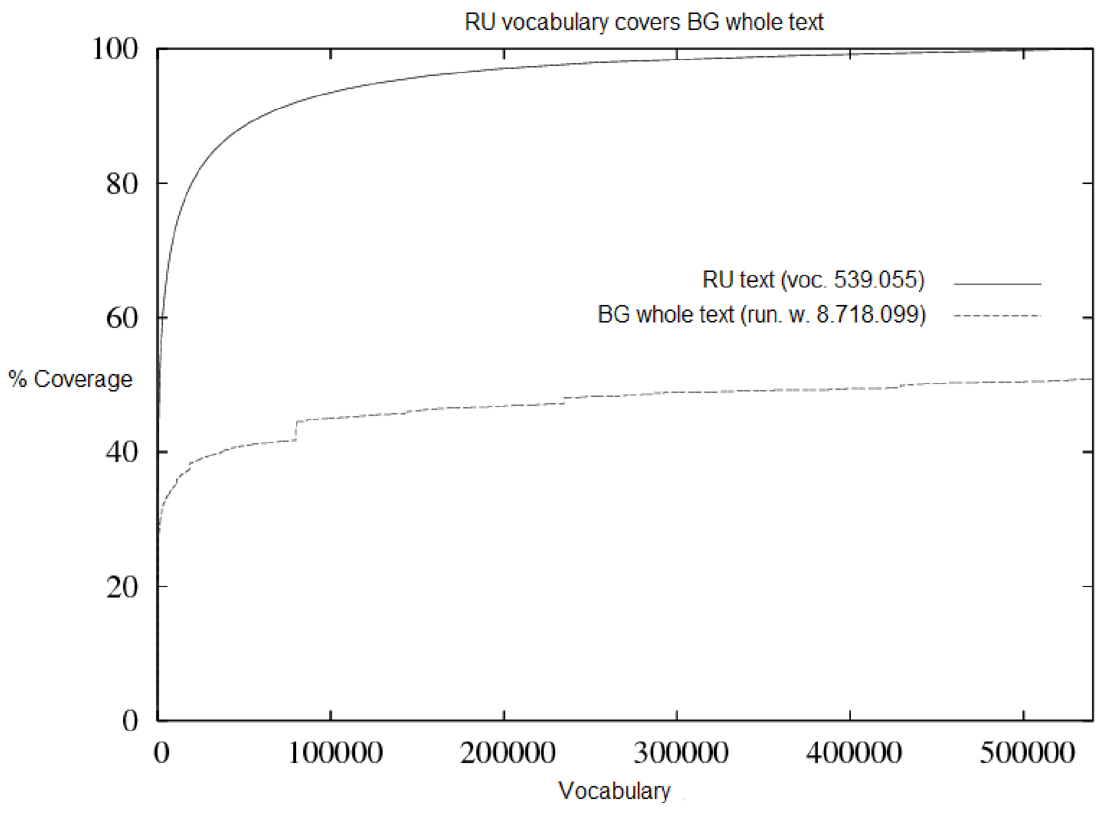


Figure 15 illustrates the 50,85% coverage of Bulgarian whole text by the Russian vocabulary. Both figures are very similar to each other, but they are still not the same. This is because the words that have the highest number of occurrences in both Bulgarian texts are more or less the same. Only about 125000 running words from about 3 million common running words in the Bulgarian whole text contain numbers or symbols, and thus, they do not play a relevant role in the coverage.

Figure 15 Self-coverage of Russian text and coverage of Bulgarian whole text by the Russian vocabulary



The next two figures (Figure 16 and Figure 17) show the coverage of the Russian text by the two different Bulgarian texts. The values are lower than the previously estimated coverage values, but they are still high for two different languages. The Bulgarian training text covers about 24% of the Russian text. The Bulgarian whole text covers about 41% of the Russian text.

Figure 16 Self-coverage of Bulgarian training text and coverage of the Russian text by the training vocabulary

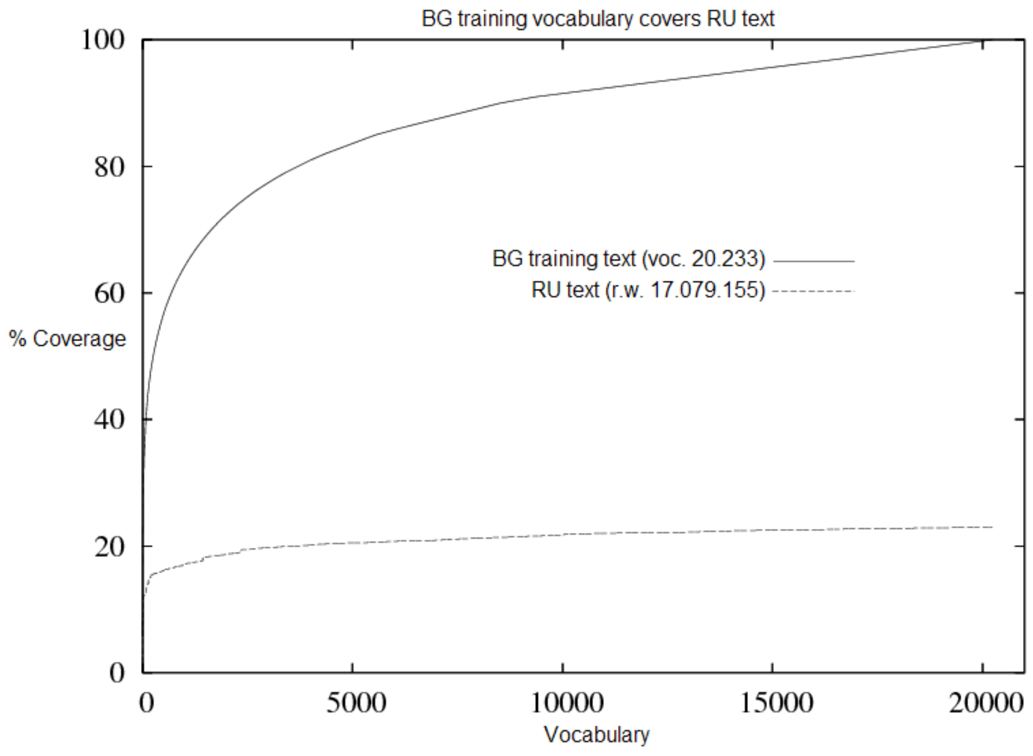
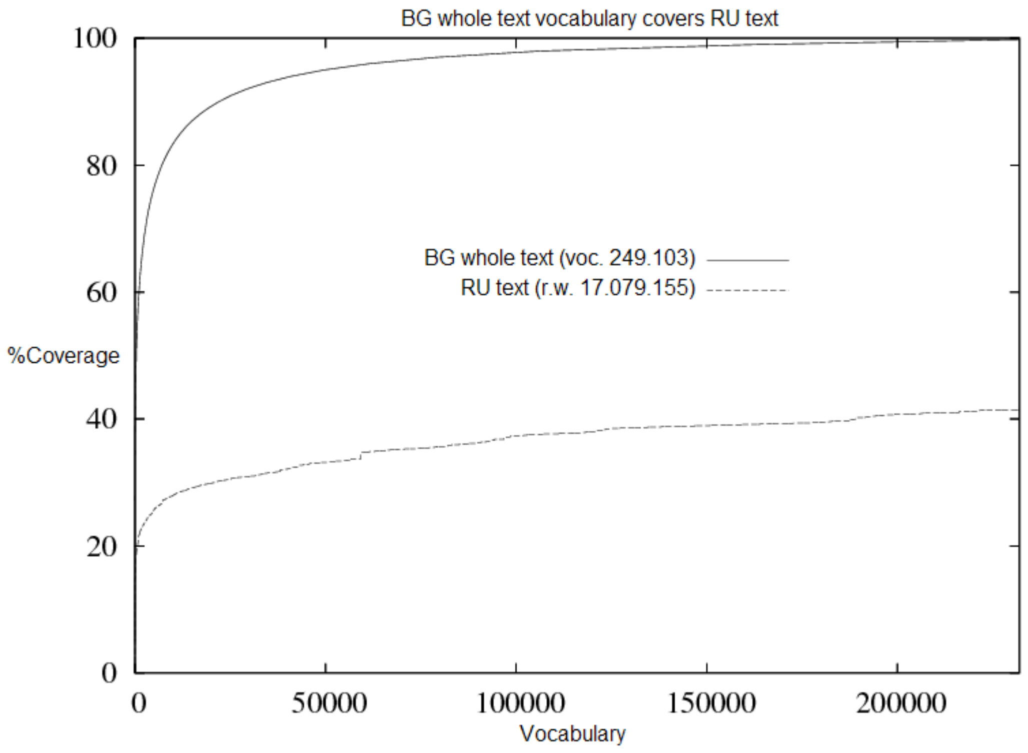


Figure 17 Self-coverage of Bulgarian whole text and coverage of Russian text by the Bulgarian whole vocabulary



The good results of the study of common vocabularies and coverage between Bulgarian and Russian motivate towards experiments with Russian data in the language modeling. However, the context of words in the texts has not been explored yet. Until now, the common vocabularies and running words have been considered without thinking about the context, within they occur. Now, the number of common bigrams and trigrams in the Bulgarian whole text and the Russian text is estimated. Tcl scripts are created in order to estimate these numbers.

The Bulgarian whole text (training + additional, 8.718.099 running words) and the Russian text (17.079.155 running words), contain:

- **27.324 common unigrams**
- **50.444 common bigrams**
- **8.277 common trigrams**

As mentioned, 24.304 of the unigrams do not contain any digits or Roman letters. Among those, 20.518 words are uniquely case-insensitive. The results in coverage of Bulgarian whole text using both vocabularies do not differ significantly – the number of covered running words is about 50%. The number of common bigrams between both Bulgarian whole text and Russian text is 50.444. Among those, 6.700 bigrams contain digits or unreal words. This does not necessarily mean that the bigrams with digits do not make sense. It means that most likely at least 40.000 bigrams contain only real words. The number of common trigrams in both Bulgarian and Russian texts is also high (8.227). About 2.500 of them contain digits. Again, this does not mean that these trigrams do not make any sense. The rest of the trigrams seem to be real combinations of words. It can be concluded that there is not only a big amount of common words but also a similarity in the way that sentences in both languages are constructed.

5.2.2.3 Mixed language models without interpolation

First, four language models are created using the Russian text and four different vocabularies. These vocabularies are shown in Table 13, where the perplexity and the OOV rate estimated on the development text set is also given.

Table 13 Language models built on Russian text data with different vocabularies

Language Model built on Russian text data with: (17079155 running words)	Perplexity (devel.)	OOV (developm.) in %
unclean Russian vocabulary (539 055)	3 799.4	50.6
clean Russian + training vocabulary (377 787)	22 874.4	8.8
big Vocabulary (258 637)	81 783	1.2
common Vocabulary (25 717)	1 585.1	50.6

These four language models are just examples of models, which are based only on Russian text. Their goal is to show that the use of such models in the Bulgarian speech recognition is not appropriate. Again, the lower perplexity indicates the higher OOV rate. The lowest perplexity is achieved when the big vocabulary and the Russian text are used. However, the so estimated perplexity for this language model is too high. The bottom line is that the perplexity of the language models that are created based on the Russian text is too high and it does not make sense to use any of these language models for Bulgarian speech recognition (without any further interpolation – see next subsection).

Next, a language model based only on Bulgarian training text is created. It is built on a mix of clean Russian and Bulgarian training vocabulary. This is made in order to compare the new language model with the one based on the same Bulgarian training text, but using only Bulgarian training vocabulary. Table 14 shows both language models.

Table 14 Language models built on Bulgarian text data with different vocabularies

Language models based on Bulgarian training text (120.719 running words) using:	Perplexity (devel.)	OOV (devel.) in %
training vocabulary (20.233)	424,89	10,2
clean Russian+training vocabulary (377.787)	482,89	8,8

BG/RU LM (clean RU+training voc) - The only one mixed language model created without interpolation.

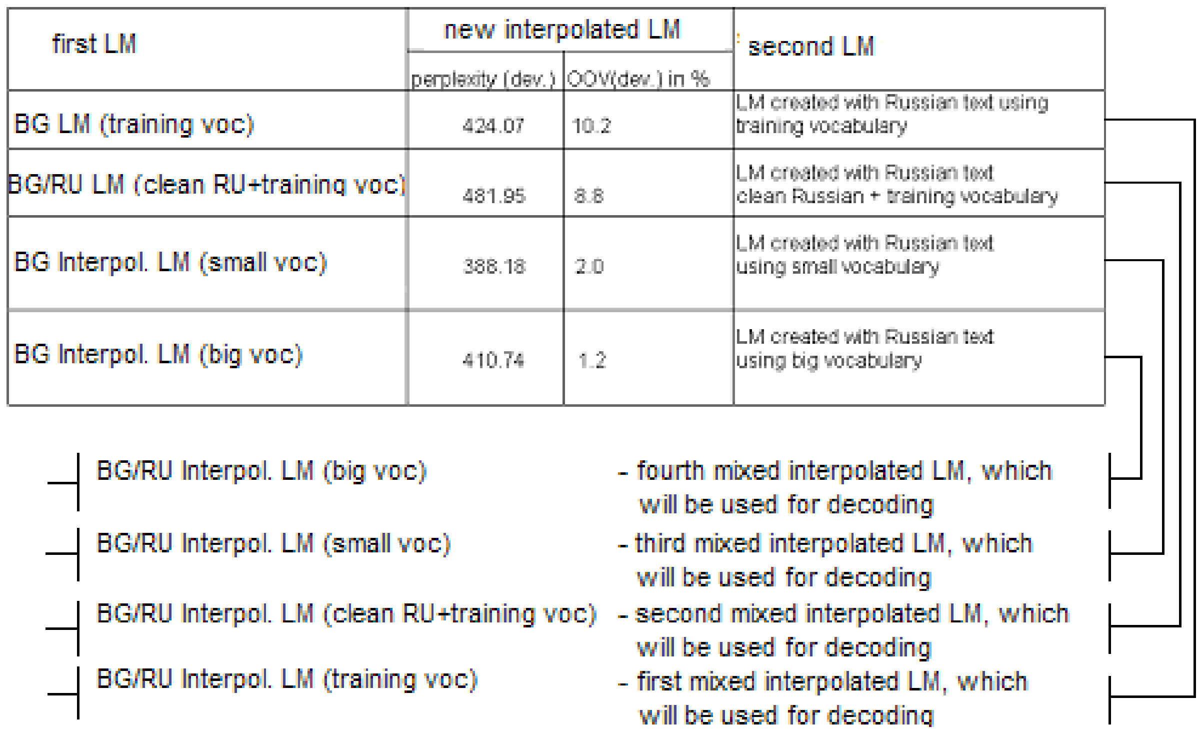
BG LM (training voc)

The first language model, which uses only Bulgarian training text data, can also be seen in Table 8. The perplexity of the second language model is now higher, but the OOV rate decreases after the clean Russian vocabulary has been added. This might lead to a better recognition. This is why these two language models will be later used in the decoding and the performance of the recognizer will be reported.

5.2.2.4 Mixed interpolated language models built via interpolation

To create interpolated language models that make use of Russian text and/or vocabulary, the interpolation technique described in subsection 5.2.1.3 is applied. Table 15 contains the four interpolated language models.

Table 15 Mixed language models created via interpolation



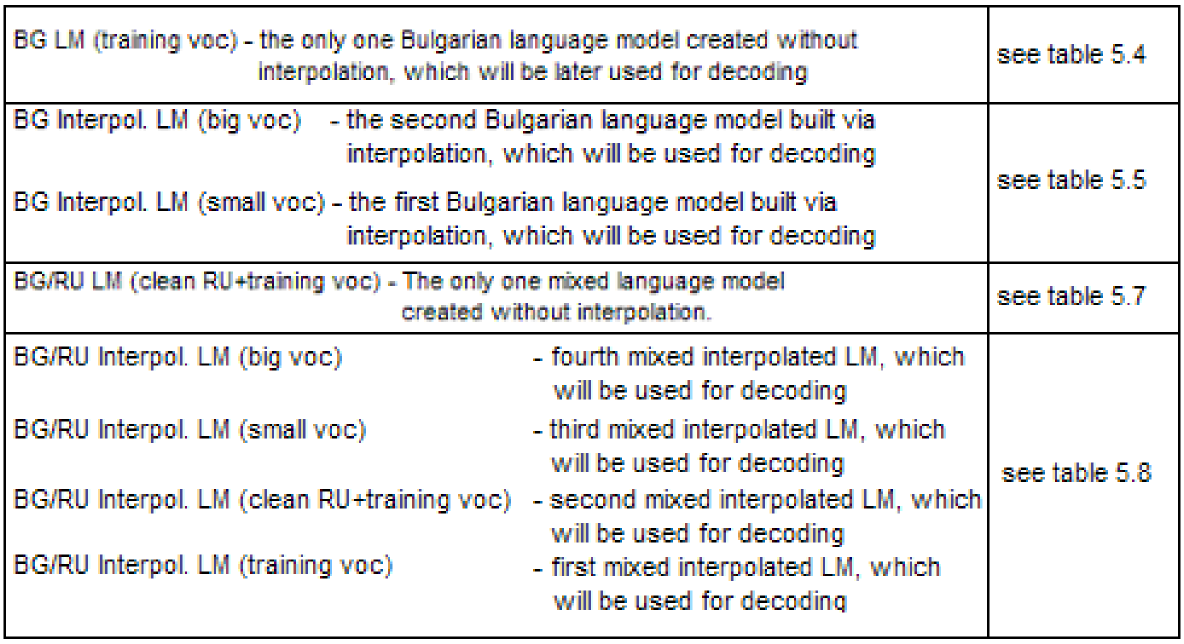
first LM	new interpolated LM		second LM
	perplexity (dev.)	OOV(dev.) in %	
BG LM (training voc)	424.07	10.2	LM created with Russian text using training vocabulary
BG/RU LM (clean RU+training voc)	481.95	8.8	LM created with Russian text clean Russian + training vocabulary
BG Interpol. LM (small voc)	388.18	2.0	LM created with Russian text using small vocabulary
BG Interpol. LM (big voc)	410.74	1.2	LM created with Russian text using big vocabulary

BG/RU Interpol. LM (big voc)	- fourth mixed interpolated LM, which will be used for decoding
BG/RU Interpol. LM (small voc)	- third mixed interpolated LM, which will be used for decoding
BG/RU Interpol. LM (clean RU+training voc)	- second mixed interpolated LM, which will be used for decoding
BG/RU Interpol. LM (training voc)	- first mixed interpolated LM, which will be used for decoding

The first and the last columns represent the two language models, which are to be interpolated. In the middle of the table, the perplexity and the OOV rate of the new language model is given. Again, it is estimated on the development text. The language models in the first column are already described in the previous subsections. The language models in the last column are only created with the goal to be used for interpolation. All of them are created based on the Russian text and using the same vocabulary, which is used in the corresponding language model in the first column. The idea is to interpolate language models, which have been created using the same vocabulary and to use this vocabulary later in the decoding. Thus, the performances of Bulgarian and mixed language models can later be compared. In all of the presented interpolations, it is easy to see that the Russian language models do not play significant role in the interpolation even though the number of common uni-, bi- and trigrams is high. The interpolation weights of the language models based on Russian data are not given in the table, but they all are under 2%.

In this section many language models have been presented. For decoding, only the eight models from Table 16 are used. The table contains a brief description of each of the models and reference to another table, which gives more information about the model.

Table 16 All language models used for decoding



BG LM (training voc) - the only one Bulgarian language model created without interpolation, which will be later used for decoding	see table 5.4
BG Interpol. LM (big voc) - the second Bulgarian language model built via interpolation, which will be used for decoding BG Interpol. LM (small voc) - the first Bulgarian language model built via interpolation, which will be used for decoding	see table 5.5
BG/RU LM (clean RU+training voc) - The only one mixed language model created without interpolation.	see table 5.7
BG/RU Interpol. LM (big voc) - fourth mixed interpolated LM, which will be used for decoding BG/RU Interpol. LM (small voc) - third mixed interpolated LM, which will be used for decoding BG/RU Interpol. LM (clean RU+training voc) - second mixed interpolated LM, which will be used for decoding BG/RU Interpol. LM (training voc) - first mixed interpolated LM, which will be used for decoding	see table 5.8

5.3 Obtained Word Error Rates

The last two sections described how the Bulgarian acoustic model and different Bulgarian or mixed language models have been created. Now, decoding is performed and the performance of the Bulgarian speech recognizer is estimated in a sequence of various experiments. First, the Bulgarian development text is used to tune the parameters of the recognizer. Once the parameters are defined, the Bulgarian evaluation text is used to produce the main values of performance of the recognizer.

5.3.1 Decoding

Section 4.3 contains a brief introduction in decoding. In this work, decoding has been performed via a Janus-script. The three parameters, which have to be tuned, are lz, lp and fillPen. The first one – lz – represents the weight of the language model vs. the acoustic model, the second one – lp – is the word penalty, which normalizes the different lengths of the observed word sequences, and the last one – fillPen – is the filler penalty, which controls the number of occurrences of the filler word (silence SIL) in the hypotheses. In the same Janus-script the path to a file is specified that contains the IDs of the speakers in the database, which participate in the decoding (these can be the people in the development or in the evaluation set). First, the development set is used to tune the three parameters and later, the performance of the recognizer is estimated on the evaluation set. In the decoding-script, the same description file (desc.tcl) is specified as was used in the training process. Again, there are paths to adjust in it, but this time, the most important paths are the paths to the codebooks (acoustic model), to the language model, to the vocabulary and the dictionary, which we use for decoding and creating hypotheses.

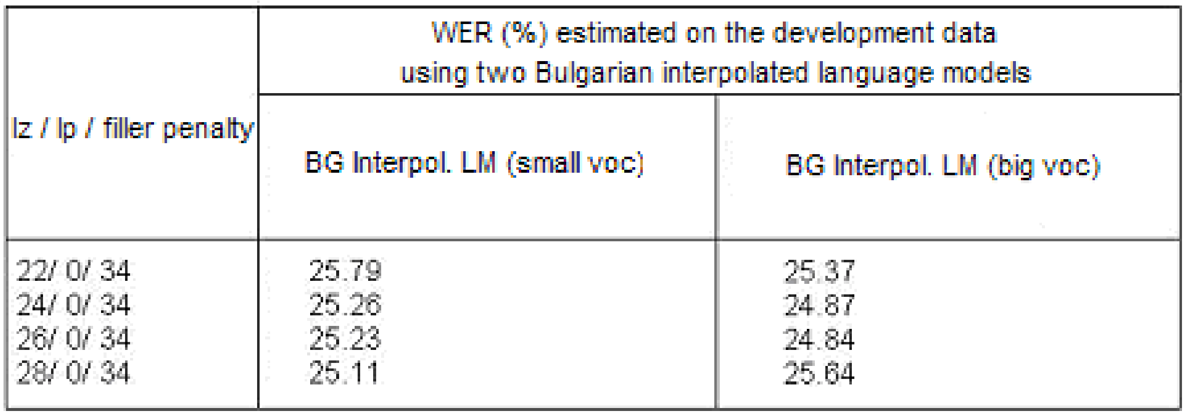
The decoding script creates a directory, where all final hypotheses are written into one text file. The hypothesis of each of the sentence is written onto a different line in this file. The score of each hypothesis (the minus log of the probability) and the ID of the corresponding sentence are given too.

After the hypotheses are written into a text file, another Janus-script is used to compute the word error rate (see section 4.4). The WER is estimated for every sentence separately and in the end, the mean WER is computed. A log file is created, which contains each of the hypotheses and its corresponding original sentence. For every such pair of sentences the number of inserted, deleted and replaced words, and the WER are estimated. Again, a mean WER is estimated for all the sentences.

5.3.2 Results

The first experiment aims to find the best parameters (lz/ lp/ filler penalty) for the Bulgarian speech recognizer. Table 17 presents the final results of this experiment. The development data is decoded using different parameters which can be seen in the first column of the table. The decoding is performed twice for each of the parameter sets (using two different Bulgarian language models) and the WER is estimated. The first language model is the “BG Interpol. LM (small voc)” and the second one is the “BG Interpol. LM (big voc)” - see Table 16. The only difference between the two language models is the size of vocabularies used to create them (respectively pronunciation dictionaries used for decoding). The second one uses all of the available Bulgarian vocabulary (big vocabulary) and the first one uses only those words that are seen more than once in the Bulgarian whole text (small vocabulary).

Table 17 Experiment to tune the Bulgarian speech recognizer - WER



lz / lp / filler penalty	WER (%) estimated on the development data using two Bulgarian interpolated language models	
	BG Interpol. LM (small voc)	BG Interpol. LM (big voc)
22/ 0/ 34	25.79	25.37
24/ 0/ 34	25.26	24.87
26/ 0/ 34	25.23	24.84
28/ 0/ 34	25.11	25.64

The best parameters of the recognizer are different for each of the two language models. The WERs computed using the second language model are better on average compared to using the first language model (even though not significantly). The best WER of all is 24.84% and it is estimated using the second language model and the parameters (26/ 0/ 34). Even if the perplexity of the second language model is higher than the first one, its OOV rate is lower, which explains its better performance.

The estimated parameters are left unchanged in the second experiment where six different language models are used to estimate the performance of the recognizer on evaluation text. All of those language models have in common that they are created using only Bulgarian vocabularies. The three well known vocabularies are the training, the small and the big vocabulary. Three of the six language models are based on Bulgarian text and three of them on the Bulgarian and Russian texts together. The idea is to estimate the performance of the recognizer using each of the three different Bulgarian language models on the evaluation text and compare it to the performance of the recognizer using the three corresponding mixed language model (always the same Bulgarian vocabularies are used). Each of the six language models can be found in Table 16. The first line of Table 18 shows the results (WER) of the evaluation using the three different Bulgarian language models (no Russian at all). Each column indicates the use of a different Bulgarian vocabulary. The second line shows the results of evaluation using the three corresponding mixed language models, built using each of the three Bulgarian vocabularies. All language models are described in section 5.2 of this chapter.

Table 18 Experiment with Bulgarian and mixed language models – WER estimated on evaluation data (only Bulgarian vocabularies are used)

text used for the language model:	Bulgarian vocabulary used for the language model:		
	Training	Small	Big
Bulgarian	41,031 BG LM (training voc)	27,337 BG interpol. LM (small voc)	26,567 BG interpol. LM (big voc)
Bulgarian and Russian	41,124 BG/RU Interpol. LM (training voc)	27,366 BG/RU Interpol. LM (small voc)	27,229 BG/RU Interpol. LM (big voc)

Table 18 studies how adding Russian text in the language models without adding its vocabulary influences the performance of recognition. On the one hand, the tendency of improvement when using a bigger vocabulary does not change. On the other hand, there is no improvement when using any of the mixed language models. What can be noticed is that the difference in WER is smaller when a smaller Bulgarian text and vocabulary are available in the language model. This is why the next experiment concentrates on the Bulgarian training text and vocabulary.

Table 19 presents the last experiment performed in this work. Now, not only Russian text is added in the language models, but also its cleaned vocabulary (clean RU+training voc). The first line presents the evaluation of the recognizer using two language models based on the same Bulgarian training text, but using two different vocabularies. In the first column always the Bulgarian training vocabulary is used and in the second – the clean RU+training vocabulary. In the second line, the two new language models are now based on the Bulgarian training text and the Russian text together, and the difference between the language models is again in the vocabulary used. The idea is to study how the performance of the recognizer is now influenced by the use of Russian vocabulary. All four language models can be seen in Table 16.

Table 19 Experiment with Bulgarian and mixed language models – WER on evaluation data (Bulgarian and Russian vocabularies are used)

text used for the language model:	Vocabulary used for the language model: (both Bulgarian and Russian included)	
	BG training voc.	clean RU+BG training voc.
Bulgarian	41,031 BG LM (training voc)	39,174 BG/RU LM (clean RU+training voc)
Bulgarian and Russian	41,124 BG/RU Interpol. LM (training voc)	38,938 BG/RU Interpol. LM (clean RU+training voc)

The results in Table 19 indicate that the use of Russian vocabulary helps no matter if only Bulgarian text or Bulgarian and Russian texts together are used to create language models. The best result is achieved when both Russian text and Russian vocabulary are used. The WER in this case is 38.938% compared to 41.031% WER when using only Bulgarian text and vocabulary. In summary, the use of Russian text in the language model is more appropriate when less Bulgarian text data is available. However, an improvement of the speech recognition is achieved when not only the Russian text, but also its vocabulary is used to create a language model. More detailed experiments have to be performed to reveal the real power of using mixed language models to improve the performance of recognition.

6. Summary

Speech recognition is still a major area of research, and the complexity of algorithms used to create speech recognition systems is constantly increasing. Sophisticated research is being performed in this field all over the world and addresses different problems. This report did not aim at dealing with every detail of the algorithms used to build the Bulgarian recognizer. It described an entire process - studying the characteristics of the language, collecting and preparing speech and text data, initializing and training acoustic models and experimenting with different Bulgarian and mixed language models. The so created Bulgarian speech recognizer achieved a best Word Error Rate of recognition of 24.84% estimated on development text and of 26.57% estimated on evaluation text. In both cases, Bulgarian language models have been used. Next, a language close to Bulgarian is presented – the Russian language. Common vocabularies and coverage between both languages are estimated and analyzed. Mixed language models are created using different (mixed) texts and vocabularies and a greatest benefit for Bulgarian recognition is established when using a language model based on both Russian and Bulgarian texts, and at the same time including the Russian vocabulary. The large number of common words and common word sequences in both texts turned out to be more helpful for a smaller amount of Bulgarian text available. For example, the 41.03% WER estimated using only Bulgarian is decreased to 38.94% by adding both Russian text and vocabulary. The achieved results motivate to continue the research and experiments using more than one language for language modeling. When further benefits are established, this will allow the creation of better speech recognizers for languages, which do not have much text and speech data available.

Annex A

Mapping rules (taken out from the Perl script, which creates a pronunciation dictionary).

```
    $word =~ s/АиS/ CZ_e  CZ_j  CZ_i  CZ_e  CZ_s /g;
    $word =~ s/BBC/ CZ_b  CZ_i  CZ_b  CZ_i  CZ_s  CZ_i /g;
    $word =~ s/DHL/ CZ_d  CZ_i  CZ_e  CZ_j  CZ_ch  CZ_e  CZ_l /g;
    $word =~ s/EADS/ CZ_e  CZ_a  CZ_d  CZ_e  CZ_e  CZ_s /g;
    $word =~ s/GPS-ът/ CZ_dzh  CZ_i  CZ_p  CZ_i  CZ_e  CZ_s  CZ_Y  CZ_t /g;
    $word =~ s/GPS/ CZ_dzh  CZ_i  CZ_p  CZ_i  CZ_e  CZ_s /g;
    $word =~ s/GSM/ CZ_dzh  CZ_i  CZ_e  CZ_s  CZ_e  CZ_m /g;
    $word =~ s/IBM/ CZ_a  CZ_j CZ_b  CZ_i  CZ_e  CZ_m /g;
    $word =~ s/IDS/ CZ_a  CZ_j  CZ_d  CZ_i  CZ_e  CZ_s /g;
    $word =~ s/IQ/ CZ_a CZ_j  CZ_k  CZ_ju /g;
    $word =~ s/MG/ CZ_e  CZ_m  CZ_dzh  CZ_i /g;
    $word =~ s/Siemens/ CZ_s  CZ_i  CZ_m  CZ_e  CZ_n  CZ_s /g;
    $word =~ s/PR/ CZ_p  CZ_i  CZ_a  CZ_r /g;
    $word =~ s/SBS/ CZ_e  CZ_s  CZ_b  CZ_i  CZ_e  CZ_s /g;
    $word =~ s/SMS-и/ CZ_e  CZ_s  CZ_e  CZ_m  CZ_e  CZ_s  CZ_i /g;
    $word =~ s/i-България/ CZ_i  CZ_b  CZ_Y  CZ_l  CZ_g  CZ_a  CZ_r  CZ_i  CZ_ja
/g;
      $word =~ s/БЗНС-Обединен/ CZ_b CZ_e CZ_z CZ_e CZ_n CZ_e CZ_s CZ_e
CZ_o CZ_b CZ_e CZ_d CZ_i CZ_n CZ_e CZ_n /g;
      $word =~ s/ЕООД-та/ CZ_e CZ_o CZ_o CZ_d CZ_e CZ_t CZ_a /g;
      $word =~ s/ЕООД/ CZ_e CZ_o CZ_o CZ_d CZ_e /g;
      $word =~ s/БГНЕС/ CZ_b CZ_e CZ_g CZ_e CZ_n CZ_e CZ_s /g;
      $word =~ s/ВМРО/ CZ_v CZ_e CZ_m CZ_e CZ_r CZ_e CZ_o /g;
      $word =~ s/БЗНС/ CZ_b CZ_e CZ_z CZ_e CZ_n CZ_e CZ_s CZ_e /g ;
      $word =~ s/НДСВ/ CZ_e CZ_n CZ_d CZ_e CZ_s CZ_e CZ_v CZ_e /g;
      $word =~ s/НСБОП/ CZ_n CZ_e CZ_s CZ_e CZ_b CZ_o CZ_p /g;
      $word =~ s/РДВР-Добрич/ CZ_r CZ_e CZ_d CZ_e CZ_v CZ_e CZ_r CZ_e CZ_d
CZ_o CZ_b CZ_r CZ_i CZ_ch /g;
      $word =~ s/РДВР/ CZ_r CZ_e CZ_d CZ_e CZ_v CZ_e CZ_r CZ_e /g;
      $word =~ s/РМД-тата/ CZ_e CZ_r CZ_e CZ_m CZ_e CZ_d CZ_e CZ_t CZ_a
CZ_t CZ_a /g;
      $word =~ s/ЕАД-та/ CZ_e CZ_a CZ_d CZ_e CZ_t CZ_a /g;
      $word =~ s/МТел/ CZ_e CZ_m CZ_t CZ_e CZ_l /g;
      $word =~ s/ДРОМ/ CZ_d CZ_e CZ_r CZ_o CZ_m /g;
      $word =~ s/КНСБ/ CZ_k CZ_a CZ_n CZ_e CZ_s CZ_e CZ_b CZ_e /g;
      $word =~ s/Логос-ТМ/ CZ_l CZ_o CZ_g CZ_o CZ_s CZ_t CZ_i CZ_e CZ_m /g;
      $word =~ s/СССР/ CZ_s CZ_e CZ_s CZ_e CZ_s CZ_e e CZ_r /g;
```

```
  $word =~ s/ЕАД/ CZ_e CZ_a CZ_d CZ_e /g;
  $word =~ s/РМД/ CZ_e CZ_r CZ_e CZ_m CZ_d CZ_e /g;
  $word =~ s/БМД/ CZ_b CZ_e CZ_e CZ_m CZ_d CZ_e /g;
  $word =~ s/БСП/ CZ_b CZ_e CZ_s CZ_e CZ_p CZ_e /g;
  $word =~ s/БТА/ CZ_b CZ_e CZ_t CZ_e CZ_a /g;
  $word =~ s/БТВ/ CZ_b CZ_i CZ_t CZ_i CZ_v CZ_i /g;
  $word =~ s/БТР-и/ CZ_b CZ_e CZ_t CZ_e CZ_e CZ_r CZ_i /g;
  $word =~ s/БТР/ CZ_b CZ_e CZ_t CZ_e CZ_e CZ_r /g;
  $word =~ s/ВиК/ CZ_v CZ_e CZ_i CZ_k CZ_a /g;
  $word =~ s/ДДС/ CZ_d CZ_e CZ_d CZ_e CZ_s CZ_e /g;
  $word =~ s/ДЗИ/ CZ_d CZ_e CZ_z CZ_e CZ_i /g;
  $word =~ s/ДЗУ/ CZ_d CZ_e CZ_z CZ_e CZ_u /g;
  $word =~ s/ДНК/ CZ_d CZ_e CZ_e CZ_n CZ_k CZ_a /g;
  $word =~ s/ДПС/ CZ_d CZ_e CZ_p CZ_e CZ_s CZ_e /g;
  $word =~ s/БДЖ/ CZ_b CZ_e CZ_d CZ_e CZ_zh CZ_e /g;
  $word =~ s/ДСБ/ CZ_d CZ_e CZ_s CZ_e CZ_b CZ_e /g;
  $word =~ s/ЕНП/ CZ_e CZ_n CZ_e CZ_p CZ_e /g;
  $word =~ s/МВР/ CZ_m CZ_e CZ_v CZ_e CZ_r CZ_e /g ;
  $word =~ s/МВФ/ CZ_m CZ_e CZ_v CZ_e CZ_f CZ_e /g;
  $word =~ s/МПС/ CZ_m CZ_e CZ_p CZ_e CZ_s CZ_e /g;
  $word =~ s/НСИ/ CZ_n CZ_e CZ_s CZ_e CZ_i /g;
  $word =~ s/ОДС/ CZ_o CZ_d CZ_e CZ_s CZ_e /g;
  $word =~ s/ОМО/ CZ_o CZ_m CZ_e CZ_o /g;
  $word =~ s/ОНД/ CZ_o CZ_e CZ_n CZ_d CZ_e /g;
  $word =~ s/ООД/ CZ_o CZ_o CZ_d CZ_e /g;
  $word =~ s/ООН/ CZ_o CZ_o CZ_n CZ_e /g;
  $word =~ s/ПЗУ/ CZ_p CZ_e CZ_z CZ_e CZ_u /g;
  $word =~ s/ЦРУ/ CZ_tsh CZ_e CZ_r CZ_e CZ_u /g;
  $word =~ s/СДС/ CZ_s CZ_e CZ_d CZ_e CZ_s CZ_e /g;
  $word =~ s/ЖП/ CZ_zh CZ_e CZ_p CZ_e /g;
  $word =~ s/ДП/ CZ_d CZ_e CZ_p CZ_e /g;
  $word =~ s/ТВ/ CZ_t CZ_i CZ_v CZ_i /g;
  $word =~ s/ТМ/ CZ_t CZ_i CZ_e CZ_m /g;
  $word =~ s/АП/ CZ_a CZ_p CZ_e /g;
  $word =~ s/БА/ CZ_b CZ_e CZ_a /g;
$word =~ s/Ч/ CZ_ch /g;
$word =~ s/ч/ CZ_ch /g;
$word =~ s/ДЗ/ CZ_dz /g;
$word =~ s/Дз/ CZ_dz /g;
$word =~ s/дз/ CZ_dz /g;
$word =~ s/ДЖ/ CZ_dzh /g;
$word =~ s/Дж/ CZ_dzh /g;
$word =~ s/дж/ CZ_dzh /g;
$word =~ s/Щ/ CZ_sh  CZ_t /g;
$word =~ s/щ/ CZ_sh  CZ_t /g;
$word =~ s/Ц/ CZ_tsh /g;
$word =~ s/ц/ CZ_tsh /g;
$word =~ s/Ш/ CZ_sh /g;
$word =~ s/ш/ CZ_sh /g;
$word =~ s/Ж/ CZ_zh /g;
```

```
$word =~ s/ж/ CZ_zh /g
$word =~ s/БЮ/ CZ_bj  CZ_u /g;
$word =~ s/БЯ/ CZ_bj  CZ_a /g;
$word =~ s/БЬО/ CZ_bj  CZ_o /g;
$word =~ s/Бю/ CZ_bj  CZ_u /g;
$word =~ s/Бя/ CZ_bj  CZ_a /g;
$word =~ s/Бьо/ CZ_bj  CZ_o /g;
$word =~ s/бю/ CZ_bj  CZ_u /g;
$word =~ s/бя/ CZ_bj  CZ_a /g;
$word =~ s/бьо/ CZ_bj  CZ_o /g;
$word =~ s/ВЮ/ CZ_vj  CZ_u /g;
$word =~ s/ВЯ/ CZ_vj  CZ_a /g;
$word =~ s/ВЬО/ CZ_vj  CZ_o /g;
$word =~ s/Вю/ CZ_vj  CZ_u /g;
$word =~ s/Вя/ CZ_vj  CZ_a /g;
$word =~ s/Вьо/ CZ_vj  CZ_o /g;
$word =~ s/вю/ CZ_vj  CZ_u /g;
$word =~ s/вя/ CZ_vj  CZ_a /g;
$word =~ s/вьо/ CZ_vj  CZ_o /g;
$word =~ s/ГЮ/ CZ_gj  CZ_u /g;
$word =~ s/ГЯ/ CZ_gj  CZ_a /g;
$word =~ s/ГЬО/ CZ_gj  CZ_o /g;
$word =~ s/Гю/ CZ_gj  CZ_u /g;
$word =~ s/Гя/ CZ_gj  CZ_a /g;
$word =~ s/Гьо/ CZ_gj  CZ_o /g;
$word =~ s/гю/ CZ_gj  CZ_u /g;
$word =~ s/гя/ CZ_gj  CZ_a /g;
$word =~ s/гьо/ CZ_gj  CZ_o /g;
$word =~ s/ДЮ/ CZ_dj  CZ_u /g;
$word =~ s/ДЯ/ CZ_dj  CZ_a /g;
$word =~ s/ДЬО/ CZ_dj  CZ_o /g;
$word =~ s/Дю/ CZ_dj  CZ_u /g;
$word =~ s/Дя/ CZ_dj  CZ_a /g;
$word =~ s/Дьо/ CZ_dj  CZ_o /g;
$word =~ s/дю/ CZ_dj  CZ_u /g;
$word =~ s/дя/ CZ_dj  CZ_a /g;
$word =~ s/дьо/ CZ_dj  CZ_o /g;
$word =~ s/КЮ/ CZ_kj  CZ_u /g;
$word =~ s/КЯ/ CZ_kj  CZ_a /g;
$word =~ s/КЬО/ CZ_kj  CZ_o /g;
$word =~ s/Кю/ CZ_kj  CZ_u /g;
$word =~ s/Кя/ CZ_kj  CZ_a /g;
$word =~ s/Кьо/ CZ_kj  CZ_o /g;
$word =~ s/кю/ CZ_kj  CZ_u /g;
$word =~ s/кя/ CZ_kj  CZ_a /g;
$word =~ s/кьо/ CZ_kj  CZ_o /g;
$word =~ s/ЛЮ/ CZ_lj  CZ_u /g;
$word =~ s/ЛЯ/ CZ_lj  CZ_a /g;
$word =~ s/ЛЬО/ CZ_lj  CZ_o /g;
$word =~ s/Лю/ CZ_lj  CZ_u /g;
```

```
$word =~ s/Ля/ CZ_lj  CZ_a /g;
$word =~ s/Льо/ CZ_lj  CZ_o /g;
$word =~ s/лю/ CZ_lj  CZ_u /g;
$word =~ s/ля/ CZ_lj  CZ_a /g;
$word =~ s/льо/ CZ_lj  CZ_o /g;
$word =~ s/МЮ/ CZ_mj  CZ_u /g;
$word =~ s/МЯ/ CZ_mj  CZ_a /g;
$word =~ s/МЬО/ CZ_mj  CZ_o /g;
$word =~ s/Мю/ CZ_mj  CZ_u /g;
$word =~ s/Мя/ CZ_mj  CZ_a /g;
$word =~ s/Мьо/ CZ_mj  CZ_o /g;
$word =~ s/мю/ CZ_mj  CZ_u /g;
$word =~ s/мя/ CZ_mj  CZ_a /g;
$word =~ s/мьо/ CZ_mj  CZ_o /g;
$word =~ s/НЮ/ CZ_nj  CZ_u /g;
$word =~ s/НЯ/ CZ_nj  CZ_a /g;
$word =~ s/НЬО/ CZ_nj  CZ_o /g;
$word =~ s/Ню/ CZ_nj  CZ_u /g;
$word =~ s/Ня/ CZ_nj  CZ_a /g;
$word =~ s/Ньо/ CZ_nj  CZ_o /g;
$word =~ s/ню/ CZ_nj  CZ_u /g;
$word =~ s/ня/ CZ_nj  CZ_a /g;
$word =~ s/ньо/ CZ_nj  CZ_o /g;
$word =~ s/ПЮ/ CZ_pj  CZ_u /g;
$word =~ s/ПЯ/ CZ_pj  CZ_a /g;
$word =~ s/ПЬО/ CZ_pj  CZ_o /g;
$word =~ s/Пю/ CZ_pj  CZ_u /g;
$word =~ s/Пя/ CZ_pj  CZ_a /g;
$word =~ s/Пьо/ CZ_pj  CZ_o /g;
$word =~ s/пю/ CZ_pj  CZ_u /g;
$word =~ s/пя/ CZ_pj  CZ_a /g;
$word =~ s/пьо/ CZ_pj  CZ_o /g;
$word =~ s/РЮ/ CZ_rj  CZ_u /g;
$word =~ s/РЯ/ CZ_rj  CZ_a /g;
$word =~ s/РЬО/ CZ_rj  CZ_o /g;
$word =~ s/Рю/ CZ_rj  CZ_u /g;
$word =~ s/Ря/ CZ_rj  CZ_a /g;
$word =~ s/Рьо/ CZ_rj  CZ_o /g;
$word =~ s/рю/ CZ_rj  CZ_u /g;
$word =~ s/ря/ CZ_rj  CZ_a /g;
$word =~ s/рьо/ CZ_rj  CZ_o /g;
$word =~ s/СЮ/ CZ_sj  CZ_u /g;
$word =~ s/СЯ/ CZ_sj  CZ_a /g;
$word =~ s/СЬО/ CZ_sj  CZ_o /g;
$word =~ s/Сю/ CZ_sj  CZ_u /g;
$word =~ s/Ся/ CZ_sj  CZ_a /g;
$word =~ s/Сьо/ CZ_sj  CZ_o /g;
$word =~ s/сю/ CZ_sj  CZ_u /g;
$word =~ s/ся/ CZ_sj  CZ_a /g;
$word =~ s/сьо/ CZ_sj  CZ_o /g;
```

```
$word =~ s/ТЮ/ CZ_tj  CZ_u /g;
$word =~ s/ТЯ/ CZ_tj  CZ_a /g;
$word =~ s/ТЬО/ CZ_tj  CZ_o /g;
$word =~ s/Тю/ CZ_tj  CZ_u /g;
$word =~ s/Тя/ CZ_tj  CZ_a /g;
$word =~ s/Тьо/ CZ_tj  CZ_o /g;
$word =~ s/тю/ CZ_tj  CZ_u /g;
$word =~ s/тя/ CZ_tj  CZ_a /g;
$word =~ s/тьо/ CZ_tj  CZ_o /g;
$word =~ s/Ф/ CZ_fj  CZ_u /g;
$word =~ s/ФЯ/ CZ_fj  CZ_a /g;
$word =~ s/ФЬО/ CZ_fj  CZ_o /g;
$word =~ s/Фю/ CZ_fj  CZ_u /g;
$word =~ s/Фя/ CZ_fj  CZ_a /g;
$word =~ s/Фьо/ CZ_fj  CZ_o /g;
$word =~ s/фю/ CZ_fj  CZ_u /g;
$word =~ s/фя/ CZ_fj  CZ_a /g;
$word =~ s/фьо/ CZ_fj  CZ_o /g;
$word =~ s/ЗЮ/ CZ_zj  CZ_u /g;
$word =~ s/ЗЯ/ CZ_zj  CZ_a /g;
$word =~ s/ЗЬО/ CZ_zj  CZ_o /g;
$word =~ s/Зю/ CZ_zj  CZ_u /g;
$word =~ s/Зя/ CZ_zj  CZ_a /g;
$word =~ s/Зьо/ CZ_zj  CZ_o /g;
$word =~ s/зю/ CZ_zj  CZ_u /g;
$word =~ s/зя/ CZ_zj  CZ_a /g;
$word =~ s/зьо/ CZ_zj  CZ_o /g;
$word =~ s/Я/ CZ_ja /g;
$word =~ s/я/ CZ_ja /g;
$word =~ s/Ю/ CZ_ju /g;
$word =~ s/ю/ CZ_ju /g;
$word =~ s/А/ CZ_A /g;
$word =~ s/а/ CZ_a /g;
$word =~ s/Б/ CZ_B /g;
$word =~ s/б/ CZ_b /g;
$word =~ s/В/ CZ_V /g;
$word =~ s/в/ CZ_v /g;
$word =~ s/Г/ CZ_G /g;
$word =~ s/г/ CZ_g /g;
$word =~ s/Д/ CZ_D /g;
$word =~ s/д/ CZ_d /g;
$word =~ s/Е/ CZ_E /g;
$word =~ s/е/ CZ_e /g;
$word =~ s/З/ CZ_Z /g;
$word =~ s/з/ CZ_z /g;
$word =~ s/И/ CZ_I /g;
$word =~ s/и/ CZ_i /g;
$word =~ s/Й/ CZ_J /g;
$word =~ s/й/ CZ_j /g;
$word =~ s/К/ CZ_K /g;
```

```
$word =~ s/к/ CZ_k /g;
$word =~ s/Л/ CZ_L /g;
$word =~ s/л/ CZ_l /g;
$word =~ s/М/ CZ_M /g;
$word =~ s/м/ CZ_m /g;
$word =~ s/Н/ CZ_N /g;
$word =~ s/н/ CZ_n /g;
$word =~ s/О/ CZ_O /g;
$word =~ s/о/ CZ_o /g;
$word =~ s/П/ CZ_P /g;
$word =~ s/п/ CZ_p /g;
$word =~ s/Р/ CZ_R /g;
$word =~ s/р/ CZ_r /g;
$word =~ s/С/ CZ_S /g;
$word =~ s/с/ CZ_s /g;
$word =~ s/Т/ CZ_T /g;
$word =~ s/т/ CZ_t /g;
$word =~ s/У/ CZ_U /g;
$word =~ s/у/ CZ_u /g;
$word =~ s/Ф/ CZ_F /g;
$word =~ s/ф/ CZ_f /g;
$word =~ s/Х/ CZ_X /g;
$word =~ s/х/ CZ_x /g;
$word =~ s/Ъ/ CZ_Y /g;
$word =~ s/ъ/ CZ_y /g;
```

References and Literature

[CC93] **Bernard Comrie and Greville G. Corbett.** *The Slavonic Languages*, London and New York: Routledge, 1993. ISBN 0-415-04755-2.

[CS98] **Kenan Carki (Betreuerin: T. Schultz).** *Entwicklung eines türkischen Spracherkennungssystems für große Vokabulare.* Diplomarbeit, Institut für Logik, Komplexität und Deduktionssysteme, Lehrstuhl Prof. Waibel, Universität Karlsruhe, September 1998.

[FAQ06] **FAQ about Bulgarian.** General Question about Bulgaria. *Q30: How are pronounced Bulgarian letters?* Internet, http://get.info.bg/faq/, February 2006.

[FGH+97] **M. Finke, P. Geutner, H. Hild, T. Kemp, K. Ries and M. Westphal.** *The Karlsruhe-Verbmobil Speech Recognition Engine*. In Proceedings of ICASSP-97. Munich, Germany, 1997.

[Fur86] **S. Furui.** *Speaker-independent isolated word recognition using dynamic features of speech spectrum*. IEEE Trans. ASSP, 1986.

[IPA99] **International Phonetic Association.** *Handbook of the International Phonetic Association*, 1999. ISBN 0-521-63751-1.

[Mad84] **I. Maddieson.** *Patterns of Sounds*. Cambridge: Cambridge University Press. Internet, http://classweb.gmu.edu/accent/nl-ipa/bulgarianipa.html, 1984.

[Omn05] **Omniglot – a guide to written language.** *Bulgarian (Български),* Internet. http://www.omniglot.com/writing/bulgarian.htm, 2005

[Rab89] **Lawrence Rabiner**. *A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition*. Internet, http://www.caip.rutgers.edu/~lrr/Reprints/tutorial%20on%20hmm%20and%20applications.pdf , 1989.

[RS98] **Jürgen Reichert (Betreuerin: T. Schultz).** *Spracherkennung im Chinesischen.* Diplomarbeit, Institut für Logik, Komplexität und Deduktionssysteme, Lehrstuhl Prof. Waibel, Universität Karlsruhe, Dezember 1998.

[Sch00] **Tanja Schultz.** *Dissertation: Language Independent and Language Adaptive Speech Recognition (Multilinguale Spracherkennung - Kombination akustischer Modelle zur Portierung auf neue Sprachen).* 2000.

[Sch02] **Tanja Schultz.** *Globalphone: a multilingual speech and text database developed at Karlsruhe university*. In ICSLP-2002, 345-348. 2002.

[Sch04] **Tanja Schultz.** *Towards Rapid Language Portability of Speech Processing Systems.* In Conference on Speech and Language Systems for Human Communication, Delhi, India. Nov. 2004.

[Sch05] **Tanja Schultz.** *ML-Book.* Not published.

[Sha49] **C. E. Shannon.** *Communication in the presence of noise.* Proc. Institute of Radio Engineers, vol. 37, no.1, pp. 10-21. Jan. 1949.

[SOK$^+$02] **Kiril Simov, Petya Osenova, Sia Kolkovska, Elisaveta Balabanova, Dimitar Doikoff, Krassimira Ivanova, Alexander Simov, Milen Kouylekov.** *Building a Linguistically Interpreted Corpus of Bulgarian: the BulTreeBank*. In: Proceedings of LREC 2002, pages 1729-1736. Canary Islands, Spain, 2002.

[SR94] **Tanja Schultz (Betreuer: I. Rogina).** *Akustische Modellierung sprachlicher und nichtsprachlicher Geräusche.* Studienarbeit, Institut für Logik, Komplexität und Deduktionssysteme, Lehrstuhl Prof. Waibel, Universität Karlsruhe, Juni 1994.

[ST95] **E.G. Schukat-Talamazzini.** *Automatische Spracherkennung*. Verlag Vieweg Braunschweig/Wiesbaden, 1995.

[SW01] **Tanja Schultz und Alex Waibel.** *Language-independent and language adaptive acoustic modeling for speech recognition.* Speech Communication, vol. 35, pp. 31-51, 2001.

[SW97] **Tanja Schultz and Alex Waibel.** *Fast Bootstrapping of LVCSR Systems with Multilingual Phoneme Sets.* Proceedings of the 5th European Conference on Speech Communication and Technology (Eurospeech-1997), Vol. 1 pp 371--373, Rhodes, Greece, September 1997.

[SW98] **Tanja Schultz and Alex Waibel.** *Das Projekt GlobalPhone: Multilinguale Spracherkennung.* Computers, Linguistics, and Phonetics between Language and Speech. Proceedings of the 4th Conference on NLP (Konvens-1998), pp 179-189, Bonn, Germany, October 1998.

[SW98_EN] **Tanja Schultz and Alex Waibel.** *Development of Multilingual Acoustic Models in the GlobalPhone Project*. Proceedings of the 1st Workshop on Text, Speech, and Dialogue (TSD-1998), pp 311-316, Brno, Czech Republic, September 1998.

[SWFeb98] **Tanja Schultz and Alex Waibel.** *Multilingual and Crosslingual Speech Recognition.* Proceedings of the DARPA Broadcast News Transcription and Understanding, pp 259-262, Lansdowne Virginia, February 1998.

[WA-W$^+$94] **M. Woszczyna, N. Aoki-Waibel, F.D. Buo, N. Coccaro, K. Horiguchi, T. Kemp, A. Lavie, A. McNair, T. Polzin, I. Rogina, C.P. Rose, T. Schultz, B. Suhm, M. Tomita, A. Waibel.** *JANUS 93: Towards Spontaneous Speech Translation*. Proceedings of the ICASSP, 1994.

[WGM-TSW] **Alex Waibel, Petra Geutner, Laura Mayfield-Tomokiyo, Tanja Schultz, and Monika Woszczyna.** *Multilinguality in Speech and Spoken Language Systems.* Proceedings of the IEEE, Special Issue on Spoken Language Processing, Volume 88(8), pp 1297-1313, August 2000.

[Wiki06] **English language version edition of Wikipedia.** *Bulgarian Language.* Internet, http://en.wikipedia.org/wiki/Bulgarian_language/, February 2006.

[WL90] **Alex Waibel, Kai-Fu Lee.** *Readings in Speech Recognitions*. Morgan Kaufmann Publishers, Inc. San Mateo, California, 1990

[WLL97] **Alex Waibel, Alon Lavie, Lori Levin.** *JANUS: A System for Translation of Conversational Speech.* KI - Kunstliche Intelligenz, 1997.

[WSSSM00] **Alex Waibel, Hagen Soltau, Tanja Schultz, Thomas Schaaf, and Florian Metze.** *Multilingual Speech Recognition. Verbmobil: Foundations of Speech-to-Speech Translation.* Wolfgang Wahlster (Ed.), Springer Verlag, 2000.

[WSW98] **Martin Westphal, Tanja Schultz, Alex Waibel.** *Linear Discriminant – A New Criterion for Speaker Normalization.* In ICSLP 1998, Sydney

[YW94] **S. J. Young and P. C. Woodland.** *State clustering in hidden markov model-based continuous speech recognition.* Computer Speech and Language, vol. 8, no. 4, pp. 369-384, 1994.