Institutul de Cercetări pentru Inteligență Artificială
“Mihai Drăgănescu”

Contribuții la modelarea și implementarea sistemelor de sinteză a vorbirii. Studiu de caz: limba română

●

Romanian Academy, Research Institute for Artificial Intelligence “Mihai Drăgănescu”

Contributions to the modeling and implementation of Text–To–Speech synthesis systems

Case study on the Romanian Language

Tiberiu BOROȘ

Coordinator: Academician **Dan TUFIȘ**

2013

Acknowledgments

First and foremost I would like to thank my advisor academician Dan Tufiș for his valuable advice and contributions during the preparation of this thesis. It has been an honor to work along side him and my coleagues, Radu Ion, Dan Ștefănescu, Ștefan Daniel Dumitrescu, Verginica Barbu Mititelu, Elena Irimia, and Alexandru Ceaușu – their advice was productive and their enthusiasm contagious.

The work described in this thesis would not have been possible without the initiative of Dr. Adriana Stan to make the Romanian Speech Synthesis (RSS) database publically available for research purposes. Also, the collective initiative of the METANET project, under the supervision of academician Dan Tufiș and professor Dan Cristea, to make natural language processing resources and tools available for the Romanian language have undoubtedly helped the development of Romanian Spoken Language Processing field.

The research carried out on statistical parametric speech synthesis by the Speech Processing Group from the Technical University of Cluj, under the supervision of Professor Mircea Giurgiu, is truly an asset to Romanian speech synthesis and the SIMPLE4ALL project represents a scientific and technological leap–forward in TTS synthesis.

Furthermore, I would like to thank Dr. Vasile Apopei, Dr. Adriana Stan and Anca Munteanu for their help in the development and evaluation of the RSS–TOBI corpus.

Special thanks to Ștefan Daniel Dumitrescu and Radu Ion for their contributions in the preparation for the Blizzard Challenge Campaign and Dan Ștefănescu for his help in the development of the Romanian data–driven phonetic transcription system.

Additionally, I would like to thank all the volunteers that participated in the online evaluation campaign form the Romanian Spoken Language Processing portal.

Contents

1. **Introduction 1**
1.1 The TTS task 1
1.2 Applications of TTS synthesis 3
1.3 Original contributions to Romanian text–to–speech synthesis 3
1.4 Thesis structure 4
2. **Fundamentals 6**
2.1 The production of human speech 6
2.2 Sounds of the Romanian language 7
2.3 Digital signal processing 11
2.3.1 Sampling 11
2.3.2 The Nyquist–Shannon sampling theorem 12
2.3.3 Speech signal representations 12
2.4 Basic Machine Learning 16
3. **Basic natural language processing for text–to–speech synthesis 20**
3.1 General description of text–processing front–ends 20
3.2 Case study of DFKI MARY, Festival, Flite and FreeTTS 21
3.3 Part–of–speech tagging 24
3.3.1 Introduction 24
3.3.2 Large tagset part–of–speech tagging with feed–forward neural networks 26
3.3.3 The MSD binary encoding scheme 28
3.3.4 The training and tagging procedure 28
3.3.5 Network hyper parameters 30
3.3.6 Network pattern analysis 34
3.3.7 Genetic optimization for the Neural POS tagger 36

3.3.8 Experiment results 39

3.3.9 Conclusions 40

3.4 Diacritic restoration and word casing 41

3.4.1 Evaluation 43

3.4.2 Conclusions 49

3.5 Spellchecking 50

3.6 Text normalization 51

3.7 A unified lexical processing framework based on the Margin Infused Relaxed Algorithm 52

3.7.1 Syllabification 53

3.7.2 Lemmatization 55

3.7.3 Phonetic transcription 59

3.7.4 Lexical stress prediction 62

3.8 Foreign words in TTS synthesis 64

3.8.1 Detecting which words require transliteration in TTS 65

3.9 Conclusions 66

4. Digital Signal Processing for Text–to–speech synthesis 68

4.1 An introduction to speech synthesis back–ends 68

4.2 Corpora based speech synthesis methods 69

4.2.1 Unit–selection speech synthesis 70

4.2.2 Statistical parametric speech synthesis 72

4.2.3 Hybrid approaches 73

4.3 The unit–selection speech synthesis baseline system 73

4.4 Conclusions 74

5. Speech and prosody 76

5.1 What is prosody 76

5.2 Prosody annotation systems 79

5.3 Building a prosody annotated Romanian corpus 80

5.4 Evaluation of the Romanian prosody annotated corpus 82

5.5 Conclusions 84

6. Evaluation of the RACAI text–to–speech synthesis system 85

6.1 Evaluation methodologies for text–to–speech synthesis systems 85

6.2 The Blizzard Challenge 86

6.2.1 Preparation phase 87

6.2.2 Results 90

6.3 Conclusions 92

7. Applications of text–to–speech synthesis 93

7.1 An overview of TTS applications 93

7.2 Building a speech–to–speech translation prototype 95

7.2.1 An overview of the adjacent technologies 96

7.2.2 Issues in the design and implementation of a speech translation system 97

7.3 Conclusions 99

8. Conclusions 100

8.1 Personal contributions 101

9. Appendixes 106

10.1 List of abbreviations 106

10.2 List of figures 108

10.3 List of tables 109

10. References 110

Chapter 1

In this chapter, we introduce the concept of text–to–speech (TTS) synthesis, we analyze its benefits and we offer an overview of the structure of this work.

Introduction

1.1 The TTS task

The recent advances in technology have led to a large spread of software and devices that are focused on delivering high level accessibility to end–users. The technologies involved in this process fall into the field of Human Computer Interaction (HCI) and the current trend in research and commercial application development is toward building dialog enabled interfaces that allow fast and reliable interaction between humans and computers (see chapter 7 for a detailed description). Text–To–Speech (TTS) synthesis falls into the larger domain of Spoken Language Processing.

Spoken Language Processing (SLP) is a research domain oriented toward processes, methods and technologies designed to enable communication between users and computers through spoken language. It contains three primary research domains: (1) automatic speech recognition (ASR), (2) text–to–speech synthesis (TTS) and (3) specific areas of the broader field of Natural Language Processing (NLP), providing the infrastructure and essential support for the TTS process.

The main difference between TTS and other types of speech synthesis (e.g. domain specific speech synthesis) is that TTS has to be able to synthesize voice from an ***arbitrary*** or ***unrestricted*** text. The conversion from speech to text is a process which involves information loss. On the other side, converting back from text to spoken language requires the recovery of this information, which in some cases is a near to impossible-to-achieve desideratum, thus rendering the TTS task extremely difficult (we will further address this issue in chapter 5, when we will refer to speech and prosody). Of course, the level of quality in terms of speed, intelligibility and naturalness of a state–of–the–art limited–domain speech synthesis system is unmatched by TTS: the restricted system works with a limited number of possible input and output combinations and, together with meticulous data preparation will produce almost flawless results. In comparison, TTS has to work with unpredictable inputs that require carefully crafted and computationally expensive NLP and Digital Signal Processing (DSP) steps (see figure 1).

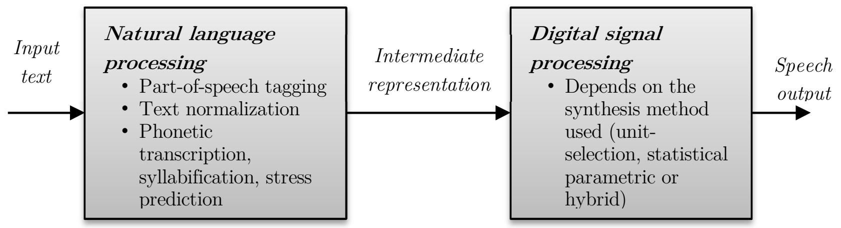

Input text
Natural language processing
• Part-of-speech tagging
• Text normalization
• Phonetic transcription, syllabification, stress prediction
Intermediate representation
Digital signal processing
• Depends on the synthesis method used (unit-selection, statistical parametric or hybrid)
Speech output

Figure 1 – Basic TTS architecture

The large interest shown toward mimicking human voice, starting with the first generation of rule–based speech synthesis systems (Allen et al., 1987), followed by corpora based methods such as concatenative unit selection or statistical parametric speech synthesis (Tokuda et al., 2000) combined with an abundant number of studies addressing the issue of out–of–vocabulary (OOV) words plaguing specific sub–tasks of the process, shows just how complex the TTS task is. Regardless of the approach, significant effort has been invested in trying to improve the naturalness of the synthetized voice and to increase the level of acceptance of TTS systems amongst its users. However, as pointed out by the Lessac research group, "the expressive human speech is prosodic – it is musical, with melodies and rhythms that clarify words and phrases, as well as create the auditory context to enhance a listener's comprehension. The prosodic musical overlay communicates emotion and lowers the cognitive load. Natural sounding speech, whether spoken or synthesized, enables the human mind to handle listening and comprehension as a background task, not requiring full concentration."[1]. Text does not share the same ability to encode information as speech itself, and even skilled writers find it hard to correctly embed the message they want to share without requiring an intensive effort from the reader, sometimes failing in this attempt.

[1] http://lessactech.com/

1.2 Applications of TTS synthesis

To outline the benefits of TTS synthesis we will start by pointing out that speech is the preferred and common way of interaction between humans and it is expected that the same type of interaction would be beneficial in generic HCI systems. TTS also contributes in more specific areas, such as in closing the gap between elderly and/or disabled people and the rest of the society in terms of autonomy.

Examples of applications that involve TTS synthesis are:

- Accessibility for the visually impaired;
- Narrators: e–book readers, email readers etc.;
- Speech translation;
- General HCI (e.g. navigation systems);
- E–learning systems.

These general examples are provided to demonstrate the importance of TTS synthesis; a fine–grained description will be later presented in section 7.

1.3 Original contributions to Romanian text–to–speech synthesis

One of the main impairments of Romanian TTS synthesis was the lack of (freely available) resources. It is only in recent years that the Romanian Speech Synthesis (RSS) database (Stan et al., 2011) has been freely released for research purposes. This has undoubtedly boosted research in this field. One major contribution to Romanian TTS synthesis was to build and evaluate a prosody annotated corpus, based on a section of the RSS database (see chapter 5). The corpus is composed of a mixture of data obtained from all the NLP methods that will be later presented (chapter 3), with an additional manually–created prosody layer. This layer uses the Tone and Break Indices (ToBI) (Silverman et al., 1992) standard for annotation with the adjustments introduced by Jitcă et al. (2012) to suite the Romanian prosody phenomena. The corpus is available through the META–SHARE platform[2].

[2] http://ws.racai.ro:9191

Another contribution was to create a NLP Framework specifically designed for Romanian TTS synthesis (see chapters 3 for details and 6 for evaluation). Most of the tools and methods included in this framework are statistically–based because (1) they can automatically extract rules from existing corpora, (2) they offer language independence and (3) they do not require extensive linguistic knowledge in order to provide highly accurate solutions to typical TTS–related NLP problems. The work focused on adapting state–of–the–art methods to Romanian and developing and testing original solutions such as (1) large tagset labeling with Neural Networks for part–of–speech tagging, (2) lexical stress prediction using the Margin–Infused Relaxed Algorithm in a sequence labeling setting and (3) detection and transliteration of foreign words for TTS.

The final section of this thesis covers a typical application of Spoken Language processing, namely Speech to Speech Translation, which involves ASR, Machine Translation (MT) and TTS. To the best of the author's knowledge, this type of application has never been applied to Romanian. Besides the external ASR solution (Google ASR), the rest of the system is built using locally developed resources and tools such as RACAI MT System, RACAI TTS system (which is the subject of the current work) and RACAI Spellchecker.

RACAI Spellchecker is also an original contribution consisting of set of tools and methods designed to provide spellchecking, word–casing and diacritic restoration. Its main purpose is to provide necessary text–processing between the ASR and MT components in Speech to Speech synthesis, as well as offering generic functionality as a text–proofing tool.

1.4 Thesis structure

Chapter 2 contains general theoretical aspects regarding the production of human speech, analogue and digital signals, models for signal representation and methods for signal analysis, in order to create the foundation and context for the rest of the presentation.

Chapter 3 offers insight into the NLP processing steps involved in TTS synthesis, providing a review of the already available NLP tools and frameworks and continues with original contributions brought to the field of Romanian TTS synthesis.

Chapter 4 expands on the processing involved in the actual generation of speech from text and introduces general knowledge about speech analysis for TTS and corpora based speech synthesis methods.

Chapter 5 discusses the complex process of prosody prediction in TTS synthesis, starting with typical prosody annotation systems and continuing with the development and evaluation of the RSS–ToBI corpus.

Chapter 6 contains a thorough real–world evaluation of the work presented in this thesis and describes the steps involved in adapting this system to English during the 2013 Blizzard Challenge TTS evaluation campaign.

Finally, chapter 7 introduces an application of TTS synthesis, namely Speech translation and describes the work involved in creating a Romanian–English bi–directional speech translation system implemented on a networked mobile Android–based platform.

Before continuing, it must be mentioned that the theoretical information in chapter 2 is only designed to provide a simple context of the TTS task and the rest of the theoretical notions, theories, methods, models and techniques will be presented when needed in each chapter or sub–chapter.

Chapter 2

This chapter is focused on the theoretical aspects of human speech production and digital signal processing for speech synthesis

Fundamentals

2.1 The production of human speech

Human speech is a complex physical and psychological process. Knowing how this process happens at the physical level is important for understanding the parameter extraction process used both in speech synthesis and speech recognition. The human speech production apparatus is composed of (1) the lungs, (2) the vocal folds and (3) the articulators.

The lungs represent a pump responsible for producing the air flow for vibrating the vocal folds. *The vocal folds* vibrate and generate audible pulses which control the *pitch* and *tone* of the voice. The modulated airflow from the vocal folds is than *filtered* by the articulators, which are the tongue, palate, cheeks, nose and lips. The articulation process produces classes of sounds which are primarily divided into vowels and consonants.

Sounds are best characterized using the frequency domain (see section 2.2 for details) and as a consequence of the speech apparatus physiognomy we can observe a number of properties regarding the human voice. First of all, the pulses generated by the vocal folds determine a so called *fundamental frequency* or *pitch* of the voice and a number of harmonics that are dependent on this fundamental frequency. The exact pitch value varies with speakers but it is usually inside the 100–250 Hz interval for male speakers and higher for female speakers. An analysis of the speech spectrum reveals that most of the energy during normal human speech is located between 0 and 4 kHz; above 10 kHz the energy level is very low. Thus, by using a sampling rate of 8 kHz in the acquisition process, the human voice is still intelligible, a fact which is commonly exploited in communications.

Different sounds, especially voiced sounds (see section 2.3) have specific regions inside the spectrum where the energy is higher. These regions are commonly referred to as formants (Fant, 1970) and they are used to (1) identify a phoneme or (2) extract information about the speaker.

2.2 Sounds of the Romanian language

This section covers theoretical aspects about the sounds of the Romanian language, their classification and their articulatory features and it is motivated by two of the main processes involved in TTS synthesis: (1) the conversion of text into phonemes (see section 3.7 for details) and (2) the feature extraction process for corpora based speech synthesis systems (see sections 3.1, 4.1, 4.2 for details):

(1) One important task of TTS is the conversion of the sequence of input words into a sequence of sounds. This is a complex process which involves a lot of text pre–processing (see chapter 3 for challenges of natural language processing for text–to–speech synthesis) inside which one of the core components is phonetic transcription. Finding the relationship between letters of a word (usually referred to as graphemes) and their phonetic equivalents is a complex task for all languages (especially English) and although Romanian has a preponderantly phonetic orthography, the task of phonetic transcription is not straight–forward (see section 3.7 for details);

(2) Corpora based speech synthesis methods such as unit selection or statistical parametric speech synthesis (see chapter 4, section 4.2) use the articulatory features extracted from a window of three to five phonemes (centered on the current phoneme) as context information in the synthesis process. This is mandatory for building a prosodic context and for being able to group similar sounds in the process of parameter extraction and generation.

Before proceeding with the description of the Romanian sound inventory some general information about phonetic transcription and computer annotation standards must be provided:

(1) The **International Phonetic Alphabet (IPA)** is an alphabetic system used for phonetic annotation, intended for providing a written representation for the oral language (IPA, 1999). It is composed of an inventory of 107 phonetic symbols and 52 diacritical marks.

(2) **Speech Assessment Methods Phonetic Alphabet (SAMPA)** represents a computer readable phonetic alphabet which is used to

encode the IPA in American Standard Code for Information Interchange (ASCII).

The following presentation is compiled from multiple sources (Vasiliu, 1956; Burileanu, 1999; Stan, 2011). The phonetic notation conventions are not SAMPA. They are identical to those proposed in (Stan, 2011) (see Table 1 for correspondence between the Romanian phonetic inventory, the SAMPA standard and RSS conventions) because most work was carried around the RSS corpus (Stan et al., 2011) in which these conventions are used.

Table 1 – Correspondence between the RSS notation standard and SAMPA – adapted from Stan (2011)

Phoneme	Sample word	SAMPA	Phoneme	Sample word	SAMPA
a	m***a***ria	a	**zh**	a***j***utor	Z
@	cas***ă***	@	**l**	a***l***ta	l
a@	m***â***nă/***î***nceput	1	**m**	amară	m
b	a***b***ac	b	**n**	în***n***să	n
k	act	k	**o**	m***o***tor	o
ch	a***ch***eea	tS	**o@**	***o***aie	o_X
d	***d***acă	d	**p**	a***p***ă	p
e	d***e***spre	e	**r**	a***r***tă	r
e@	c***e***as	e_X	**s**	a***s***ta	s
f	***f***apt	f	**sh**	***ș***i	S
g	a***g***onie	g	**t**	ta***t***a	t
dz	***g***eam	dZ	**ts**	***ț***ară	ts
h	***h***artă	h	**u**	m***u***nca	u
i	in***i***mă	i	**w**	plo***u***at	w
j	***i***epure	j	**v**	***v***ara	v
ij	câin***i***	i_0	**z**	a***z***i	z

In Romanian the phonetic inventory is divided into two main categories:

(1) **Consonants**, which represent a distinctive class of sounds which are generated by obstacles inside vocal filter

(2) **Vowels**.

This coarse classification is further refined based on articulatory properties:

(1) **Vowels (see table 2)**:
  a. **Based on the opening angle of the jaw** (apperture):
    i. *Open vowels*: **a**
    ii. *Mid vowels*: ***e, e@***, ***@***, ***o, o@***
    iii. *Closed vowels*: ***i, j***, *ij*, ***a@, u, w***
  b. **Based on the articulation point**:
    i. *Rear vowels:* ***e, e@***, ***i, ij, j***
    ii. *Central vowels*: ***a***, ***@***, ***a@***
    iii. *Forward vowels*: ***o, o@***, ***u, w***
  c. **Based on the contour of the lips**:
    i. *Rounded vowels:* ***o, o@, u, w***
    ii. *Unrounded vowels:* ***a, e, e@, i, j, ij, @, a@***

Table 2 – **Vowel classification based on articulatory features**

Phoneme	**Opening**	**Articulation point**	**Lips**
@	Mid	Central	Unrounded
a	Open	Central	Unrounded
a@	Closed	Central	Unrounded
e	Mid	Front	Unrounded
e@	Mid	Front	Unrounded
i	Closed	Front	Unrounded
ij	Closed	Front	Unrounded
j	Closed	Front	Unrounded
o	Mid	Back	Rounded
o@	Mid	Back	Rounded
u	Closed	Back	Rounded
w	Closed	Back	Rounded

(2) **Consonants (table 3)**: Consonants are divided into two main groups: voiced and unvoiced
  a. Unvoiced consonants:
    i. Articulation type:
      1. *Occlusives*: ***p***, ***t***, ***k*** – they are formed by a complete closing and sudden openning of the vocal tract

2. *Fricatives*: ***f, s, sh, h*** – they are formed by a narrowing of the vocal tract
3. *Affricates*: ***tz, ch, k(e/i)*** – they start as occlusive consonants and end like fricative consonants.

ii. Based on the possition of the obstacle inside the vocal filter:
1. *Bilabial*: ***p***
2. *Labio–dental*: ***f***
3. *Dental:* ***t, s, tz***
4. *Pre–palatal*: ***sh***, ***ch***
5. *Palatal*: ***k(e/i)***
6. *Velar*: ***k***
7. *Glottal:* ***h***

b. Voiced consonants: this class of consonants shares some similarities with vowels such as the fact that their articulation process is accompanied by the vibration of the vocal folds. However, like the rest of the consonants they are always unstressed inside a sillable.

i. Articulation type:
1. *Oclussive*: ***m***, ***n, b***, ***d, g***
2. *Affricates: g(e/i), gh*
3. *Fricatives: v, zh, z,*
4. *Liquid*: ***l***
5. *Vibrant*: ***r***

ii. Based on the possition of the obstacle inside the vocal filter:
1. *Bilabial*: ***m, b***
2. *Labio*-dental: ***v***
3. *Dental*: ***d, n***, ***r, z***
4. *Lateral*: ***l***
5. *Pre-palatal:* ***zh, gh***
6. *Palatal:* ***g(e/i)***
7. *Velar:* ***g***

iii. Based on nasality – the only nasal sounds in Romanian are: ***m***, ***n***

Table 3 – Classification of consonants based on articulatory features

Articulation type	Voicing	Occlusive	Affricate	Fricative	Occlusive nasal	Liquid	
						Lateral	Vibrant
Bilabial	Unvoiced	p					
	Voiced	b			m		
Labio–dental	Unvoiced			f			
	Voiced			v			
Dental	Unvoiced	t	tz	s			
	Voiced	d		z	n	l	r
Pre–palatal	Unvoiced		k*e*/k*i*	sh			
	Voiced		g*e*/g*i*	zh			
Palatal	Unvoiced		ch				
	Voiced		gh				
Velar	Unvoiced	k					
	Voiced	g					
Glottal	Unvoiced			h			
	Voiced						

2.3 Digital signal processing

Digital signal processing is a technological area supported by a rigorous mathematical apparatus designed for (1) obtaining, (2) manipulating and (3) extracting information from a signal which is represented as discrete values.

The speech signal originating from the human speech production system is an *analog* (continuous) signal that cannot be processed directly. The process of obtaining a discrete representation of a continuous signal is called *sampling.*

2.3.1 Sampling

For convenience, a continuous signal x can be represented as a continuous function $x_a(t)$. For converting this signal into a digital (discrete) form we are required to use a *sampling period* T, which is used to define our discrete signal $x[n]$ as $x[n] = x_a(nT)$ (see figure 2). Sampling can be actually regarded as a pulling mechanism which periodically (depending on the sampling period) checks the value of the analog signal x. How well does $x[n]$ approximate the original signal x depends on the value of the sampling period: the larger T is, the coarser the approximation becomes; the smaller T is, the more $x[n]$ resembles x.

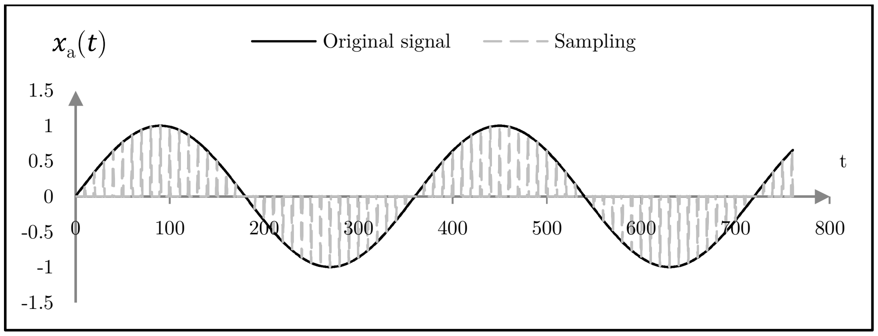


Figure 2 – Example of a sampled signal

It is easier to describe a digital signal using a function of T, called *sampling frequency* and defined as $F_s = {}^{1}/_{T}$, mainly because of the implications of the *Nyquist–Shannon sampling theorem* (Nyquist, 1928;Shannon, 1949).

2.3.2 The Nyquist–Shannon sampling theorem

The Nyquist–Shannon sampling theorem, commonly referred to as the sampling theorem, is a result of information theory, which states that if a signal, represented by a continuous time function $x_a(t)$, contains no frequencies higher than F, than the signal is completely determined by a series of points equally spaced at ${}^{1}/_{2F}$ apart. This means that from a sampled signal $x[n]$ we can fully reconstruct the original signal $x_a(t)$ only if it is band–limited to half of the sampling rate.

2.3.3 Speech signal representations

The most straight–forward way to represent a signal is through the use of a continuous time–series $x_a(t)$ or a discrete time series $x[n]$, a method which is referred to as *time–domain representation*. However, this type of representation is not always suitable for processing signals and extracting parameters since most of these parameters are frequency related.

One basic type of speech representation, inspired by the production of human speech (presented in section 2.1), is the source–filter model, in which speech is decomposed into a source (vocal air at the vocal cords) that passes through a linear time varying filter (resonances of the vocal tract).

The dynamics of the speech signal give it a property that is routinely used by ASR and TTS systems of being quasi–stationary (almost stationary) for short periods of time (from 20 to 50 milliseconds). By exploiting this property and performing a segmentation of the time–domain representation into a series of analysis frames, the speech signal can be converted into the frequency domain representation through the use of a mathematical transform, called the *Fourier Transform.*

2.3.3.1 Fourier Transform

The Fourier Transform is a mathematical transform, named after the mathematician and physicist Joseph Fourier, which, applied to a mathematical function of time $f(t)$, transforms it into another function, usually denoted as $F(\varepsilon)$ (see equation 1), where ε represents cycles per second (*hertz*) or radians per second.

$$F(\varepsilon) = \int_{-\infty}^{+\infty} f(t) e^{-2\pi i t \varepsilon} dt \tag{1}$$

The original function $f(t)$ can be recovered using the *Inverse Fourier Transform* (equation 2):

$$f(t) = \int_{-\infty}^{+\infty} F(\varepsilon) e^{2\pi i x \varepsilon} d\varepsilon \tag{2}$$

The Fourier transform originates from the study of the *Fourier series*, in which complex mathematical functions are rewritten as *sums of waves* expressed as *sines* and *cosines*.

A variation of the Fourier Transform that is used in computational applications is the *Discrete Fourier Transform* (DFT). In mathematics, the discrete Fourier transform (DFT) converts a finite list of equally spaced samples of a function into the list of coefficients of a finite combination of complex sinusoids, ordered by their frequencies, that has the same sample values. It can be said to convert the sampled function from its original domain (often time or position along a line) to the frequency domain.

The input samples as well as the output coefficients are complex numbers. The frequencies of the output sinusoids are integer multiples of a fundamental frequency, whose corresponding period is the length of the sampling interval. The combination of sinusoids obtained through the DFT is therefore periodic with that same period. The DFT differs from the discrete–time Fourier transform (DTFT) in that its input and output sequences are both finite; it is therefore said to be the Fourier analysis of finite–domain (or periodic) discrete–time functions.

The *Fast Fourier Transform* algorithm (Cooley and Tukey, 1965) is a highly efficient algorithm for solving the DFT and its inverse, which takes as input a sequence of N complex numbers and transforms it into another sequence of complex numbers. It is used to either decompose the input sequence of numbers into components of different frequencies (see equation 3) or vice–versa (see equation 4).

$$X_k = \sum_{n=0}^{N-1} x[n] e^{-\frac{2\pi i}{N}kn}, k = 0,1 \dots N-1, \tag{3}$$

$$x[n] = \frac{1}{N}\sum_{k=0}^{N-1} X_k e^{\frac{2\pi i}{N}kn}, n = 0,1 \dots N-1. \tag{4}$$

If one would naively perform the DFT using its definition, the complexity of the algorithm would be $O(n^2)$ mathematical operations. However, the Cooley–Tukey FFT algorithm takes advantage of a symmetrical mathematical property of DFT and reduces the complexity of the calculus to $O(n \log n)$. This particular FFT algorithm falls into the divide and conquer class, requiring that the input data size is a power of 2 and because of the previously mentioned cvasi–stationary property of the speech signal it is convenient to perform FFT analysis by dividing the input data into sequences of 2^k elements that fit between 20 and 50 ms. This process is performed by applying a *window function* over the input data and, in some cases, by adding zeros to the input data (0–padding).

The choice of the window function is highly important mainly because of *spectral leakage*. Spectral leakage is an unwanted effect in which the Fourier components of an analyzed signal are spread around the spectrum. The most common windowing function is the rectangular window (see equation 5)

$$w(n) = \begin{cases} 1, for\ n\ belonging\ to\ a\ given\ interval \\ 0,\ \ otherwhise \end{cases} \tag{5}$$

The wide–spread of the leakage makes the rectangular window not a good choice in speech signal processing. Instead, depending on the application, some popular choices for window functions are: the Blackman window (equation 6), the Hann window (equation 7), the Hamming window (equation 8) or the Tukey window (equation 9) (Blackman and Tukey, 1959).

$$w(n) = 0.35875 - 0.48829\cos\left(\tfrac{2\pi n}{N-1}\right) + 0.14128\cos\left(\tfrac{4\pi n}{N-1}\right) - 0.01168\cos\left(\tfrac{6\pi n}{N-1}\right) \tag{6}$$

$$w(n) = 0.5\left(1 - cos\frac{2\pi n}{N-1}\right) \tag{7}$$

$$w(n) = 0.54 - 0.46\left(\cos\frac{2\pi n}{N-1}\right) \tag{8}$$

$$w(n) = \begin{cases} \frac{1}{2}\left(1 + \cos\pi\frac{|n| - \alpha\frac{n}{2}}{(1-\alpha)\frac{N}{2}}\right), if\,\frac{\alpha N}{2} \le |n| \le \frac{N}{2} \\ 1, if\ 0 \le |n| \le \frac{\alpha N}{2} \end{cases} \tag{9}$$

2.3.3.2 *Low–dimensional representations*

By computing the input and output data size involved in the Fourier analysis of the signal, one can see that it ranges from 256 to 1024 elements, depending on the sampling rate. This high dimensional representation of the speech signal generates a number of problems that are usually related to data–sparseness in statistical methods and poses a series of challenges for both ASR and statistical parametric TTS synthesis. For this reason, a lot of research has been focused in finding a lower dimensional representation that would preserve the properties of the original one and would be suitable for statistical processing.

One of the popular methods for obtaining low–dimensional spectral representations of speech is the Mel–frequency cepstral (MFC) analysis. It is based on the source–filter theory, which suggests that speech can be modeled as a two stage process: (1) in the first stage, speech is generated with its own spectral structure (usually by the glottal source) and (2) in the second stage, the signal is filtered by the vocal tract. In the Mel–frequency cepstral analysis, the *transfer function* of the vocal tract is modeled using Mel–frequency cepstral coefficients (MFCCs). In the MFC analysis, the frequency bands are placed on the Mel–scale so that they approximate the human auditory system response better than linearly spaced frequency bands. In order to calculate the MFCCs from a windowed signal the following steps are required:

(1) Applying the DFT on the windowed signal
(2) Filter the Fourier Spectrum through the Mel–scale filter bank
(3) Calculate the Log power spectrum
(4) Apply the Discrete Cosine Transform[3] over the resulting powers as they were a signal

The MFCC representation is only one of the possible low–dimensional spectral representations of speech. Other examples are: the linear predictive coding (LPC) (Tremain, 1982), the perceptual linear predictive analysis (PLP) (Hermansky, 1990), the Relative Spectral Transform Perceptual Linear Prediction (RASTA–PLP) (Hermansky et al., 1992) etc.

2.4 Basic Machine Learning

Machine Learning (ML) is a sub–domain of *Artificial Intelligence* (AI) that refers to the study of methods and techniques and the implementation of algorithms that are able to *learn* from available data. One of the important properties of ML algorithms is their *ability to generalize*, which means that they work well on *previously unseen data*.

The relationship between text (usually regarded as a sequence of letters or graphemes) and the parameters used to characterize the human voice (spectral and prosodic) is not straightforward. The number of variations in which a single utterance can be spoken is virtually unlimited and currently the best solution for TTS systems is to find a general and acceptable solution, in which any combination of input graphemes (previously seen or unseen) is converted into speech based on observing patterns in previously seen data. While this is achieved through the use of ML algorithms or classifiers, it is not a simple task as the performance of these algorithms depends on a number of factors:

Note: in the following description, the term *model size/structure* refers to the internal representation used by each classifier to store model parameters (e.g. number of hidden layer and their size for neural networks, number of

[3] The *discrete cosine transform* (DCT) expresses a finite sequence of data points in terms of a sum of cosine functions oscillating at different frequencies.

clusters, polynomial order for regressions etc.), not to the size of the training dataset, which is the number of examples provided during training.

(1) it is critically important to choose the correct ML algorithm for a given task: some classifiers are intended to provide a multinomial output (a discrete set of labels), while others are designed for real–valued output; there are online machine learning algorithms, meaning that they are trained with one example at a time thus making them adaptable to changing inputs and there are offline machine learning algorithms that train initially on a static dataset and do not change in time; there are standard algorithms for which the classification result of an instance does not affect the following instance to be classified and there are sequence labeling algorithms for which the classification result is dependent on the previous results,

(2) the performance of any classifier is dependent on the (1) feature–set that is used as input, (2) the size/structure of the model it is supposed to build for representation and (3) the quality and quantity of the training data. Complex features that rarely appear in the training corpus combined with *over–sized models* can easily lead to *over–fitting the data* (working very well on previously seen data but not being able to generalize for unseen data), as opposed to using simple features and small models which generalize well on unseen data but often they *under–fit* the training examples (the classifier works the same on seen and unseen data, but the accuracy figures are low). The training data has to be carefully chosen to resemble test data as much as possible and to provide sufficient examples, since it is impossible to generalize if there is no connection between the training and the actual data.

There are two main classes of ML algorithms: (1) *supervised* – which require that the training data is composed of examples containing pairs of input values associated with output values and (2) *unsupervised* – which use examples composed of input values (no associated output values) which are then grouped together by similarity.

The operation performed by most *unsupervised ML* algorithms is known as clustering and it refers to grouping together or partitioning similar input examples which are close to one another based on a predefined metric. The most common unsupervised ML algorithm is K–means clustering, which is designed to separate a set of n observations $(x_{1,n})$ into a set of k clusters (see equation 10), where each observation is placed in a cluster with the nearest *mean*.

$$\underset{S_1 \dots S_k}{\operatorname{argmin}} \sum_{i=1}^{k} \sum_{x_j \in S_i} \left\| x_j - \mu_i \right\| \tag{10}$$

where S_i is a partition with the mean μ_i, x_j is an observation and $\left\| x_j - \mu_i \right\|$ is a distance between two n–dimensional vectors (usually Euclidian distance).

A well–known and frequently used algorithm for automatically optimizing clustering decisions is Expectation Maximization (EM). EM is a monotonically converging iterative algorithm which is used for finding the maximum likelihood estimates of parameters in statistical methods. It alternates between an Expectation step (in which it calculates a function for the expectation of the log–likelihood based on its current parameters) and a Maximization step (in which in re–computes the parameters in order maximize the previously constructed expectation function), hence the name Expectation–Maximization. Besides data–clustering, variations of the EM algorithm are often used in NLP, TTS and ASR to automatically induce alignments between input sequences and output states.

In *supervised ML* the training data is composed of a pair of input values and desired output values. Both the input and output can use real–valued vectors or a discrete set of labels.

There is a large variety of supervised ML algorithms such as: linear, polynomial and logistic regression, Support Vector Machine (SVM) (Cortes and Vapnik, 1995), Maximum Entropy classifiers (MaxEnt) (Berger et al., 1996), Decision Trees Learning (Iterative Dichotomiser 3 (ID3), C4.5, Classification and Regression Trees (CART) etc.), Random Forests, Artificial Neural Networks and many others. This presentation will only focus on two algorithms, namely Artificial Neural Networks (ANNs) and the Margin Infused Relaxed Algorithm (MIRA) (Crammer and Singer, 2003) because they will be later used in the following sections for performing various TTS related tasks.

Artificial Neural Networks are biologically inspired computational models composed of a network of interconnected computationally simulated artificial neurons. The links between neurons are called synapses and they are usually numerically represented as a real–valued weight. Based on connection graph between neurons, ANNs can be (1) feed–forward when neurons are grouped in layers and connections are formed only between neurons belonging to consecutive layers or (2) recurrent, which allow directed cycles in their structure. The simplest feed–forward network architecture is usually composed of an input layer, a hidden layer and an output layer and it is easily trained using the Backpropagation algorithm (Rumelhart et al., 1986). Depending on the task, several ANN architectures such as Long–Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997), Restricted Boltzmann Machine (RBM) (Ackley et al., 1985; Hinton and Sejnowski, 1986; Chen and Murray, 2003; Tieleman et al., 2008), Hopfield networks (Hopfield, 1982), Elman networks (Elman, 1990) etc. can be used, but the work described in this thesis relies on a classical three layer network trained using Backpropagation (see section 3.3 for details).

The Margin Infused Relaxed Algorithm (MIRA) was initially designed as an online ML algorithm for multiclass classification by Cramer and Singer (2003). It is a modified version of a perceptron (a special ANN architecture with only two layers) learning algorithm, which works by processing the examples one by one and updating the weights so that the current example is correctly classified with a *margin* against the incorrect labels. Its functionality is extended by teaching the classifier to predict label bi–grams instead of unigrams and applying a dynamic programming algorithm such as Viterbi (Forney, 1973) in order to select an optimal sequence over a number of consecutive inputs that actually compose the example, thus making it suitable for the task presented in section 3.7.

Chapter 3

This chapter covers aspects regarding the text–to–speech processing required for any TTS synthesis system. The first section is a review of currently freely–available TTS synthesis systems and the methods and techniques they employ during the text pre–processing phase of TTS. As shown, most of the methods implemented in these systems are not state–of–the–art or offer limited performance on Romanian. The work further presented adapts state–of–the–art methods to Romanian or introduces original approaches such as lexical stress prediction and large–tagset part–of–speech labeling with Neural Networks.

Basic natural language processing for text–to–speech synthesis

3.1 General description of text–processing front–ends

As previously mentioned, the work inside a TTS system is divided between the NLP front–end and a speech processing back–end. The role of the front–end is to transform the input text into an intermediate representation which can be then used by the back–end in the voice synthesis process. In most TTS synthesis systems, the tasks performed by the front–end are two–fold: (1) text preparation and (2) feature extraction; this does not imply that the tasks are independent.

Text preparation ensures (a) the correctness of the input data by performing spellchecking, diacritic restoration and word–recasing and (b) normalization of the input data by expanding abbreviations, acronyms, numbers, dates and other formulas or numerical expressions into their spoken form.

In the *feature extraction* stage, the front–end converts the input text into the previously mentioned intermediate representation, which must aid the back–end in the process of speech generation. This representation is usually composed of phonemes with associated *local context features* and in some cases with additional prosodic features extracted from the *global context*.

The *local context* contains information such as: (1) part–of–speech, (2) current syllable, (3) next syllable, (4) previous syllable, (5) number of syllables of the current word, (6) syllable position inside the word, (7) syllable position inside the utterance, (8) next punctuation mark, (9) previous punctuation mark, (10) local phonetic information (surrounding phonemes), (11) articulatory features, etc.

Features extracted from the *global context* are usually designed to enhance the prosody modeling capabilities of the speech synthesis backend by providing information such as speaking style, word emphasis and intonation patterns.

3.2 Case study of DFKI MARY, Festival, Flite and FreeTTS[4]

Building the local context features for TTS synthesis requires that the words inside the input text are (1) syllabified and (2) phonetically transcribed and (3) that the stress information is known. These tasks are routinely achieved through the use of lexicons for known words, but regardless of the size and coverage of these lexicons, arbitrary texts are likely to contain previously unseen words that are referred to as out–of–vocabulary (OOV) words. Usually most of these words are either proper names, locations or technical terms and they abound inside any text starting from newspaper articles or novels to webpages that are likely to contain at least one proper name. Miss–processing of these words usually degrades the output quality of the speech synthesis system – thus it is highly important to integrate OOV word treatment inside the TTS flow.

Since the quality of the synthesized voice is significantly affected by the front–end's ability to correctly process OOV words, a review of the current freely available NLP frontends is critically important. Though a lot of research has been

[4] This review of open-source TTS systems was first presented as a section of the author's book chapter "Handling Two Difficult Challenges for Text-to-Speech Synthesis Systems: Out-of-Vocabulary Words and Prosody: A Case Study in Romanian", in Amy Neustein's book "Where Humans Meet Machines"

carried out for the above–mentioned NLP steps, and current state–of–the–art methods do yield good results, most of the existing open–source implementations rely either on poorly documented or outdated methods that offer only baseline performance. We enumerate several such TTS open–source systems which are mainly intended for English but, with proper lexicons and training data, can work for other languages:

- *DFKI MARY* (**M**odular **A**rchitecture for **R**esearch on speech s**Y**nthesis) is a result of the collaboration between DFKI and the Institute of Phonetics at Saarland University. It was initially designed for German (De) but it includes a Voice Creation Toolkit and it currently provides a TTS interface for English (En), Russian (Ru), Italian (It), Turkish (Tr) and Telugu (Te). Some of the modules for English use tools provided with Festival. MARY TTS uses a HMM POS Tagger implemented after TnT (Brants, 2000) and a custom method for letter–to–sound (LTS) conversion of OOV words;
- *Festival* is a TTS system developed at Carnegie Mellon University (CMU). The NLP processing module of Festival implements CLAWS for POS tagging (DeRose, 1988) and it uses the CART (Black et al, 1998) method for OOV words processing. Typical pronunciation errors in Festival have the following reasons[5]: (i) letter to sound rules fail on OOV words, (ii) foreign proper names often fail, (iii) wrong POS identified (newspaper headlines are particularly difficult), (iv) POS is right but it is not in the lexicon and (v) POS is not enough to differentiate pronunciation (and not yet dealt with by homograph disambiguation CART);
- *Flite* is a derivative of Festival and uses similar methods and techniques for OOV words processing;
- *FreeTTS*, which is based on Flite, is likewise a derivative of Festival and uses similar methods and techniques for OOV words processing.

[5] http://festvox.org/festtut/notes/festtut_toc.html#TOC42

Table 4: Details on open–source TTS systems NLP modules. Accuracy figures are shown on all / only–OOV words. For phonetic transcription and lexical stress assignment the statistics are computed on CMUDICT

System	POS–tagging	Phonetic Transcription	Syllabification	Lexical stress assignment
DFKI MARY	TnT 96.7% / 85.5%	ML/Under–documented – / –	Rule–based/ Under–documented – / –	Rule–based/ not documented – / –
Festival	similar to CLAWS 97% / –	CART with EM–derived L2P alignments – / 57.8%	Not documented – / –	CART – / 62.79%
Flite (based on Festival)	Not documented – / –	CART – / 57.8%	Not documented – / –	Not documented – / –
FreeTTS (based on Flite)	N/A – / –	CART – / 57.8%	Not documented – / –	Not documented – / –

To the author's knowledge (see Table 4), little information is available about the individual performance of each module embedded in the above mentioned TTS systems. Only letter–to–sound and POS tagging modules are reasonably documented, as they implement well established methods for performing subtasks in the text pre–processing step of TTS synthesis. Although some of these methods were considered to be optimal around a decade ago, they are now far behind the current state–of–the–art for designing well–functioning TTS systems. For example, the CART method (Black et al., 1998) for building letter–to–sound rules obtains an accuracy of 57.8% (measured using 10-fold validation) on OOV words when used on the English CMUDICT, while current state–of–the–art letter–to–sound methods (e.g. MIRA) have an accuracy of about 70% using the same data. Far less information is provided for other modules such as syllabification and lexical stress prediction. Most systems implement custom, undocumented methods for such tasks, and computing the OOV accuracy for them is complicated, requiring substantial effort.

3.3 Part–of–speech tagging[6]

3.3.1 Introduction

Part–of–speech tagging is a key process for various tasks such as information extraction, text–to–speech synthesis, word sense disambiguation and machine translation. It is also known as lexical ambiguity resolution and it represents the process of assigning a uniquely interpretable label to every word inside a sentence. The labels are called POS tags and the entire inventory of POS tags is called a tagset.

There are several approaches to part–of–speech tagging, such as Hidden Markov Models (HMM) (Brants, 2000), Maximum Entropy Classifiers (Berger et al., 1996; Ratnaparkhi, 1996), Bayesian Networks (Samuelsson, 1993), Neural Networks (Marques and Lopes, 1996) and Conditional Random Fields (CRF) (Lafferty et al., 2001). All these methods are primarily intended for English, which uses a relatively small tagset inventory, compared to highly inflectional languages. For the later mentioned languages, the lexicon tagsets (called morpho–syntactic descriptions (Calzolari and Monachini, 1995) or MSDs) may be 10–20 times or even larger than the best known tagsets for English. For instance, the Czech MSD tagset requires more than 3000 labels (Collins et al., 1999), Slovene more than 2000 labels (Erjavec and Krek, 2008), and Romanian more than 1100 (Tufiș, 1999). The standard tagging methods, using such large tagsets, face serious data sparseness problems due to lack of statistical evidence, manifested by the non–robustness of the language models. Accuracy decreases significantly when tagging new texts that are not in the same domain as the training data. Even tagging in–domain texts may not be satisfactorily accurate.

One of the most successful methods used for this task, called Tiered Tagging (Tufiș, 1999), exploits a reduced set of tags derived by removing several

[6] Section 3.3 is a close adaptation of the author's paper "Large tagset labeling using Feed Forward Neural Networks. Case study on Romanian Language" presented at ACL 2013 (Boroș et al., 2013), followed by relevant sections from "Improving the RACAI Neural Network MSD Tagger" (Boroș and Dumitrescu, 2013)

recoverable features from the lexicon morpho–syntactic descriptions. According to the MULTEXT EAST lexical specifications (Erjavec and Monachini, 1997), the Romanian tagset consists of a number of 614 MSD tags (by exploiting the case and gender regular syncretism) for wordforms and 10 punctuation tags (Tufiș et al., 1997), which is still significantly larger than the tagset of English. The MULTEX EAST version 4 (Erjavec, 2010) contains specifications for a total of 16 languages: Bulgarian, Croatian, Czech, Estonian, English, Hungarian, Romanian, Serbian, Slovene, the Resian dialect of Slovene, Macedonian, Persian, Polish, Russian, Slovak, and Ukrainian.

The strategy of the Tiered Tagging methodology is to use a reduced tagset (called CTAG–set), where a CTAG (from Corpus POS tags) is a generalization of a MSD, from which recoverable context–irrelevant features are removed. For instance, the attribute for gender (masculine 'm' or feminine 'f') from MSDs 'Ncfsrn' and 'Ncmsrn' is deleted to obtain the CTAG 'NSRN', because the gender information can be deterministically recovered based on the CTAG and the wordform itself. The recovering of the left–out attributes (Figure 1) is based on lexicons, linguistic rules, and, in the case of unknown words, on ML techniques (Ceauşu, 2006). When tagging with CTAGs, one can use any statistical POS tagging method such as HMMs, Maximum Entropy Classifiers, Bayesian Networks, CRFs, etc., followed by the CTAG to MSD recovery.

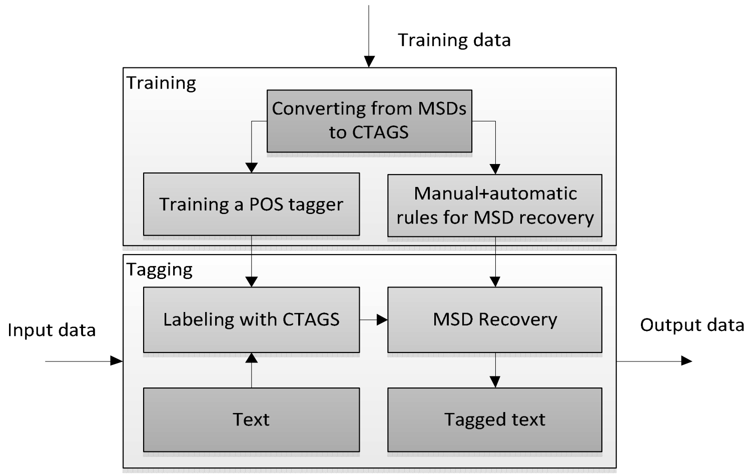


Figure 3 – Tiered Tagging methodology

The language dependent process of manually inferring linguistic rules for MSD recovery requires good knowledge of the target language and also extensive amounts of time invested in testing and re–design. It is difficult even for a native speaker to create such rules without in–depth linguistic knowledge.

An alternative solution based on local optimizations with feed–forward neural networks is further proposed. This method eliminates the need for the two stage processing, is much faster at run time, and is comparatively accurate with the Tiered Tagging implemented in the TTL tagger (Ion, 2007), available on the METASHARE Platform[7].

3.3.2 Large tagset part–of–speech tagging with feed–forward neural networks

Although removing recoverable attributes, as proposed in the Tiered Tagging approach, helps the goal of shrinking the lexical tagset to a reasonable size, valuable information for contextual disambiguation is also lost. For example, the gender agreement rule, valid in many languages, cannot be exploited unless the gender attribute is present in the tags.

The author's to deal with large tagsets without removing contextually useful information is based on Feed Forward Neural Networks (FFNN). FFNN are known for their simplicity and robustness in finding and recombining patterns inside data. Several neural network architectures have been proposed for the task of part–of–speech tagging. Schmid (1994) proposed a FFNN architecture for part–of–speech tagging obtaining a 96.22% accuracy. In his paper, he argues that neural networks are preferable to other methods, when the training set is small. He compares his results with a HMM tagger (94.24%) and a trigram tagger (96.02%), both trained and tested on the same corpora as his FFNN tagger (the Penn–Treebank corpus). A similar approach is presented by Marques and Lopes (1996). In their paper, the authors come to similar conclusions as those presented in Schmid (1994). The author's current research supports these findings with an additional argument, namely the better fit for managing large tagsets.

In both approaches mentioned before, the network is trained so that from the input vector, to output a real valued vector. Each value in the output vector

[7] http://ws.racai.ro:9191/

is generated by a distinct neuron, and corresponds to a unique tag in the tagset (e.g. 100 tags means the network contains 100 neurons on the output layer). The input vector for predicting the tag of the current word encodes the tags for the previously tagged words and the probable tags for the current and following two words, estimated using Maximum Likelihood Estimation (MLE):

$$P(t|w) = \frac{C(w,t)}{C(w)} \tag{11}$$

$P(t\|w)$	–	The probability of the word w having tag t
$C(w,t)$	–	The total number of times, the word w appears with tag t in the training corpus
$C(w)$	–	The total number of times, the word w appears in the training corpus

In the case of out–of–vocabulary (OOV) words, both approaches use suffix analysis to determine the most probable tags that can be assigned to the current word.

To clarify how these two methods work, if one wants to train the network to label the current word, using a context window of 1 (previous tag, current possible tags, and possible tags for the next word) and if there are, say 100 tags in the tagset, the input is a real valued vector of 300 sub–unit elements and the output is a vector which contains 100 elements, also sub–unit real numbers. As mentioned earlier, each value in the output vector corresponds to a distinct tag from tagset and the tag assigned to the current word is chosen to correspond to the maximum value inside the output vector.

The previously proposed methods still suffer from the same issue of data sparseness when applied to MSD tagging. However, in our approach, we overcome the problem through a different encoding of the input data.

The power of neural networks results mainly from their ability to attain activation functions over different patterns via their learning algorithm. By properly encoding the input sequence, the network chooses which input features contribute in determining the output features for MSDs (e.g. patterns composed of part of speech, gender, case, type etc. contribute independently in selecting the optimal output sequence). This way, the need for explicit MSD to CTAG conversion and MSD recovery from CTAGs is removed.

3.3.3 *The MSD binary encoding scheme*

A MSD language independently encodes a part of speech (POS) with the associated lexical attribute values as a string of positional ordered character codes (Erjavec, 2004). The first character is an upper case character denoting the part of speech (e.g. 'N' for nouns, 'V' for verbs, 'A' for adjectives, etc.) and the following characters (lower letters or '–') specify the instantiations of the characteristic lexical attributes of the POS. For example, the MSD 'Ncfsrn', specifies a noun (the first character is 'N') the type of which is common ('c', the second character), feminine gender ('f'), singular number ('s'), in nominative/accusative case ('r') and indefinite form ('n'). If a specific attribute is not relevant for a language, or for a given combination of feature–values, the character '–' is used in the corresponding position. For a language which does not morphologically mark the gender and definiteness features, the earlier exemplified MSD will be encoded as 'Nc–sr–'.

The following process is used in order to derive a binary vector for each of the 614 MSDs of Romanian:

- List and sort all possible POSes of Romanian (16 POSes) and form a binary vector with 16 positions in which position *k* is equal 1 only if the respective MSD has the corresponding POS (i.e. the *k*–th POS in the sorted list of POSes);
- List and sort all possible values of all lexical attributes (disregarding the wildcard '–') for all POSes (94 values) and form another binary vector with 94 positions such that the *k*–th position of this vector is 1 if the respective MSD has an attribute with the corresponding value;
- Concatenate the vectors from steps 1 and 2 and obtain the binary codification of a MSD as a 110–position binary vector.

3.3.4 *The training and tagging procedure*

The tagger automatically assigns four dummy tokens (two at the beginning and two at the end) to the target utterance and the neural network is trained to automatically assign a MSD given the context (two previously assigned tags and the possible tags for the current and following two words) of the current word (see below for details).

A training example consists of the features extracted for a single word inside an utterance as input and it's MSD within that utterance as output. The features are extracted from a window of 5 words centered on the current word. A single word is characterized by a vector that encodes either its assigned MSD or its possible MSDs. To encode the possible MSDs equation 12 is used, where each possible attribute *a*, has a single corresponding position inside the encoded vector.

$$P(a|w) = \frac{C(w,a)}{C(w)} \tag{12}$$

Note that the probability estimates are changed to take into account attributes not tags.

To be precise, for every word w_k, we obtain its input features by concatenating a number of 5 vectors. The first two vectors encode the MSDs assigned to the previous two words (w_{k-1} and w_{k-2}).The next three vectors are used to encode the possible MSDs for the current word (w_k) and the following two words (w_{k+1} and w_{k+2}).

During training, a list of suffixes with associated MSDs is compiled and it is later used at run–time to build the possible MSDs vector for unknown words. When such words are found within the test data, we approximate their possible MSDs vector using a variation of the method proposed by Brants (2000).

When the tagger is applied to a new utterance, the system iteratively calculates the output MSD for each individual word. Once a label has been assigned to a word, the word's associated vector is updated so it will have the value of 1 for each attribute present in its newly assigned MSD.

As a consequence of encoding each individual attribute separately for MSDs, the tagger can assign new tags (that were never associated with the current word in the training corpus). Although this is a nice behavior for dealing with unknown words it is often the case that it assigns attribute values that are not valid for the wordform. To overcome these types of errors, an additional list of words with their allowed MSDs ambiguity class is used. For an OOV word, the list is computed as a union from all MSDs that appeared with the suffixes that apply to that word.

When the tagger has to assign a MSD to a given word, it selects one from the possible wordform's MSDs in its wordform/MSDs associated list using a simple distance function:

$$\min_{e \in P} \sum_{k=0}^{n} |o_k - e_k| \tag{13}$$

P	–	The list of all possible MSDs for the given word
n	–	The length of the MSD encoding (110 bits)
o	–	The output of the Neural Network for the current word
e	–	Binary encoding for a MSD in P

3.3.5 Network hyper parameters

For this particular task, a feed forward neural network with 3 layers (1 input layer, 1 hidden layer and 1 output layer) and a sigmoid activation function (equation 14) was used. While other network architectures such as recurrent neural networks may prove to be more suitable for this task, they are extremely hard to train, thus, the advantages of such architectures were traded for the robustness and simplicity of the feed–forward networks.

$$f(t) = \frac{1}{1 + e^{-t}} \tag{14}$$

$f(t)$	–	Neuron output
t	–	The weighted sum of all the neuron outputs from the previous layer

Based on the size of the vectors used for MSD encoding, the output layer has 110 neurons and the input layer is composed of 550 (5 x 110) neurons.

In order to fully characterize our system, the following parameters were taken into account: accuracy, runtime speed, training speed, hidden layer configuration and the number of optimal training iterations. These parameters have complex dependencies and relations among each other. For example, the accuracy, the optimal number of training iterations, the training and the runtime speed are all highly dependent on the hidden layer configuration. Small hidden layers give high training and runtime speeds, but often under–fit the data. If the hidden layer is too large, it can easily over–fit the data and also has a negative impact on the training and runtime speed. The number of optimal training iterations changes with the size of the hidden layer (larger layers usually require more training iterations).

To obtain the trade–offs between the above mentioned parameters a number of experiments were performed, all of which relied on the “1984” MSD annotated corpus, which is composed of 118,025 words. Approximately 1/10 (11,960 words) of the training corpus was kept out for building a test set. The baseline accuracy on the test set (i.e. returning the most frequent tag in the ambiguity class) is 93.29%. An additional inflectional wordform/MSD lexicon composed of approximately 1 million hand–validated entries was also used.

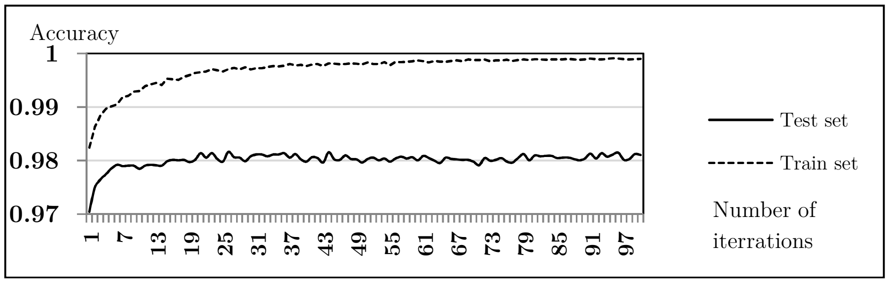


Figure 4 – 130 hidden layer network test and train set tagging accuracy as a function of the number of iterations

The first experiment was designed to determine the trade–off between the run–time speed and the size of the hidden layer. In this set of experiments the tagging accuracy was disregarded.

Table 5 – Execution time vs. number of neurons on the hidden layer[8]

Hidden size	Time (ms)	Words/sec
50	1530	7816
70	1888	6334
90	2345	5100
110	2781	4300
130	3518	3399

[8] The tests were performed on a Core I7-970 Extreme Edition at 3.2 Ghz, with 16GB of RAM

150	5052	2367
170	5466	2188
190	6734	1776
210	7096	1685
230	8332	1435
250	9576	1248
270	10350	1155
290	11080	1079
310	12364	967

Because, for a given number of neurons in the hidden layer, the tagging speed is independent on the tagging accuracy, several network configurations were partially trained (using one iteration and only 1000 training sentences). The first network only had 50 neurons in the hidden layer and for the next networks, and the hidden layer size was incremented by 20 neurons until it reached 310 neurons. The total number of tested networks is 14. After this, the tagging time for the 1984 test corpus (11,960 words) was measured using each individual model, as an average of 3 tagging runs in order to reduce the impact of the operating system load on the tagger (Table 5 shows the figures).

Determining the optimal size of the hidden layer is a very delicate subject and there are no perfect solutions, most of them being based on trial and error: small–sized hidden layers lead to under–fitting, while large hidden layers usually cause over–fitting. Also, because of the trade–off between runtime speed and the size of hidden layers, and if runtime speed is an important factor in a particular NLP application, then hidden layers with smaller number of neurons are preferable, as they surely do not over–fit the data and offer a noticeable speed boost.

Table 6 – Train and test accuracy rates for different hidden layer configurations

hidden layer	Train set accuracy	Test accuracy
50	99.18	97.95
70	99.20	98.02
90	99.27	98.03
110	99.29	98.05
130	99.35	98.12

150	99.35	98.09
170	99.41	98.07
190	99.40	98.10
210	99.40	98.21

As shown in Table 5, the runtime speed of our system shows a constant decay when we increase the hidden layer size. The same decay can be seen in the training speed, only this time by an order of magnitude larger. Because training a single network takes a lot of time, this experiment was designed to estimate the size of the hidden layer which offers good performance in tagging. To do this, we individually trained a number of networks in 30 iterations, using various hidden layer configurations (50, 70, 90, 110, 130, 150, 170, 190, and 210 neurons) and 5 initial random initializations of the weights. For each configuration, we stored the accuracy of reproducing the learning data (the tagging of the training corpus) and the accuracy on the unseen data (test sets). The results are shown in Table 6. Although a hidden layer of 210 neurons did not seem to over–fit the data, the experiment was ended, as the training time got significantly longer.

The next experiment was designed to see how the number of training iterations influences the tagging performance of networks with different hidden layer configurations. Intuitively, the training process must be stopped when the network begins to over–fit the data (i.e. the train set accuracy increases, but the test set accuracy drops). Our experiments indicate that this is not always the case, as in some situations the continuation of the training process leads to better results on the test data (as shown in Figure 4). So, the problem comes to determining which is the most stable configuration of the neural network (i.e. which hidden unit size will be most likely to return good results on the test set) and establish the number of iterations it takes for the system to be trained. To do this, the training procedure was run for 100 iterations and for each training iteration, the accuracy rate of every individual network on was measured on the test set (see Table 7 for the averaged values). As shown, the network configuration using 130 neurons on the hidden layer is most likely to produce better results on the cross–validation set regardless of the number of iterations.

Although, some other configurations provided better figures for the maximum accuracy, their average accuracy is lower than that of the 130 hidden unit network. Other good candidates are the 90 and 110 hidden unit networks, but not the larger valued ones, which display a lower average accuracy and also significantly slower tagging speeds.

The most suitable network configuration for a given task depends on the language, MSD encoding size, speed and accuracy requirements.

Table 7 – Average and maximum accuracy for various hidden layer configuration calculated over 100 training iterations on the test set

Hidden units	Avg. acc.	Max. acc.	St. dev.
50	97.94	98.37	0.139762
70	98.03	98.43	0.124996
90	98.07	98.39	0.134487
110	98.08	98.45	0.127109
130	98.14	98.44	0.136072
150	98.01	98.36	0.143324
170	97.94	98.36	0.122834

To obtain the accuracy of the system, in the last experiment the 130 hidden–unit network was used to perform the training/testing procedure on the 1984 corpus, using 10–fold validation and 30 random initializations. The final accuracy was computed as an average between all the accuracy figures measured at the end of the training process (after 40 iterations). The first 1/10 of the 1984 corpus on which we tuned the hyper–parameters was not included in the test data, but was used for training. The mean accuracy of the system (98.41%) was measured as an average of 270 values (9 test sets x 30 initializations).

3.3.6 Network pattern analysis

Using feed–forward neural networks gives the ability to outline what input features contribute to the selection of various MSD attribute values in the output layer which might help in reducing the tagset and thus, redesigning the network topology with beneficial effects both on the speed and accuracy.

To determine what input features contribute to the selection of certain MSD attribute values, one can analyze the weights inside the neural network and extract the input $\rightarrow$ output links that are formed during training. We used the network with 130 units on the hidden layer, which was previously trained for 100 iterations. Based on the input encoding, the features were divided between 5 groups (one group for each MSD inside the local context – two previous MSDs, current and following two possible MSDs). For a target attribute value (noun, gender feminine, gender masculine, etc.) and for each input group, the top 3 input values which support the decision of assigning the target value to the attribute (features that increase the output value) and the top 3 features which discourage this decision (features that decrease the output value) were selected. For clarity, the following notations will be used for the groups:

- G_{-2}: group one – the assigned MSD for the word at position i_{-2}
- G_{-1}: group two – the assigned MSD for the word at position i_{-1}
- G_0: group three – the possible MSDs for the word at position i
- G_1: group four– the possible MSDs for the word at position i_{+1}
- G_2: group five – the possible MSDs for the word at position i_{+2}

where i corresponds to the position of the word which is currently being tagged. Also, the attribute values are classified into two categories (C): *(P) want to see* (support the decision) and *(N) don't want to see* (discourage the decision).

Table 8 shows partial (G_{-1} G_0 G_1) examples of two target attribute values (cat=Noun and gender =Feminine) and their corresponding input features used for discrimination.

Table 8 – P/N features for various attribute values.

Target value	Group	C	Attribute values
Noun	G_{-1}	P	main (of a verb), **article**, masculine (gender of a noun/adjective
		N	particle, conjunctive particle, auxiliary (of a verb), demonstrative (of a pronoun)
	G_0	P	**noun, common/proper (of a noun)**
		N	adverb, pronoun, numeral, interrogative/relative (of a pronoun)
	G_1	P	**genitive/dative (of a noun/adjective)**, particle, punctuation
		N	conjunctive particle, strong (of a pronoun), non–definite (of a noun/adjective), exclamation mark
Fem.	G_{-1}	P	main (of a verb), preposition, **feminine (of a**

			noun/adjective)
		N	auxiliary (of a verb), particle, demonstrative (of a pronoun)
	G_0	P	**feminine (of a noun/adjective)**, nominative/accusative (of a noun/adjective), past (of a verb)
		N	masculine (of a noun/adjective), auxiliary (of a verb), interrogative/relative (of a pronoun), adverb
	G_1	P	dative/genitive (of a noun/adjective), indicative (of a verb), **feminine (of a noun/adjective)**
		N	conjunctive particle, future particle, nominative/accusative (of a noun/adjective)

For instance, when deciding on whether to give a noun (N) label to current position (G_0), it can be seen that the neural network has learned some interesting dependencies: at position G_{-1} we find an article (which frequently determines a noun) and at the current position it is very important for the word being tagged to actually be a common or proper noun (either by lexicon lookup or by suffix guessing) and not be an adverb, pronoun or numeral (POSes that cannot be found in the typical ambiguity class of a noun). At the next position of the target (G_1) we also find a noun in genitive or dative, corresponding to a frequent construction in Romanian, e.g. "mașina băiatului" being a sequence of two nouns, the second at genitive/dative.

If the neural network outputs the feminine gender to its current MSD, one may see that it has actually learned the agreement rules (at least locally): the feminine gender is present both before (G_{-1}) the target word as well as after it (G_1).

3.3.7 Genetic optimization for the Neural POS tagger

One of the main problems with neural networks is their tendency for over–fitting especially when using a relatively small corpus. In the pattern analysis process the feminine gender attribute was positively influenced by the feminine gender attribute of a neighboring noun/adjective, but it is also positively influenced by the presence of a preposition or a main verb, which, from a linguistic perspective should not happen, and it is most likely an effect of over–fitting.

One way to cope with over–fitting is to disable certain links between neurons that would otherwise produce this undesired effect. Genetic algorithms have been successfully applied for finding optimal network topologies (Schaffer et al, 1992; Fischer et al, 1998; Fiszelew et al., 2007). They excel at this task because they can explore many different parts of a large solution and narrow down the search space in comparison to full–grid optimizations. Starting from an initial random population and based on the fitness of each individual, the best suited candidates are recombined using different methods. If the fitness and recombination methods are properly designed, genetic algorithms have the ability to generate new solutions to a problem while also preserving partial ones.

To assess the performance of genetic algorithms in this particular task, three experiments were applied to both Romanian and Czech (see section 3.3.8 for results). It is important to mention that while the Romanian tagset contain about 600 tags, which is considered difficult to handle by standard POS tagging methods, the Czech inventory is composed of much larger number of **more than 3000 tags**. In all the experiments the multilingual „1984" annotated corpus (Romanian and Czech versions) was used, maintaining the 10–fold cross validation method for testing. In the first experiment (E1), a custom designed topology and in the second (E2) and third (E3) experiments two different genetic approaches were used to determine an optimal network topology. As a design choice, the neural network used in E2 has a 50 neuron hidden layer because the training/testing procedure takes a lot of time on larger networks and also because a smaller hidden layer helps avoid having a sparse network given the fact that roughly half of the synapses will be disabled by the genetic algorithm.

Tagging accuracy is measured as an average after 50 training iterations and 10 random initializations (for each topology).

3.3.7.1 Experiment E1: Manually designed simplified network topology

In the first experiment, a hand–made custom network topology in which only attributes belonging to the same class were fully connected. For example, the gender attributes (m/f) from all neighboring MSDs were fully connected to a variable number of neurons in the hidden layer, which again was connected to the neurons used to encode the gender attribute of the output MSD in the output layer.

The motivation behind manually designing this topology was to provide another baseline that was as simple as possible, as opposed to the standard fully connected network that was previously presented.

Experiments were performed with a number of different combinations of the number of neurons in the hidden layer used for each individual attribute group. Because the purpose was to establish a baseline system, the topology with the highest yielding results was selected.

3.3.7.2 Experiment E2: Unrestricted genetic algorithm generated topology

In the second experiment (E2) a genetic algorithm was used to find a custom network topology. Each individual belonging to a population is represented by a binary vector and each value inside this vector is associated to a synapse inside the neural network, on any layer. The binary values describing the individuals are used to select the state of the synapses (0 – disabled, 1 – enabled). During training and testing, the network disregards the disabled synapses. The genetic algorithm uses a uniform cross–over probability of 50%, with a 10% chance of mutation. Based on the fitness function, the first two best–fit individuals are kept in the new population. The following individuals are generated from the Cartesian product of the first n–best individuals[9]. Each pair of individuals spawns two children. The accuracy of the tagger on a test set was used as a measurement of fitness.

3.3.7.3 Experiment E3: Restricted genetic algorithm generated topology

In E2 synapses were randomly enabled or disabled both between the input and hidden layer and between the hidden and the output layer. In the third experiment (E3), while similar to E2, the genetic algorithm was adapted to take into account attribute groups so that we enable or disable all synapses that link groups from different layers. As such, an individual is represented by a vector whose bits encode whether a specific group in a layer should connect to another group in the next layer.

The size of the hidden layer was fixed to be equal to the size of a MSD (110 neurons for Romanian and 135 for Czech), to facilitate the segmentation

[9] In our experiments we used n=5

procedure. Each individual layer was segmented into groups of neurons in accordance with the attribute types. The intuition behind E3 was that the links between different attribute types should all be either enabled or disabled at a time rather than just randomly some of them as is the case for E2. This change leads to a reduced number of possible variations that can be generated by the genetic algorithm.

The accuracy was also measured as an average after 50 training iterations and 10 random initializations (for each topology).

3.3.8 Experiment results

Table 9 summarizes the accuracy figures obtained on Romanian and Czech by the four network topologies: the unmodified, fully connected standard topology (U), the manually designed simplified topology (E1), the topology generated by the unrestricted genetic algorithm (E2) and the topology generated by the restricted genetic algorithm (E3). For experiments E2 and E3, the values for the first two best–fit individuals I1 and I2 are provided.

Table 9. Manual and genetic network topology results

Language	Unmodified topology (U)	Manual topology (E1)	Unrestricted genetic (E2)	Restricted genetic (E3)
Romanian	97.96%	97.17%	I1: **98.17%**	I1: 98.09%
			I2: 98.12%	I2: 98.05%
Czech[10]	90.70%	90.76%	I1: **91.69%**	I1: 91.06%
			I2: 91.64%	I2: 90.04%

As shown, the manually designed topology (E1) provides slightly better results than the unmodified topology on Czech, but performs worse on Romanian. The intuition behind E1 was that local agreements between attribute values should only appear between similar attribute groups (e.g. the gender of a noun should be influenced by the gender of a neighboring adjective). However, the

[10] Because of the complexity of the tagset inventory, the accuracy figures for Czech are lower than the baseline accuracy for Romanian. The baseline accuracy for Czech is 88%.

results obtained in experiment U and E1 on Romanian show that there are dependencies across different attribute groups. Somewhat surprisingly, the unrestricted genetic algorithm (E2) obtained better candidates than the restricted algorithm (E3). To make sure the results are reliable, we repeated experiments E2 and E3 several times, each time obtaining similar accuracy figures.

As shown in table 10 besides improving the accuracy of the system, a custom network topology also provides increased tagging speeds. The results were computed by averaging the time it took to re–label the entire training sets for both Romanian and Czech. The experiment was repeated three times in order to reduce the effect of system load on tagging speed.

Table 10. Tagging speeds of different network topologies[11]

Topology	Romanian **Words/sec**	Czech **Words/sec**
U	7,183	6,981
E1	14,366	13,962
E2–I1	**14,659**	**14,189**
E3–I1	11,402	12,466

3.3.9 Conclusions

A new approach for large tagset part–of–speech tagging using neural networks was presented. An advantage of using this methodology is that it does not require extensive knowledge about the grammar of the target language. When building a new MSD tagger for a new language one is only required to provide the training data and create an appropriate MSD encoding system and as shown, the MSD encoding algorithm is fairly simple and our proposed version works for any other MSD compatible encoding, regardless of the language.

Observing which features do not participate in any decision helps design custom topologies for the Neural Network, and provides enhancements in both speed and accuracy. The configurable nature of the system allows users to provide their own MSD encodings, which permits them to mask certain features that are not useful for a given NLP application.

[11] The tests were performed on a Core I7-970 Extreme Edition at 3.2 Ghz, with 16GB of RAM

If one wants to process a large amount of text and is interested only in assigning grammatical categories to words, he can use a MSD encoding in which he strips off all unnecessary features. Thus, the number of necessary neurons would decrease, which assures faster training and tagging. This is of course possible in any other tagging approaches, but our framework supports this by masking attributes inside the MSD encoding configuration file, without having to change anything else in the training corpus. During testing the system only verifies if the MSD encodings are identical and the displayed accuracy directly reflects the performance of the system on the simplified tagging schema.

A genetic method for selecting a network configuration was also successfully applied showing that custom designed network topologies are able to deliver better accuracy figures and faster tagging speeds.

3.4 Diacritic restoration and word casing[12]

The diacritic restoration problem is particularly relevant for the Romanian language. Diacritic restoration is one type of spelling correction in which the correct diacritical mark of a letter is inserted in a word which would otherwise be incorrect, have a different (unintended) meaning or violate different syntactic constraints for the language in question. Thus, the decision to insert a diacritic is based on the context of the word and, for Romanian, we differentiate among the following cases:

- The word is incorrect according to a predefined (large) lexicon but a diacritic version of it exists in the lexicon, e.g. ”mașina” is correct, ”masina” is not;
- The word does not possess the correct diacritic form to agree with its syntactic constraints, e.g. the indefinite noun in “o mamă” (”mother”) is correct but its definite form is not ”o mama” (”the a mother”);
- The word does not have the intended meaning in context, e.g. the word ”fata” means ”the girl” but word ”fața” means ”the face”.

[12] Section 3.4 is a close adaptation of the author’s paper “A unified corpora-based approach to Diacritic Restoration and Word Casing” accepted for publication at the Language Technology Conference (LTC) 2013

In Romanian (cf. Tufiș and Ceaușu, 2008), on average, every third word of an arbitrary text contains at least one diacritical character. In terms of characters, more than 8.2% have diacritical signs.

Word casing is another type of spell–checking in which a word is corrected by recapitalizing its letters. It is particularly relevant for part–of–speech tagging and machine translation since it reduces a number of ambiguities that arise in the processing of incorrectly spelled words (e.g. the misspelled sequence “marea neagră” that translates into „large dark” versus the correctly spelled sequence „Marea Neagră” which means “Black Sea”).

Both diacritic restoration and word casing tasks can be viewed as a single abstract problem: given a sequence of words (that contains any number of incorrect words), attempt to correct it by generating for every word a series of possibly correct variants and then choosing one in such a way that the chosen sequence maximizes its global probability with respect to a given language model (LM). The algorithm that maximizes this probability is a Viterbi–based decoder that computes the sequence probability estimates using a trigram HMM LM (equation 15).

$$\underset{v_1 \dots v_n}{\operatorname{argmax}} \left[\prod_{i=1}^{V} P(v_i | v_{i-2}, v_{i-1}) \, P(w_i | v_i) \right] \tag{15}$$

where

$v_1 \dots v_n$	represent word alternatives
$w_1 \dots w_n$	original sequence words
$P(v_i \| v_{i-2}, v_{i-1})$	the transition probability from v_{i-2}, v_{i-1} to v_i
$P(w_i \| v_i)$	the probability of observing word w_i given v_i

By finding the optimal state sequence we actually select the word alternatives that are likely to be the correct words based on the used LM. We refer to this as surface processing because it relies only on word–forms (words) that appear in the text without using any other information (e.g. part–of–speech tags, text segmentation, named entity labels, etc.).

In all our experiments we used the fixed value of 1 for $P(w_i|v_i)$, because $P(w_i|v_i) = \frac{C(w_i,v_i)}{C(v_i)}$, where $C(w_i, v_i)$ is the number of times word v_i appears with the diacritical–stripped form w_i and $C(v_i)$ is the number of times v_i appears with other forms, thus making the equation constant. In other words, the system only relies on the LM to find the optimal state sequence. This is true for diacritic restoration and word–recasing. However, if faced with the task of spelling correction using the same settings and access to a corpus of raw sentences and their manually corrected versions, it is possible for a single variant v_k to be associated with more than one incorrect word–form, thus making it mandatory to compute the probability of a misspelling. Such a corpus however is hard to obtain for English and almost impossible to get for Romanian without involving a time and resource consuming process of manually compiling this corpus by manually correcting sentences. In practice, spellchecking can be performed without requiring the use of such a corpora and it will be later addressed in section 3.5.

3.4.1 Evaluation

This section presents our experiments with the proposed system. However, we must note that the system's performance is entirely dependent on the LM used. As such, in the next section we present the process of preparing the corpus from which the LM is generated from before moving to the experiments themselves.

3.4.1.1 System setup and testing procedure

We start with an initial Romanian corpus of 5.2M sentences. The corpus consists of smaller corpora of different genres and sizes, presented in table 11.

Table 11. Corpus composition

Corpus	Genre	Size (# of lines)
DGT–TM	Legal	1.7M
Europarl	Legal	400K
Wikipedia–S	Encyclopedia	2.7M
SETIMES	News	100K
Varied novels	Novel	240K
TED	Free speech	156K

DGT–TM[13] (Steinberger et al., 2012) is a processed subset of the Aquis Communautaire[14] meant as a multilingual resource for MT. We used only the

[13] http://langtech.jrc.it/DGT-TM.html

Romanian side of the multilingual parallel Ro–En corpus. Europarl[15] (Koehn, 2005) is a juridical corpus but it has a different register than DGT–TM. Wikipedia–S is also a parallel Ro–En corpus obtained by using the Lexacc extraction tool (Ștefănescu et al., 2012), from which we used only the Romanian side. The SETimes[16] resource represents a news corpus. The novels were gathered from a free online source[17], most of them belonging to the Sci–Fi genre. Finally, TED[18] is a transcribed speech corpus.

A series of processing and cleaning steps were applied:

1. The different corpora were joined into a single file;

2. Letters ș and ț have been corrected from the old format with a cedilla underneath to the standard comma diacritic: ş→ș and ţ→ț. Romanian texts older than a few years have this common problem due to Microsoft Windows not using Unicode by default until Windows Vista. As such, currently, even official sources still use the legacy cedilla diacritics.

3. The following sentences have been removed: sentences that contain less than 3 words; that contain only uppercased letters; that contain no diacritics; that contain foreign words (this has been done by forcing all sentences to contain at least 80% of words that are at least 3 letters long to exist in a Romanian lexicon). This step reduced the number of lines in the corpus by 1.6M, to a total of 3.56M, containing 85.6M words.

4. The corpus was tokenized. 50K sentences have been randomly extracted from the corpus and set aside as the *reference file*; the corpus is further referenced as the *training corpus/file*. The *reference file* was stripped of diacritics and kept as the *test file* that will be given to the system as input.

5. At this point we keep the *training*, *reference* and *test* files as final. We then lowercase and keep them as well for the lowercase–only tests for the diacritic restoration experiment.

[14] http://eurovoc.europa.eu/
[15] http://www.statmt.org/europarl/
[16] http://www.setimes.com/
[17] http://www.gutenberg.org/
[18] http://www.ted.com

6. We generate LMs from the *train files* (both the unmodified and lowercased versions) yielding two language models, further referenced in the article as the *LM* and the *lowercase LM*. The *LM* contains 1.01M 1–grams, 10.9M 2–grams and 7,2M 3–grams, while the *lowercased LM* contains a marginally smaller number of ngrams (0.9M, 10.2M and 7.2M 1–,2– and 3–grams).

The testing procedure is simple: we give the system the test file as input, obtaining an output file (either with the diacritics restored or with the words cased, or both) that is compared with the reference file. We give two types of word accuracy rate (WAR) measurements: all–word and targeted: all–word means that from the total number of words we subtract the number of incorrect words and we divide everything by the total number of words; targeted means that we divide the number of correct words to the number of targeted words (depending on the experiment, targeted words will mean words that contain diacritics and should be corrected (E1), words that should be case corrected (E2), and words that should be diacritic and/or case corrected (E3).

For the diacritic restoration experiment the input files do not contain any diacritics; for the word casing experiment the input file is fully lowercased (but contains the original diacritics); for the final experiment the input file will be all lowercased and without diacritics.

3.4.1.2 Experiment 1: Diacritic Restoration

Diacritic restoration means that given a sentence, words that do not have diacritics but should, are correctly transformed into their correct form.

For example, the sentence "O fata sta in fata." is corrected to "O fat**ă** st**ă** **în** fa**ță**." (translation: A girl sits in front). The restoration algorithm must make the correct global choice for each word's alternatives array. The spelling alternatives are initially generated from a Romanian lexicon as follows: every word from the lexicon has its diacritics removed; for each such word the lexicon is scanned again and every word that is a spelling alternative is added into an array of alternatives. For example, for word "fa**ță**" (translation: front or face) after diacritics removal of every word in the lexicon ("fa**ță**" now becomes "fata" (note: "fata" is also a valid word in Romanian, meaning the definite form of the noun "girl")) we want to add its spelling alternatives: we scan the lexicon again and find "fat**ă**" ("a girl"–noun or "to give birth/to calve/to whelp"–verb, used in the context of animals giving birth), "fata" ("the girl"), "fa**ță**" ("face" or "front"), "fa**ța**" (the definite form of "fa**ță**"), "f**ă**ta" (the root form of the verb "to give birth"), etc. In our example, there are two "fata" words in the sentence; each will receive the same spelling alternatives list, and the Viterbi decoder helped by the LM must choose the correct form for each word. Even if a word has no spelling alternatives, the diacritics stripped form is always added, so every word has at least one spelling alternative that the Viterbi decoder can choose from. Most proper nouns (those that are not in the lexicon), numbers, foreign words, dates, etc., have only a single spelling alternative and thus pass the diacritic restoration process unaltered.

We performed two tests: one on the lowercased test file and one on the unmodified test file. For each test we used the appropriate LM. Results are presented in table 12. The test file has a total of 1,198,539 words, out of which 265,128 have diacritics (meaning a 22.12%).

Table 12. Diacritic restoration results

Test type	All–Word WAR	Targeted WAR
Lowercased test	99.31 %	96.89 %
Unmodified test	99.28 %	96.77 %

We believe that a 96% targeted Word Accuracy Rate (WAR) figure makes the system a powerful tool for error correction. A manual evaluation of the results has yielded a number of observations:

- The results can be further improved if we do not consider proper nouns in the WAR evaluation. A large number of proper nouns have not been "corrected" because they are not present in either the LM or the lexicon. For example "puskin" was not corrected to "pușkin" in the sentence "*lucrarea este inspirată de poemul evgheni oneghin al lui aleksandr puskin*";
- proper nouns mishandling also introduce another error type: in the lowercase test, proper noun "franța" (translation: "France") was corrected to "frântă" (translation: "broken" – adjective, feminine form) as in the sentence "*cel mai bun pianist din frântă.*"
- a significant fraction of the words that have not been correctly restored is due to the fact that they are misspelled: for example, a few of the imported novels that are part of the corpus and implicitly from the test file have words that are split into syllables at the end of the line as in "*mergând în direcția opu-sa , înspre nord*", "opu–sa" was impossible to correct.

- a quick overview of the “legitimate” nouns that should have been corrected but were not, were actually left unchanged. This is a “desired” behavior for specialized tasks as MT, where a part–of–speech tagging preprocessing phase would do better to label a word as unknown rather than misclassify it and subsequently get translated into a wrong word.

3.4.1.3 Experiment 2: Word Casing

Word–casing is the task of determining whether a word should be written in lower case, upper case, or capitalized first letter. It is often the case where incorrectly written words impede tasks such as Part–of–Speech tagging from correctly labeling a word, or MT from being able to find appropriate translation equivalents.

The word casing process differs slightly from the diacritic restoration in respect to the alternatives array generated for every word in a sentence. Whereas for diacritic restoration the alternatives array was filled with spelling variants of the same diacritic–free word, for word casing, the alternatives array is filled with exactly three alternatives for every word: the lowercased word, the full uppercased word and the first–letter–capitalized word. Ex: for word “franța”, the alternatives array will contain “franța”, “FRANȚA” and “Franța”. Here too we perform two tests. For the first test each spelling alternative in the alternatives array has the same emission probability (test 1: equal emission probability). For the second test, for each word, we perform a lexicon lookup: if the word is found in the lexicon, nothing changes; if the word is not found, then we increase the emission probability of the first letter capitalized spelling alternatives at the expense of the other two options (which remain equally probable). The second test is named lexicon biased emission probability. The Viterbi decodes remains unchanged. In this experiment we use only the unmodified LM. Results are presented in table 13. The test file has a total of 149,538 words that should be capitalized, a 12.47 % out of all words.

Table 13 – Word Casing results

Test type	All–Word WAR	Targeted WAR
Equal emission probability	98.78%	92.13%
Lexicon biased emission probability	99.27%	94.17%

As with the diacritic restoration experiment, we can draw a number of conclusions:

- the type of error that is made most of the time is that of leaving the word unchanged whereas it should have its first letter capitalized or entirely uppercased;
- there are cases where the correct capitalization is neither first letter or the entire word. For example, “mig” should have become “MiG” in sentence “*…MiG – 29OVT , care este …*”. This type of error however is very difficult to solve. There are two plausible solutions: 1. Have a large list of named entities with non–standard capitalization and 2. Generating all possible lowercase/uppercase letter variations. But, solution 1 will never be complete and solution 2 is basically exponential; thus, this error type remains for future study.

With this experiment we have a similar problem as with the previous: we have no gold standard on which to scale our results against. We can only speculate that a 92+ percent WAR is a good result, qualifying our system as usable in sensitive tasks such as MT.

3.4.1.4 Experiment 3: Jointly processing Diacritic Restoration and Word Casing

The final experiment treats both problems as unitary. The only change that is made is in the alternatives array. Whereas in the diacritic restoration task the alternatives array would be filled with spelling alternatives and in the word casing task with lowercase, uppercase and first capitalized letter, in the joint task the alternatives array will be filled with the three capitalization variants for each of the spelling alternatives. The test file given as input was lowercased and had all diacritics removed. Results are presented in table 14 and table 15 presents the WAR differences when compared to experiments E1 and E2.

Table 14 – Joint task results

Component test	All–Word WAR	Targeted WAR
Diacritic Restoration	99.29 %	96.82 %
Word Casing	99.02 %	92.19 %

Table 15 – Comparison with previous experiments

Experiment comparison	All–Word WAR	Targeted WAR
vs. Diacritic Restoration (lowercased)	–0.02 %	–0.05 %
vs. Diacritic Restoration (non–lowercased)	+0.01 %	+0.05 %
vs. Word Casing (lexicon biased)	–0.25 %	–1.98 %

A performance drop is to be expected versus the individual experiments due to the fact that the alternatives the Viterbi decoder has to choose from are increased three–fold versus the diacritic restoration only, and multiplied by the number of spelling alternatives versus the word casing task. We argue that a targeted WAR drop of a .05% vs. diacritic restoration only and of almost 2 percent vs. word casing is perfectly acceptable, given that both tasks are solved in a single pass.

In this experiment the LM used was standardly generated from the training corpora without performing lower–casing or stripping diacritics from the date.

Another possible experiment is to assess the diacritic restoration and word–casing performance in a sequential approach. However, if we would chain the output of one system to the input of the other system we would surely obtain worst results since none of the modules has a perfect accuracy. The only other viable alternative is to perform diacritic restoration and word–casing separately and then match the cases on the two obtained results.

3.4.2 Conclusions

We propose a surface processing system that attempts to solve two important problems: diacritic restoration and word casing. At its core, the system uses a Viterbi algorithm to select the optimal state sequence from a variable number of possible options for each word in a sentence. The options (stored in an alternatives array) are generated using a lexicon (for the diacritic restoration task) or directly in the code (three capitalization alternatives for every word). The Viterbi transition score is given by calculating the perplexity of tri–grams (chosen from alternatives arrays) against a previously trained language model.

The system is language independent. The only change in its code required to adapt it from Romanian to any other language is to specify the target language's diacritics.

Regarding external resources, the system uses a language model and a lexicon. The language model is used to estimate a transition scores for the sequential alternatives generated from the lexicon. The use of the system for another language thus also requires the existence of a language model and a lexicon (though the lexicon is only used in the diacritic restoration task).

The results we presented here are encouraging. More work on the system, as well as possible other NLP techniques applied to it (such as text chunking or the usage of large proper noun lists) will certainly increase word accuracy figures even more. We couldn't perform an external evaluation of our system as we could not find any reference corpus for these tasks, or, for that matter, obtain any other systems (or parts of systems) performing them that did not require extensive modifications or specialized resources to make them functional. For example, we have found a few websites that allow only a limited number of sentences at a time to have their diacritics restored – manually evaluating a few sentences we noted that our approach provides significantly better results; there are office plug–ins that perform diacritic restoration, but all are closed–source and not easily adaptable for automatic testing.

3.5 Spellchecking

The spellchecker for Romanian and English is a corpora–based method, thoroughly presented in Ştefănescu et al. (2011), which combines three algorithms for spellchecking using a voting mechanism and it is designed to produce alternative spellings with a decision threshold to replace words within an utterance. The three algorithms use similar approaches to spellchecking:

a. Detect if a word is correctly spelled using dictionaries;
b. If the word is not found in any lexicon, produce spelling alternatives by deleting, replacing or adding letters and spaces;
c. Re–rank spelling alternatives for the entire utterance using n–gram frequencies (see equation 16).

The system's performance was assessed during the Microsoft Speller Challenge Competition (placing 4th) in which it obtained an F–score of 97% on the TREC DATASET (Qin et al., 2007) (English search queries).

$$S(q1) = \left(\alpha \sum P(w_i) + \beta \sum P(w_i, w_{i+1}) + \gamma \sum P(w_1, w_{i+1}, w_{i+2})\right) F_{(q,q1)} \quad (16)$$

α, β, γ	– weights
$P(w_i)$, $P(w_i,w_{i+1})$ and $P(w_i,w_{i+1},w_{i+2})$	– n–grams log probabilities

$F_{(q,q1)}$– Factor dependent on Levenshtein distance between the original query q and the spelling variant q_1.

3.6 Text normalization

This step is responsible for expanding certain expressions into words, before speech synthesis can happen. Arbitrary texts contain numbers, dates, abbreviations, acronyms, numbers, etc., which are not suitable for a direct conversion into a phonetic representation.

Text normalization raises a series of challenges mainly because of the ambiguities that may occur in selecting the optimal expansion course for each type of expression (number, abbreviation, acronym etc.).

The input text is normalized using a set of handwritten rules and a list of well–known abbreviations. Every DOT character is removed from the text (after the expansion of abbreviations and expressions) except for sentence boundaries. Known abbreviations and some expression types are expanded and other (unknown) abbreviations or acronyms are converted to spoken form based on a letter by letter rule (eg. „LRC" --> "lerece"[19]). The accuracy of this module cannot be measured on OOV words, because there is no possible way of determining how to expand an abbreviation which is not found within the lexicon.

[19] This example corresponds to a conversion for Romanian

3.7 A unified lexical processing framework based on the Margin Infused Relaxed Algorithm[20]

As previously mentioned the output of the text pre–processing and analysis front–end is a set of features that is later used by the speech synthesis back–end in its speech synthesis process. The typical feature set is composed of: (1) part–of–speech, (2) current syllable, (3) next syllable, (4) previous syllable, (5) number of word syllables, (6) syllable position inside the word, (7) syllable position inside the utterance, (8) next punctuation mark, (9) previous punctuation mark, (10) local phonetic information (surrounding phonemes) and (11) other articulatory features. Extrapolating from this feature set, the derived basic tasks that a front–end must perform are: (1) syllabification, (2) phonetic transcription (grapheme to phoneme conversion – G2P), (3) stress prediction and in some cases (4) lemmatization helps in extracting other features which will be further addressed in section 7.

There are various methods proposed in the literature for each of the previously mentioned subtasks of TTS. For each of them, a short literature review of available methods will be provided and the results obtained with the new proposed unified methodology will be compared to current state–of–art systems. The previously proposed methods vary from rule–based to data–driven and different authors employ different classifiers (in data–driven approaches), such as Maximum Entropy Classifiers, Classification and Regression Trees, Support Vector Machines (SVM), Structured SVMs, Conditional Random Fields (CRFs), etc. While these are all powerful methodologies, the newly introduced unified approach uses the *Perceptron classifier with the MIRA update learning* in a sequence labeling setting, because of its robustness and its ability to obtain highly accurate results that compare to the ones obtained using CRFs. All the lexical processing methods that are proposed share the following similarities:

- All of them are reformulated as sequence labeling tasks;
- The same classifier for all the tasks (MIRA);

[20] Note: section 3.7 is an adaptation of the author's paper "A unified lexical processing framework based on the Margin Infused Relaxed Algorithm. Case study on Romanian Language" (Boroş, 2013), which was presented at the Recent Advances in Natural Language Processing (RANLP 2013) Conference.

- The classification context is based on different and mostly orthographic (except for lemmatization and lexical stress prediction, which use the morpho–syntactic context) feature sets;
- The performance is measured in terms of word accuracy rates (WAR);
- All the tests are reported on OOV words, lookup lexicons can be used for known words;
- All the tests are performed on Romanian;
- The trial and error process involved in the design of feature set is skipped and only the set which yielded the best results is described.

3.7.1 Syllabification

Syllabification is the process of decomposing words into their phonological units, which is an important requirement in modern approaches to TTS synthesis and speech recognition.

All languages have phonetic rules that govern the syllabification process, but it is often the case that these rules are contradicted by etymological principles, a fact which complicates the task of automatic syllabification. Phonetic transcription or the position of the lexical stress both provide useful information for syllabification, but more often than not, G2P and lexical stress are not accurate enough on OOV words to help the syllabification process. Also, syllabification lexicons are usually larger than G2P lexicons, thus providing more training data, which helps the syllabification system obtain better results than G2P. Because of the above mentioned reasons, the proposed syllabification method is based on purely ortographic features (i.e. the word’s letters).

Several algorithms have been proposed for the syllabification task divided between rule–based and data–driven. While, rule–based methods are centered on theoretical aspects of the syllabification problem, data–driven methods are usually preferable, since they are language independent and they only require the construction of syllabified words lexicons.

In the following description, the term *juncture point* is used to denote the places where hyphen marks (syllable breaks) are placed within a word.

The look–up procedure was introduced by Weijters (1991). It constructs a table of letter n–grams from the training corpus and uses this table to predict juncture points. Each n–gram contains the *focus character* (the character that is being analyzed to determine if a juncture point should or should not occur after) with left and right context, including hyphen marks. When syllabification is performed on a new word, the algorithm determines if a focus character should be followed by a hyphen, using the majority of similar n–grams.

The IB1 (Daelemans et al., 1997) algorithm creates n–grams (of predetermined size) from word juncture points and stores them into a database. When a new word has to be split into syllables, every n–gram around the word's possible junctures is matched against the n–grams already available from the training step. N–grams are compared using a distance measure to determine how similar two n–grams are to one another.

Marchand and Damper (2007) introduced Syllabification by Analogy (SbA) which follows the principles of the Pronunciation by Analogy (PbA) algorithm. It works by applying a "full pattern match" on the input string using entries in a dictionary compiled from the training corpora. Marchand and Damper also investigate the possibility of using syllabification to improve grapheme to phoneme performance on English words.

Barlett et al. (2008) use structured SVMs to predict tags for letters in a given word and compare results obtained using different tagging strategies. Their method outperforms the results of the SbA method.

3.7.1.1 Syllabification with MIRA

The proposed sequence labeling approach is inspired after Barlett et al. (2008). In their paper they experimented with different tagging strategies and according to their results, the numbered ONC (onset–nucleus–coda) achieved the highest performance. This is why the same tagging strategy was employed for the proposed approach. *The main difference between this approach and theirs is the feature set that was designed and the classifier that was used (MIRA).*

A widely accepted fact is that a syllable is composed of a *nucleus* vowel with or without surrounding consonants which are divided into the *onset* (the consonants preceding the vowel) and the *coda* (the consonants succeeding the vowel) (Breen and Pensalfini, 1999). The ONC tagging strategy assigns a tag to every letter of a word based on its role inside the parent syllable. There are three types of tags: O–onset, N–nucleus and C–coda. The numbered ONC makes every tag unique, *inside a syllable*, by adding an index to the tag. To exemplify, we will use the syllabification of the Romanian word "avertisment" (English "warning"). The correct tag sequence for this word is: $N_1O_1N_1C_1O_1N_1C_1O_1N_1C_1C_2$. Determining where the junctures appear inside the word is easily done by looking for tag sequences that are unacceptable inside the same syllable such as: C_i–O_j, N_i–N_1, C_i–N_j, N_i–O_j etc. (for whatever indexes i and j). By doing so, we obtain the break sequence: N_1–$O_1N_1C_1$–$O_1N_1C_1$–$O_1N_1C_1C_2$, and with a 1–1 correspondence between tags and letters, we get the sequence "a–ver–tis–ment", which is the correct syllabification of the word.

After iterating through several feature sets we selected the one that yielded the highest results: (l_{-2},l_{-1},l), (l_{-3},l_{-2},l_{-1},l), $(l_{-4},l_{-3},l_{-2},l_{-1},l)$, (l,l_1,l_2), (l,l_1,l_2,l_3), (l,l_1,l_2,l_3,l_4), (l_{-1},l,l_1), $(l_{-2},l_{-1},l,l_1,l_2)$, where l is used to mark the current letter and l_i is used to denote the letter at relative distance i from the current one.

3.7.1.2 Experiments and results

To test this approach, a training corpus consisting of 600K syllabified words, compiled from the Romanian Academy Explanatory Dictionary was used. The accuracy was measured using 10–fold cross validation and the average value was **99.01%** on **OOV** words**.** To the best of the author's knowledge, the best performing system for Romanian syllabification is presented in Ungurean et al. (2011). In their approach, they use Katz–Backoff for determining the most probable n–gram letter split sequence using the output of a stochastic search algorithm. Their method obtained a maximum accuracy of **97.04%** using a window of 5 letter n–grams.

3.7.2 Lemmatization

Lemmatization is the process of determining a word's canonical form from its inflectional form. It is a technique useful in various natural language processing applications such as data–mining and document classification. Lemmatization is related to the technique called stemming, which is the process of extracting the longest common subsequence from all of a word's forms.

In the case of English, the lemmatization process is fairly simple, but for highly inflectional languages such as Romanian this process raises a series of challenges. There are several approaches to this task, with a trend toward rule–based transformations applied to the sequence of characters. The best–performing

Romanian lemmatizer[21] (to the best of the author's knowledge) is implemented after the methodology proposed in Ion (2007). The method builds a lookup table storing for each POS tag (named CTAG) the transformations required for word form to canonical form conversion. When the method has to predict the lemma of a previously unseen word with an associated CTAG (supplied by the POS tagging process), it searches the lookup table for the transformation rules of the CTAG and applies all of them to the unseen word, thus obtaining a set of candidate lemmas from which it probabilistically chooses the most likely one.

3.7.2.1 Lemmatization with MIRA

In order to use the MIRA framework, lemmatization was reformulated as a sequence labeling task. The labels were designed to encode transformations, hence this can be referred to as transform–tagging:

- '*' – means leave the current letter *unchanged*
- '*_nil_*' – means that the current letter must be *removed* from the word's lemma
- '*_r(<character sequence>)* – means that the current letter has to be *replaced* with the character sequence in brackets (*<character sequence>*).

To exemplify, the 2nd person, plural verb "îmbrăcați" (English "dressed") will be used. The canonical form of "îmbrăcaţi" is "îmbrăca" ("*to* dress") and the letter tag sequence is shown in Table 16.

Table 16 – Lemmatization example for word "îmbrăcați"

î	m	b	r	ă	c	a	ț	i
*	*	*	*	*	*	*	_nil_	_nil_

Lemmatization has to take into account the information provided by the word's morpho–syntactic–description (MSD) tag (Ion, 2007). This can be achieved in one of two ways: (1) by training different models for different MSDs or (2) by incorporating the MSD information inside the features that are used. The Romanian MSDs inventory is very large (more than 600 MSDs) and consequently, the MIRA model obtained by training with MSDs is extremely large, difficult to train and use. Tufiş (1999) presents a strategy for coping with the large Romanian MSD inventory, in which he eliminates lexicon–recoverable

[21] http://ws.racai.ro:9191

morpho–syntactic attributes from the MSDs. The resulting tagset is much smaller and the resulting POS tags are called CTAGs.

In order to reduce the lemmatization model size, every word's MSD from our training set was converted into a CTAG, based on the above mentioned methodology. This reduced the model size about 5 times.

The context used by the labeler is composed of both lexical and morpho–syntactic features (CTAGs): (l_{-2},l_{-1},l,C), (l_{-3},l_{-2},l_{-1},l,C), (l_{-4},l_{-3},l_{-2},l_{-1},l,C), (l,l_1,l_2,C), (l,l_1,l_2,l_3,C), (l,l_1,l_2,l_3,l_4,C), (l_{-1},l,l_1,C), (l_{-2},l_{-1},l,l_1,l_2,C), where l is used to mark the current letter, l_i is used to denote the letter at relative distance i from the current one and C is used to denote the word form's CTAG.

3.7.2.2 Experimental results

The tests were performed using a training corpus composed of 1M words from which 10% for each individual CTAG was withheld as the test set. The results of the experiments are shown in Table 17. The overall accuracy of **94%** which is **12%** higher than the results presented in Ion (2007).

In Table 17, all CTAGS beginning with an "N" are nouns, "A" are adjectives and "V" are verbs. The best result (100%) is for invariant adjectives ("A") for which the lemma is the word form. This behavior is preserved for all CTAGs for which lemma is equal to the word form: NSRN (noun, singular, nominative/accusative, non–definite form) with 99.5%, ASN (adjective, singular, non–definite form) with 98.95%, etc. At the opposite pole we find words with CTAGs that are harder to lemmatize: NPN (noun, plural, non–definite form) with 81.51% or NPOY (noun, plural, dative/genitive, definite form) with 83.01% due to their root alternation when going from singular (the number of the lemma) to plural, e.g. for "*stadio*an**elor**" (NPOY, English "of the stadiums") lemma is "*stadion*" (English "stadium") where in bold we have the inflectional ending corresponding to the CTAG NPOY and in italic we have the root of the word.

Table 17 – Lemmatization results

CTAG	# of tokens	# of errors	Accuracy %
A	16	0	100
VN	871	47	94.6
NSON	4223	190	95.5

APOY	5078	99	98.05
NSVN	79	3	96.2
ASN	6205	65	98.95
VPSM	1178	77	93.46
NSOY	6761	279	95.87
ASRY	5121	67	98.69
NP	263	35	86.69
NPRY	6443	884	86.28
VG	2973	118	96.03
NN	263	3	98.86
VPSF	748	15	97.99
APN	6062	127	97.9
NSN	2591	6	99.77
V2	8195	664	91.9
NPOY	6427	1092	83.01
V3	7312	629	91.4
ASON	3030	43	98.58
VPPM	797	58	92.72
NSRY	6701	104	98.45
VPPF	747	15	97.99
V1	6180	455	92.64
APRY	5119	95	98.14
NSRN	4244	19	99.55
ASOY	5122	59	98.85
NPN	6615	1223	81.51
NPVY	28	3	89.29
NSVY	2225	31	98.61
ASVY	626	12	98.08
AN	106	6	94.34
Overall	**112349**	**6523**	**94.19**

3.7.3 Phonetic transcription

Phonetic transcription (PT; also referred to as grapheme–to–phoneme (G2P) or letter–to–sound (L2S) conversion) can be formalized as finding a relation between letters and corresponding phonemes, which is not a straightforward task and may pose some challenges for languages such as English. For Romanian, *phonetic transcription rules are relatively simple* compared to English or French (Burileanu, 1999), but there are several exceptions that need to be considered. For the purpose of language independence, data–driven methods are preferable as they only require words and their phonetic transcription equivalents for training, which are easier to obtain than wide coverage set of phonetic transcription rules.

Several Machine Learning (ML) methods have been proposed for the PT task: Black et al. (1998), Jiampojamarn et al. (2008), Pagel et al. (1998), Bisani and Ney (2002), Marchand and Damper (2000) and Demberg (2007).

Jiampojamarn et al. (2008) presented a MIRA based method for L2S conversion of words. Their best result on the English CMU lexicon was 71%. However, the feature template provided in their paper did not turn out to be suitable in our tests. Instead, a different one was developed, which turned out to be the most discriminative for Romanian G2P conversion: (l_{-2},l_{-1},l), (l_{-3},l_{-2},l_{-1},l), $(l_{-4},l_{-3},l_{-2},l_{-1},l)$, (l,l_1,l_2), (l,l_1,l_2,l_3), (l,l_1,l_2,l_3,l_4), (l_{-1},l,l_1), $(l_{-2},l_{-1},l,l_1,l_2)$, (l_{-2},l_{-1},l,l_1), (l_{-1},l,l_1,l_2), where l is used to mark the current letter, l_i is used to denote the letter at relative distance i from the current one.

All the data–driven methods for phonetic transcription require alignments between letters and phonemes. For so–called phonetic (or pseudo–phonetic) languages (e.g. Romanian), the task of *grapheme to phoneme alignment is significantly easier* and more accurate than for many other languages (such as English). However, there are several issues, common to several languages. The simplest example is that not all words have the same number of phonemes and letters and even if this condition is satisfied, it still does not imply a one–to–one alignment (e.g. experience – IH K S P IH R IY AH N S, where the letter x spawns two phonemes "K" + "S" and the ending "e" is silent; a similar phenomenon happens when we phonetically transcribe the word Romanian "experienţă" (experience) into e k s p e r i e n ts @, where again x spawns "k"+"s"). Expectation–Maximization (EM) can be used to find one–to–one or many–to–many alignments between letters and phonemes (Black et al., 1998; Jiampojamarn et al., 2008; Pagel et al. 1998). Although it is arguable that in the case of Romanian such alignments can be easily obtained using simple heuristics, EM was used on our training data, *to keep the system portable to other languages.*

3.7.3.1 Initial approach to phonetic transcription using a Maximum Entropy classifier

Before the MIRA approach to G2P, there have been several other experiments using (1) a custom designed algorithm called Dictionary Lookup or Probability Smoothing (DLOPS), (2) a Maximum Entropy (MaxEnt) classifier and (3) a cascaded classifier approach for automatic error correction that were thoroughly described in Boroş (2013b). Briefly presenting the two other approaches is important because they provide relevant information regarding the performance of the MIRA approach when applied to Romanian G2P using the same lexicon.

(a) The DLOPS approach used a divide and conquer algorithm for approximating the G2P conversion of a sequence of letters either by doing a dictionary lookup or by approximating the transcription from sub–units (sub–groups of letters contained in the initial sequence). During training, the method creates a look–up table that contains n–grams from 2 to 8 letters associated with possible transcriptions and Maximum–Likelihood Estimation (MLE) probabilities of the transcription. Given an input sequence (S) of letters the algorithm performs a look–up in the table and either directly returns the list of possible transcriptions for known input sequences or compiles a new list by recursively performing the same procedure on smaller overlapping units and merging all possible transcription variants[22] using a smoothing function. The smoothing function and the method for choosing where to split the initial sequence were presented in the original paper.

(b) MaxEnt classifiers have been used in the NLP field to solve problems such as detecting sentence boundaries (Reynar and Ratnaparkhi, 1997; Agarwal,

[22] Because the algorithm is applied on overlapping sequences of letters it is expected that compatible transcription solutions must also be composed by overlapping phonetic sequences. Thus, the algorithm compiles the new list of transcription candidates by performing a Cartesian product on overlapping phonetic sequences of the adjoined segments.

2005), POS tagging (Ratnaparkhi, 1996), text classification (Nigam et al., 1999), etc. The principle of MaxEnt (Berger et al., 1996) is to construct a statistical prediction model from the training data and assume a uniform distribution for the unseen data, thus maximizing the entropy. In the MaxEnt experiment, the classifier was trained to predict the phonetic equivalent of every letter inside a word using features extracted from the grapheme context (combinations of letters) and features extracted from the phonetic context (the phoneme assigned to the previous token) [23].

(c) Noticing systematic errors in the first two approaches a cascaded model was used to train another classifier to automatically correct these errors based on an extended phonetic feature–set. The principle is to perform phonetic transcription in two phases: in the initial phase use any method to produce a phonetic transcription of the input sequence; in the second phase the initial phonetic transcription is used to create an enhanced feature–set. The enhanced feature–set adds a preview of the label of the next letter (that was unavailable during the phase) to the context of the current letter, thus learning to perform error correction on the initial output.

The performance and results of these methods will be further addressed in the next section.

3.7.3.2 Experiments and results

The Romanian training data was extracted from the Romanian Speech Synthesis Corpus (RSS) (Stan et al., 2011) and it is comprised of a small number of words (8K). However, due to the preponderantly phonetic nature of Romanian, this number seems to be sufficient for training a highly accurate G2P data–driven method. Using 10–fold cross validation, an accuracy of **96.29%** was obtained on OOV words, which is comparable to the state–of–the art results (**96.99%**) of a system reported in Ungurean et al. (2011). Additional experiments were made using the CMU Dictionary, NetTalk and the UK BEEP for English, BRULEX for French and CELEX for German and Dutch. Table 18 shows the results obtained using MIRA, DLOPS, MaxEnt and the cascaded model.

[23] Various combinations of features were tested in a trial and error process; the results section will only refer to the best feature-combination.

Table 18 – G2P experiments and results using various lexicons and methods

Method	**CMU dict**	**UK BEEP**	**Net Talk**	**BRULEX**	**CELEX**	**CELEX**	**Roma nian**
DLOPS*	57.00	64.07	53.14	79.17	79.27	78.11	85.74
MaxEnt*	67.22	72.41	68.55	90.99	90.17	90.49	93.29
DLOPS cascaded*	63.60	67.96	59.70	85.79	86.99	84.89	91.81
MaxEnt cascaded*	68.29	**73.56**	**69.19**	91.68	92.25	91.05	93.34
Perceptron	71.03	–	64.87	93.89	95.13	92.84	–
MIRA	**71.99**	–	67.82	**94.51**	**95.32**	**93.61**	**96.29***
CART	57.80	–	–	–	–	89.38	87
1–1 Align	60.30	–	–	87.00	–	86.60	–
1–1+CSIF	62.90	–	–	86.50	–	87.50	–
1–1 HMM	62.10	–	–	88.20	–	87.60	–
M–M Align	65.10	–	–	90.60	–	91.10	–
M–M+HMM	65.60	–	–	90.90	–	91.40	–
MeR+A–star	63.81	–	–	86.71	–	90.63	–

Note: the results of our proposed experiments are marked with '*'

As shown, the MIRA classifier outranks all other methods except the cascaded MaxEnt on the NetTalk lexicon, suggesting that it can reliably and effortless be used to create G2P models for other languages.

3.7.4 Lexical stress prediction

In natural speech certain syllables inside a word have a higher prominence compared to neighboring syllables of the same word. When this phenomenon occurs, it is said that the syllable is carrying lexical stress. Lexical stress prediction is critical in prosody generation for TTS systems as it governs the correct pronunciation of diverse words and it is used to discriminate between homographs.

Oancea and Bădulescu (2003) introduced their rule–based method for lexical stress prediction on Romanian. They extracted rules and tested their method on a lexicon of 4500 words, achieving **94%** accuracy. Ungurean et al. (2009) used Katz's back-off smoothing, for lexical stress assignment based on letter n–grams. Their algorithm works by calculating the probability of every possible combination of stress pattern on an input string. According to their evaluation, this method achieves an accuracy of over **99%** for OOV words.

3.7.4.1 *Lexical stress prediction with MIRA*

The tagging strategy used for stress prediction is also inspired after the numbered ONC style encoding used for syllabification. In this case a numbered tagging strategy was designed, in which: (a) the "BPS" tag was used to label letters which appear before the primary lexical stress; (b) "APS" was used on letters that appear after the primary lexical stress and (c) "PS" to label the letter which carries the primary lexical stress. To exemplify, the labels for the word "îmbrăc**<u>a</u>**ţi" (bolded and underlined *a*, receives the primary lexical stress) is shown in table 19. This type of encoding is available for Romanian, which only uses primary lexical stress. For other languages, which support multiple degrees of lexical stress, the encoding requires adaptations.

Table 19 – Lexical stress tagging for the word "îmbrăcaţi"

î	m	b	r	ă	c	a	ț	i
BPS1	BPS2	BPS3	BPS4	BPS5	BPS6	PS	APS1	APS2

3.7.4.2 *Experiments and results*

Franzén and Horne (1997) conducted a study on stress patterns in Romanian. They showed that stress is more influenced by derivational affixes than by inflectional ones, especially for nouns and verbs. Since the vast majority of derivational affixes change the grammatical category of a word, we were motivated to split our training data into 5 categories: nouns (N), verbs (V), adjectives (A), adverbs (R) and mixed (M). This is where the main difference between our approach and other methods can be seen: *splitting the training data based on the part–of–speech increases the overall accuracy by **3.9%*** (see Table 20).

Table 20 – Lexical stress accuracy

POS	# tokens	# errors	Accuracy
V	11403	42	99.63%
A	11180	55	99.50%
R	52	10	80.77%
N	11060	296	97.32%
Ignored (M)	**33695**	**1718**	**94.90%**
Overall	**33695**	**403**	**98.80%**

When predicting the primary lexical stress position for a given word, a model is chosen based on the POS tag of the given word. If the POS is different from the first four categories or if it is unknown (if there is no context available), the system uses the *mixed model*, which is a model created by training on the entire lexicon regardless of the POS.

The lexical feature templates used for lexical stress prediction are identical to the ones used for lemmatization.

3.8 Foreign words in TTS synthesis[24]

Besides unseen words which were not present in the training data, the OOV class also contains foreign words. Improper handling of such words has a negative impact on the TTS synthesis, as applying the same native phonetic transcription or syllabification rules on them produces undesirable results. There are two ways to handle such words, each requiring them to be first identified as foreign. The first strategy is to apply the above mentioned sub–tasks of syllabification, phonetic transcription and lexical stress prediction using a custom set of rules adapted to the foreign language from which the word originates. However, having different sets of rule packages for languages other than the native language is considered challenging and cumbersome. The second strategy is to use transliteration on these foreign words and to convert them to ***pseudo–native*** words. This facilitates the usage of a single package of native rules/learned models for the tasks of phonetic transcription, syllabification and lexical stress.

The difference between the two proposed methods is that the first applies phonetic transcription with syllabification and lexical stress rules for the foreign language(s) followed by an adaptation at phonetic level between the two languages, while the second method uses transliteration to produce a pseudo–

[24]*Section 3.8 is a close adaptation of the author's paper "Maximum entropy based machine transliteration. Applications and results" presented at Consilr 2013 (Zafiu and Boroş, 2013)*

native word and then uses its native rule sets to reach its final goal (thus not requiring training additional methods for syllabification and lexical stress).

Transliteration between two languages is the process in which the letters of a word in the first (source) language are transformed or mapped into letters that would correspond to a word in the second (target) language. Transliteration was initially introduced in Machine Translation for the task of converting words without a corresponding direct translation (e.g. proper names) between languages that are highly incompatible at phonetic or orthographic levels. For example, a Japanese native speaker cannot distinguish between the English sounds 'L' or 'R'.

Over the years, several methods for transliteration were introduced, mainly focused on languages such as Chinese, Japanese, Korean or Arabic. Knight and Graehl (1997) presented a method for transliteration between Japanese and English, using finite state transducers. This method was later adapted in Stalls and Knight (1998) for bidirectional transliteration between English and Arabic. Other methods for transliteration are presented in Jung et al. (2000), Meng et al. (2001), Virga and Khudanpur (2003). In their work, Haizhou et al. (2004) classifies the above mentioned methods as phonetic approaches to transliteration. They propose a new technique that focuses on direct orthographic mapping (DOM). Their method is also referred to as n–gram based transliteration.

In the proposed approach, a LTS inspired model performs transliteration instead of phonetic transcription from English to Romanian, because many foreign words found in Romanian written texts originate from English. Using 10–fold validation on a subset of 40K transliterated words from the CMUDICT, the measured accuracy was 78% on OOV words. To the author's knowledge, there is no similar study on transliteration between English and Romanian with which to compare the results with.

3.8.1 Detecting which words require transliteration in TTS

One common problem with both approaches to foreign word adaptation for TTS synthesis is detecting when an OOV word is a foreign word and also what is its source language. One partial solution to this problem is to use a lookup table of word–forms for each foreign language for which the system has transliteration rules. Such a list is easier to obtain than a list of fully processed words and it can be done by crawling through documents written in specific languages. Any OOV word found by the TTS system has to be checked against these precompiled lists and once the word occurs in the lexicon of some language it can be transliterated to a native pseudo–word using a specific rule set. It is also important to keep a separate word–form list for the native language as well and to check if the word is not inside that list, as words in different languages may have identical orthographies (e.g. "merge" in Romanian means "walk" but it is also a valid English word). This list is important for determining when not to apply transliteration.

There are however cases where a word or a group of words does not appear in any lexicon, as in the case of uncommon proper nouns. Based on the fact that some orthographic symbols (especially those that have diacritics) or groups of symbols are uncommon in certain languages, the assumption that a word should be transliterated can arise from testing for such occurrences. For example, characters such as 'y' or groups like "ck" are very uncommon for Romanian.

The reasons for using the transliteration method instead of custom designed methods and lexicons are straight forward:

- The resources involved in constructing lexicons and building methods for custom syllabification and phonetic transcription are far more challenging that just building transliteration lexicons.
- The statistical methods used for generating prosody are trained on native words of the language for which the system was designed. When using custom syllabification lexicons for foreign words, usually unseen syllables would show up and decrease overall performance.
- One might argue that applying native syllabification and lexical stress rules on pseudo–words does not generate the same pronunciations as for the word's native language. However, this does not cause major inconveniences, since a non–native speaker could pronounce such foreign words similarly, misplacing the lexical stress and making adaptations at the phonetic level.

3.9 Conclusions

The path from text to speech involves a number of highly complex and difficult sub–tasks that in order to provide the necessary support for high quality speech synthesis must be carefully designed and crafted. As explained, the front–end task of converting the input text into a relevant feature–set for the speech synthesizer requires a number of highly interdependent processes such as diacritic restoration, word–recasing, text normalization, part–of–speech tagging, syllabification, stress prediction, grapheme to phoneme conversion and transliteration of foreign words. All these sub–tasks have been thoroughly addressed and the solutions either adapted from other languages to Romanian or original ones are up to par with the current state–of–the–art methods and techniques.

The work presented in this section only covers the basic pre–processing steps involved in TTS and other aspects related to speech and prosody will be discussed later in chapter 5. Although basic, these are none the less difficult and challenging tasks for TTS synthesis and the rich morphology of Romanian complicates most of the processes by generating data sparseness.

Every component is useful in itself, a fact which is demonstrated later in chapter 7, where some of the methods described in this chapter are salvaged in building the speech translation prototype. Other uses for sub–modules can be easily found: for example (1) the syllabification tool can be exploited by any word processor in word–splitting; (2) spellchecking, diacritic restoration and word re–casing are useful document proofing tools and (3) part–of–speech tagging represents a good educational tool for teaching morphological analysis.

Chapter 4

The naturalness and intelligibility of the synthetic voice is a determining factor for the overall quality of any TTS synthesis system. This chapter offers an overview of today's major approaches to corpora–based speech synthesis methods and introduces the unit–selection baseline system, which was designed for the evaluation of the text pre–processing and analysis front–end that was previously described in chapter 3

Digital Signal Processing for Text–to–speech synthesis

4.1 An introduction to speech synthesis back–ends

As previously mentioned, the conversion from text to speech requires a NLP front–end that takes the input text and converts it into a feature–based representation that is later analyzed and converted into speech by the speech synthesis back–end. The speech synthesis back–end is based on a set of ML and DSP methods that are intended to model the input of the NLP front–end into a collection of attributes specific to signals and speech. Such parameters are usually prosody related and they refer to the pitch of the voice (fundamental frequency (F0)) and the duration of individual phonemes. In some cases the pitch and duration are supplied to the back–end directly by the NLP framework. The DSP processing carried out during TTS synthesis is highly dependent on the *speech synthesis method* that is used to generate the actual waveform.

The first generation of TTS systems relied on *rule–based speech synthesis* and two well–known such speech synthesis methods are (1) *articulatory speech synthesis* and (2) *formant speech synthesis*.

In articulatory synthesis, mathematical or physical models are used to simulate the mechanics and acoustics of the vocal tract and articulation process, involved in the production of human speech. There are two main problematic issues with this method: (1) it relies on a deep understanding of how human speech is produced which is good, but this process is extremely complicated and still lacks some answers and (2) the models involved in this type of speech synthesis are very complex.

Formant synthesis relies on the source–filter model of the production of speech and it explicitly represents voicing, formant resonances and noise, using additive synthesis in order to obtain the output waveform.

Modern speech synthesizers use *corpora based speech synthesis methods* such as concatenative unit–selection speech synthesis and statistical parametric speech synthesis.

4.2 Corpora based speech synthesis methods

Corpora based methods rely on available recorded speech corpora in order to (1) select and use natural speech segments in the synthesis process or (2) learn how to model speech parameters in different contexts.

There are two main approaches to corpora based speech synthesis that have become prevalent in recent years: *unit selection speech synthesis* and *statistical parametric speech synthesis*.

(1) In the *unit selection speech synthesis* the idea is to select (depending on the context) variable sized natural speech units and use them to generate the output waveform through a "concatenation" process.

(2) In the *statistical parametric speech synthesis* spectral and duration patterns based on the context are learned from the available corpora. During synthesis, the learned data is used to generate new sets of parameters, which are then converted into waveforms using various filters.

Each of the two methods has advantages and disadvantages over the other (see table 21 for details) and recently hybrid speech synthesis methods have emerged from the idea of combining the two methods in order to take advantage of their strengths and reduce their weaknesses (see section 6.2.2 for details).

Table 21 – Advantages and disadvantages of unit selection and statistical parametric speech synthesis

Speech synthesis method	Advantages	Disadvantages
Unit selection	(a) In best–case scenarios, unit selection speech synthesis is able to produce natural sounding and pleasant voices	(a) Requires a high effort to record and label the speech corpus; (b) High storage requirement, due to the fact that it relies on large scale speech corpora for selecting optimal speech units; (b) Regardless of the size of the speech corpus, arbitrary text is likely to create contexts with a small number of usable concatenation candidates, in which case the resulting voice suffers from prosody related issues and audio artifacts generated by the spectral mismatch of consequent units.
Statistical parametric	(a) The model footprint is extremely small when compared to unit selection speech synthesis; (b) The quality of the resulting voice is constant regardless of the input text;	(a) The resulting voice loses its naturalness.

4.2.1 Unit–selection speech synthesis

As mentioned earlier, concatenative speech synthesis methods rely on pre–recorded speech samples that are concatenated in order to obtain the output waveform. Their major advantage is the ability to obtain natural and pleasant sounding voices if they have sufficient data.

The first generation of concatenative speech synthesis methods used a fixed inventory of speech units, initially phonemes (highly problematic at synthesis time because of concatenation mismatches), diphones (Black, 1999), syllables etc. To avoid some difficulties in the speech database design, modern concatenative speech synthesis methods use a variable sized inventory of speech segments, hence the name unit selection speech synthesis.

In unit selection speech synthesis speech units vary from phonemes (as a fallback when there is insufficient data available) to diphones, syllables, words etc. During synthesis, an optimal sequence of concatenation units is selected in order to match the target (prosodic and phonetic) requirements and the concatenation context (minimize the spectral mismatch between adjacent units). Prosody is controlled either by (a) selecting units that appeared in a similar context (the feature set provided by the NLP framework) or (b) explicitly modeling target values for pitch and duration and (1) selecting closest units based on the distance between the actual and target parameters or (2) using a concatenation technique such as pitch synchronous overlap–add (PSOLA) in order to control the duration and pitch of the concatenation units and match the target values.

PSOLA is a method used to perform time–scale and pitch modification to the speech signal (Moulines and Charpentier, 1990) that has been widely used in concatenative speech synthesis systems. The basic principle of this method is to express the speech signal as a function of pitch cycles (Huang et al., 2001): $x[n] = \sum_i w_i[n]\, x[n - t_a[i]]$, where $x[n]$ is the speech signal; $t_a[i]$ are the epochs of the signal, spaced so that the distance between two adjacent epochs is the pitch period at that time, expressed in samples; $w_i[n]$ is a window function with has the property $\sum_i w_i\big[n - t_a[i]\big] = 1$ (satisfied by the Hanning Window). The pitch of the original signal is modified by replacing the initial sequence of epochs $t_a[i]$, with another sequence of epochs $t_s[i]$ that is calculated with respect to the target pitch values and the duration is controlled by removing or repeating epochs (see figure 6). One of the main disadvantages of PSOLA is that it requires a correct pre–estimation of the pitch epochs and automatic detection of these pitch epochs starting from recordings is not sufficiently accurate.

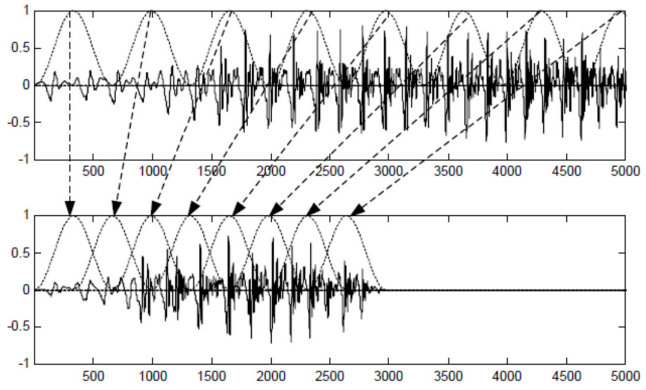


Figure 5 – PSOLA pitch and duration modification (Huang et al., 2001)

4.2.2 Statistical parametric speech synthesis

In this type of speech synthesis, the speech signal is represented using a set of parameters that actually represent the spectral envelope of the speech signal, F0 and aperiodic excitation (King, 2010). During training, a statistical model is created, in which statistical properties such as mean and variance of these parameters is modeled over time based on an input feature–set, which is usually generated using the text pre–processing and analysis component of the TTS system. During the synthesis process, the previously created model is used to generate the speech parameters of unseen data using the input of the NLP framework. These parameters are later converted into speech using a *vocoder*. There are several proposed vocoding methods, among which the most widely used because of its ability to obtain good results is the Speech Transformation and Representation using Adaptive Interpolation of weiGHTed spectrum (STRAIGHT) (Kawahara et al., 1999).

A common choice for modeling these speech parameters is the Hidden Markov–Model (HMM), which was initially designed for speech recognition. When the model is applied to speech synthesis, each model state generates a set of speech parameters used in the synthesis process. The prosodic property of duration can be implicitly modeled as the sum of state durations taking into account the self–state transitions, but in practice this model is not adequate for high quality speech synthesis. Thus, duration is explicitly modeled by adding a duration model to the HMM states and creating what is known as Hidden Semi–Markov–Model (HSMM).

4.2.3 Hybrid approaches

Unit selection and statistical parametric speech synthesis both have their strong and weak points that were presented in section 4.2. Thus, recent studies have focused on the possibility of joining the two speech synthesis approaches into what is known as the hybrid speech synthesis, resulting in different approaches such as using HMM speech synthesis to provide better target cost calculation for the unit–selection speech synthesis (Kawai et al., 2004; Rouibia and Rosec, 2005; Lu et al., 2009; Qian et al., 2010) and combining natural speech units with synthetic speech units in order to reduce the discontinuities generated by the spectral mismatch at joint points (Pollet and Breen, 2008; Tiomkin et al., 2010; Guner and Demiroglu, 2012).

4.3 The unit–selection speech synthesis baseline system

The lack of freely available resources and tools has hardened the development and evaluation of corpora based methods for speech synthesis for Romanian. The release of the Romanian Speech Synthesis (RSS) database (Stan et al., 2011) has enabled research in this direction and has allowed the testing and implementation of a base–line unit–selection speech synthesizer that was designed to work with the linguistic analysis provided by the methods and techniques presented in chapter 3 in order to provide a better overview on how these methods work when integrated in a TTS synthesis system. The speech synthesis component uses a Viterbi–style search algorithm used to select the optimal unit sequence based on a joint cost function (equations 16, 17 and 18).

$$S = \alpha T_{cost} + \beta C_{cost} \tag{16}$$

$$T_{cost} = |t_f - f| w_f + |t_d - d| w_d \tag{17}$$

$$C_{cost} = \sum_i \Delta MFCC_i + (\Delta F_0)^2 \tag{18}$$

,where

S	–	The unit cost
T_{cost}	–	The target cost
C_{cost}	–	The concatenation cost
t_f	–	Target F0
f	–	F0 value of the unit measured as an average of 6 frames
t_d	–	Target duration

d	–	Actual duration
$\Delta MFCC_i$	–	Difference of the i–th component between the MFCC feature vectors of two adjoined units
ΔF_0	–	Difference between the average F0 of two adjoined units measured over a 6 frames
α, β, w_f, w_d	–	Externally defined weights

In order to avoid the unwanted effect of pitch and duration modification of the speech signal, a decision threshold is used to select if the units will be modified or not by the PSOLA algorithm to match the desired values for pitch and duration. If the difference between the target pitch and duration and the actual values for these parameters are smaller than that threshold, the units are left unchanged.

4.4 Conclusions

The quality of speech synthesis is an important factor in any TTS synthesis system, a fact proved by the high interest shown toward refining various speech synthesis methods. The naturalness and intelligibility of the synthetic voice is highly dependent on the speech synthesis method that is used.

Currently, corpora–based methods are preferable to rule–based and the three main classes of data–driven speech synthesizers have been presented in this chapter. Each method has its own advantages and disadvantages and preferring one method over the other is influenced by multiple factors such as the size and quality of the corpora or the exact application in which the TTS system will be used. Although hybrid systems have shown promising results regardless of the corpora they were trained on and statistical parametric speech synthesis offers a good trade–off between overall stability and naturalness, the quality of unit–selection speech synthesis is still un–matched in certain scenarios.

In order to obtain a baseline TTS system a highly configurable unit–selection speech synthesizer that allows the external control off all the parameters used in the cost function was described in this chapter and future research efforts will be focused on hybrid speech synthesis.

A fair comparison between all the speech synthesis methods (unit selection, statistical parametric and hybrid) is presented in chapter 6, which also presents an evaluation of the unit–selection synthesis system implemented during the preparation of this thesis.

Chapter 5

Generating correct prosody is a delicate and complex area of study which, at this point, only relies on surface clues to what the voice from the text should sound like. Obviously, this is not sufficient because the correct way of "saying" things (e.g., proper intonation, pauses, relative phoneme durations and word contrast) in real–life scenarios requires a deeper understanding of the underlying message and although some commercial systems for TTS do obtain surprisingly good results when dealing with arbitrary text, this is more likely due to hand–written rules combined with the effect of using a large scale recorded speech corpus, while performing very little digital signal processing[25]*.In this chapter we approach the issue of prosody generation from text, describing various prosody annotation system, introducing the newly created RSS–ToBI corpus and offering an evaluation of this corpus by using a crowd–sourcing approach*

Speech and prosody

5.1 What is prosody

While there is no general agreement on description or representation systems for prosody, it is common practice to perceive prosody as a secondary communication channel next to the verbal channel. According to Huang et al. (2001), from the listener's point of view, prosody consists in the systematic perception and recovery of the speaker's intentions based on pauses, pitch, relative duration and loudness of the voice.

Before we proceed, we will briefly introduce the idea of prosody and the basic acoustic realization of this notion reflected in pitch, duration, pauses and loudness of the voice.

[25] For example, one of the current leading TTS systems named IVONA uses a unit-selection algorithm with limited time-scale modifications (Kaszczuk and Osowski, 2009) producing remarkably good results in terms of naturalness and intelligibility of the synthetic voice.

Pitch is used to encode the speaker's sentiments or to draw attention to certain aspects. In some cases, pitch is influenced by language specific rules such as the sense of a word, depending directly on the F0 contour.

In spoken language, words are continuously pronounced and there is no recognizable break between them, unless certain conditions are met. In fact, one sub–task of speech recognition is determining the word boundaries after a sequence of phonemes was recognized. It is well known that under certain circumstances, speakers insert ***pauses*** of different lengths between certain words in a phrase, and, a speech synthesis system has to be able to pinpoint the location of these pauses when generating speech from text. However, their duration is of secondary importance to the important aspect of a TTS system which must not introduce pauses in wrong positions. Misusing pauses can even result in giving different meanings to utterances:

"nu e aici" → he's not here

"nu <pause> e aici" → no, he is here

Punctuation marks are good clues towards introducing pauses in the synthetized voice, but there are a lot of cases where a deeper analysis is required for introducing breaks where there is no punctuation present.

Another aspect of prosody is the ***phoneme relative duration.*** There are several clues indicating that the duration of phonemes is linked to the pitch of the voice, but the relationship is not straight forward and practice shows that for pragmatic reasons, pitch and duration should be treated independently (Wang, 1999). Plumpe and Meredith (1999) convey that a limited context analysis is sufficient for obtaining acceptable results in relative duration prediction.

Both tone and intonation reflect a change in the speaker's pitch over time. The main difference between them is that tones are used to distinguish between words, while intonation only expresses message related information such as contrast, phrase type etc. Generally speaking, there are four types of intonation: rising, falling, dipping (falling followed by rising) and peaking (rising followed by falling) intonation.

While there are certain patterns that can be observed from sentences spoken out of the context (isolated phrases) and a set of rules can accomplish what is today's minimal requirement for TTS naturalness, the performance of prosody prediction from text is far from the level of a normal human reader.

Aspects regarding relative phoneme durations, primary lexical stress and pitch accents, based on lexical and syntactic clues are only one side of the problem. Limited context analysis offers a good starting point in prosody generation, but to fully exploit the role of prosody in message encoding and decoding a better grasp on the "encoded information" is essential. A good human speaker is able to empathize with both the author and the listener. He is able to draw attention to certain aspects of the message by employing pauses, playing with word prominence and simulating affective prosody. From the point of view of any human listener there is a strong difference between the sentences:

"I was there."

"I **WAS** there."

The contrast of the word **"WAS"** compared to the surrounding words in the second sentence dictates that the sentence is a continuation of a communication act between two speakers and that the statement comes in conflict with the suggestion of the previously unseen sentence (e.g. "You were not there!").

Obtaining clues for prosody generation is a complex area of study and prosodic elements are hard to predict using superficial text–processing techniques. Also, given an utterance, two speakers may choose to employ different patterns and even the same speaker is likely to change his preference. Another aspect is that if the prosodic channel becomes redundant, based on the verbal component, it is sometimes neglected. Even language specific rules such as those that decree the phrase type based on acoustic cues are sometimes ignored if the verbal context provides enough information for the listener to fully understand the message. An effective example is given in Taylor (2009) where the sentence:

"Is this the way to the bank?"

does not receive a final tone specific to interrogative sentences because it's exact intent is clear enough without this information (the speaker wants to **know** if this is the way to the market), as opposed to:

"This is the way to the bank?"

which will always receive an interrogative specific pattern because it can easily be mistaken for a declarative sentence.

5.2 Prosody annotation systems

There are several theories supporting the idea of the existence of a prosodic hierarchy inside the utterance (Liberman and Prince 1977; Selkirk, 1984; Beckman and Pierrehumbert 1986; Nespor and Vogel 1983; Ladd 1996) and a couple of systems proposed for prosody annotation (the International Transcription System for Intonation, the TILT intonation model or the Tones and Break Indices). Within the hierarchy models, a hierarchical tree is used to encode relative strengths of composing units of the utterance, but each theory supports different levels and depths of the tree.

One of the prosody annotation systems is the INternational Transcription System for INTonation (***INTSINT***), which is perceived as the prosodic equivalent of the International Phonetic Alphabet (IPA) (Handbook IPA, 1999). It was developed and introduced in Hirst (2000). The intonation is coded using 8 symbols (T – top, H – higher, U – up stepped, S – same, M – mid, D – down stepped, L – lower, B – bottom). The 8 symbols are followed by a set of 5 other symbols that are used to encode their timing (‘[’ → initial, ‘<’ → early, ‘:’ → medial, ‘>’ → late, ‘]’ → final)

The ***TILT*** intonation model was introduced in Taylor (1998) to describe boundary tones and pitch accents using a unified set of parameters and to facilitate automatic intonation processing and generation. The parameters used in the *tilt* model are:

- Amplitude (the size of the F0 excursion);
- Duration;
- Tilt (a parameter used to describe F0 movement; –1 indicates a pure fall, +1 a pure rise, and any other value indicates a combination of rises and falls with 0 indicating an equal number of movements in the two directions);
- F0 Position (distance of F0 from the baseline in the middle of the event);
- Time position (absolute or relative time from the beginning of the utterance or the middle of the event).

Tones and Breaks Indices (***ToBI***) (Silverman, 1992) is one of the widely accepted standards for prosodic annotation. It was originally designed to accommodate the prosodic phenomena specific to American English, but adaptations have been made for other languages, as is the case of the J–ToBI standard for Japanese (Campbell and Venditti, 1995) or the RoToBI standard for Romanian (Jitcă et al., 2012). The ToBI specific break indices (break tear) formalize breaks in five classes from 0 to 4. 0 represents the weakest break and 4 the strongest:

- 0 clitic boundary;
- 1 normal word–word boundary;
- 2 either perceived disjuncture with no intonation effect, or apparent intonational boundary but no slowing or other break cues;
- 3 intermediate phrase. Gets phrase accent, but not terminal tone;
- 4 full intonation phrase.

The tone tier of the ToBI standard includes a set of symbols for pitch accent tones and final boundary for intermediate phrases and intonational phrases. The pitch accent tones are defined as: H*, L*, L+H*, H+!H*, !H*, L+!H*, L*+!H*; the final boundary tones for intermediate phrases are: L–, H– and the final boundary tones for intonational phrases are: L%, H%. The symbols H and L stand for High and Low and a detailed description with usage examples is presented in Silverman et al. (1992) and Beckman and Hirschberg (1994) for English and Jitcă et al. (2012) for Romanian.

5.3 Building a prosody annotated Romanian corpus

As previously presented, corpora based methods have become prevalent over rule–based approaches in TTS synthesis because they provide high adaptability and language independence. On major drawback is that such methods require large–sized corpora for training and, as explained, such corpora are not easily attainable for Romanian in comparison to other international circulation languages such as English. For the later mentioned language is has been shown that, given the availability of a prosodically enhanced corpus, it is possible to successfully train ML methods to perform automatic labeling on previously unseen data (Lee and Oh, 1999; Xipeng and Bo, 2000; Sun and Applebaum, 2001; etc.).

The statistical parametric speech synthesis approach could, in theory, greatly benefit from the existence of prosody annotated corpora. In practice, most systems use rule–based algorithms in their front–ends in order to automatically add a simple layer of symbolic prosodic annotations, which is likely to be ignored in the context–clustering performed in statistical parametric speech synthesis since it does not reflect the actual parameters of speech and it does not help in the modeling of these parameters. Therefore it is preferable to have correctly annotated speech corpus with prosodic events.

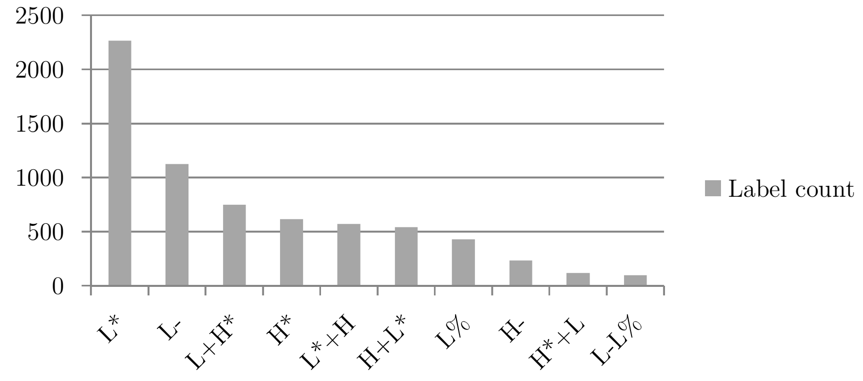
Because the ToBI annotation standard received a significant attention from other research groups, with a focus of extending the standard to suit the Romanian language (the Ro–ToBI standard) (Jitcă et al., 2012) and with the release of the RSS corpus, a decision was taken to annotate a section of the RSS corpus with Ro–ToBI labels (the RSS–ToBI corpus).

The RSS–ToBI corpus was created using the prompts available from the fairytale section of RSS: 1000 sentences totaling 67 minutes of speech. The prompts were pre–processed using the methodology proposed in chapter 3, in order to add typical local–context information used in TTS synthesis such as phonetic transcription, syllabification, stress prediction and part–of–speech (POS) tags.

The prosodic layer was built in two stages. In the first stage a number of 5 volunteers were asked to listen and label the speech corpus, with the help of a custom designed visualization and editing tool. Each of the annotators tagged the entire corpus. The initial inter–annotator agreement rate was below 40%; this is justifiable as the number of Ro–ToBI tags is large, there are tags that are very similar, and the annotators gave preference to one or to a very similar another (e.g. H* and L+H*). The second stage was needed because of the low annotator agreement level: a single speech expert went through the entire speech corpus, and, based on the primary annotations, he manually edited and resolved existing conflicts.

When grouping together both pitch accents and boundary tones, the corpus contains a total number of 7022 labels (32 unique) (see figure 6 for highest occurring labels).

Figure 6 – Highest occurring ToBI labels inside the RSS–ToBI corpus



5.4 Evaluation of the Romanian prosody annotated corpus

This section describes the tests performed to assess if using the ToBI annotated corpus as opposed to the same un–annotated corpus has a beneficial effect on a TTS system.

Two statistical parametric speech synthesis models were built using the ToBI labeled (system A) and the un–labeled (system B) versions of the RSS corpus. A number of 37 sentences were randomly chosen and manually labeled from a previously unseen test set consisting of 19 news and 18 novel sentences. The test sentences were synthesized using both models and an anonymous preference test was conducted on a purpose–built website[26]. In the preference test, anonymous volunteers were presented with speech samples consisting from both systems, and, for every individual sentence they were asked to select from a lists of 5 preference options: (1) the systems sound identical, (2) system A sounds a little better than system B, (3) system A sounds much better than system B, (4) System B sounds a little better than system A and (5) system B sounds much better than system A. The participants were asked to carefully consider the

[26] http://rslp.racai.ro/index.php?page=experiment/listening

prosodic aspect of the synthetic voices and to try not to be influenced by the naturalness of the output.

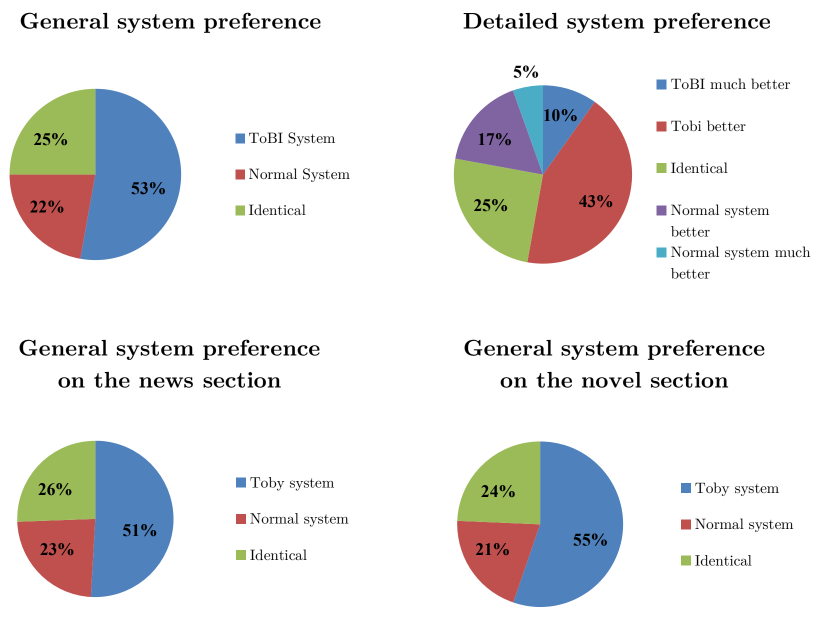


Figure 7 – Preference test results for the two systems

The evaluation campaign is still ongoing at the time of writing, with 587 collected answers – a sufficient number of results on which to draw a series of conclusions. In 52.81% cases, the RSS–ToBI labeled system was considered better than the unlabeled system, in 25.04% of the cases the systems were considered of equal quality and in 22.15% cases, the unlabeled system was considered better than the labeled one (see figure 7 for detailed results). It is important to note that the test respondents are not speech experts. Also, in almost all cases, respondents assigned identical scores to the first set of sentences they listened to, and as the test progressed they started telling the difference between the synthesized sentences; as the respondents were not made to re–listen their initial sentences, we speculate that the preference of RSS–ToBI is even higher than the currently reported figures, given the current preference distribution.

Statistically, for a confidence level of 95% with a confidence interval of 5, we only needed a sample size of 377 answers. Currently, for our 587 answers, with the worst–case 50% response distribution and a confidence level of 95%, we can be certain of the test's results with a confidence interval of 3.99%. This interval is sufficient to statistically prove that the ToBI labeled system is better.

5.5 Conclusions

The newly created speech corpus is a valuable asset to the Romanian language processing as it provides the means to train and test methods for automatically generating prosody form text. By analyzing the preference test results one can see that in more than 50% of the cases the RSS–ToBI labeled system is preferred over the unlabeled one, while the unlabeled system is only preferred in about 20% of the cases. In the cases where most test respondents have marked the systems of equal quality, by performing a subjective listening test, we noted that there are certain subtle differences that make the ToBI system seem slightly better.

The fact that the ToBI system consistently obtained better scores on both sections (news and novel) shows that it is possible to train and test a statistical parametric speech synthesis on different genres, provided that the prosodic annotations are performed according to the output needs.

The corpus, as well as the other resources and tools needed to conduct a similar experiment are freely available for research purposes either through the META–SHARE platform[27], the Romanian TTS platform[28] or by contacting the authors.

[27] http://ws.racai.ro:9191
[28] http://romaniantts.com/new/rssdb/rssdb.php

Chapter 6

During the preparation of this thesis all the proposed methods and techniques for text pre–processing and analysis for Romanian text to speech synthesis were integrated into an internally developed TTS system, referred to as RACAI TTS, which was also e deliverable of the METANET4U European Project. This chapter presents a thorough evaluation of the system that was conducted during an international TTS evaluation campaign

Evaluation of the RACAI text–to–speech synthesis system

6.1 Evaluation methodologies for text–to–speech synthesis systems

The evaluation of TTS systems has always been considered an interesting research topic, mainly because having an objective evaluation methodology would greatly help improving the TTS process of converting the input text and offering the ability to easily test and asses the performance of various associated methods and techniques. For these various methods involved in the text's pre–processing and analysis, an objective evaluation is performed by defining the desired output and measuring the accuracy of the modules in terms of word accuracy rates, phoneme error rates, etc. In fact, this type of evaluation was used throughout chapter 3 by measuring the performance of individual methods on OOV words using 10–fold cross validation.

However, the evaluation of the speech synthesizer or of the TTS system as a black–box does not share the same exact quantifiability and the most obvious reason for this, assuming that it would be possible to compare two speech samples and evaluate their similarity analogous to how a human would speak, it is impossible to strictly define a desired output. Experience has shown that the same speaker is unable to reproduce an utterance twice even if he consistently tries to, and on the other side, if he pronounces the same sentence in two sensibly different ways, they would still be considered correct by listeners. Thus, it becomes practically impossible to define a standard desired output and measure the performance of the TTS system using this method. As an objective evaluation is not possible, this challenge is currently overcome through the use of subjective evaluation methodologies. These methodologies are usually designed to evaluate two major defining properties of TTS systems: naturalness and intelligibility, and are carried out through listening tests using human–expert participants.

6.2 The Blizzard Challenge[29]

Blizzard Challenge (Black and Tokuda, 2005) is an evaluation campaign designed to provide the tools and resources for evaluating and understanding how different speech synthesis methods and techniques work on the same training data. It is primarily designed for evaluating corpora based methods such as concatenative, statistical parametric and hybrid systems. During *the preparation phase* all participants receive the same speech corpora and they can use any lexicon to train their text pre–processing and analysis components. The challenge covers multiple languages for which the organisers can collect and provide the necessary resources and requires participants to synthesize and submit a number of previously unseen sentences. The tests set is traditionally composed of a number of semantically unpredictable sentences (SUS) – designed to assess **the intelligibility** of the TTS system and sentences belonging to different domains such as news or novels – designed to assess **the naturalness** of the synthetic voice.

The evaluation of the TTS systems is carried out with the help of listening groups than can be (1) volunteers with no background of working with TTS systems, (2) speech experts and (3) paid listeners. During *the evaluation phase*, the assessors are asked to listen to sentences randomly chosen from the participating systems and answer various questions. At the end of the listening tests they have to fill in a questionnaire in order to complete the evaluation process.

[29] Note: Section 6.2 contains relevant sections extracted and adapted from the author's paper "The RACAI Text–to–Speech Synthesis System" (Boroş et al., 2013) presented at the Blizzard Challenge Workshop in 2013.

The evaluation of the intelligibility of the systems is simply carried out by asking the evaluators to listen to a sentence (from the SUS group) and write what they hear, thus being able to measure the word–error–rate (WER) based on their input and the original SUS.

Measuring the naturalness of the systems is two–fold: (a) by listening to randomly selected sentences, the evaluators are asked to select an overall evaluation score (from 1 to 5) and (b) to perform a breakdown of their evaluation and select individual scores for (1) the naturalness of the voice, (2) the listening effort involved, (3) their impression of the intonation and (4) the placement of pauses inside the utterance.

Additionally to the typical intelligibility and naturalness tests, Blizzard Challenge also tries to evaluate how different speech synthesis methods are able to reproduce the voice of the original speaker (an important task in the statistical parametric approach). This is done using an additional listening test, in which the evaluators are asked to listen to a sample from the original speaker and rate how closely does the synthetic voice come to this sample by choosing a score from 1 (sounds like a totally different person) to 5 (it is the same person).

When continuously listening to speech samples belonging to different systems, it is likely that some examples will be very good and some very poor. When being presented with a number of good speech samples in a row, the listener increases his or hers expectations and starts to penalize any irregularity in the synthetic voices, thus creating what is known as *the flooring effect* for the other systems. The opposite phenomenon is called *the ceiling effect* and is caused by listening to very poor speech samples followed by slightly better speech samples. This makes the evaluation highly dependent of the order in which the samples are played to the listeners. There are two solutions for this problem: (1) randomly change the order of the systems and (2) during the listening tests, inserted at various points specially chosen speech samples that either sound completely unnatural or are extracted from the original speaker in order to reduce the ceiling and flooring effects.

6.2.1 Preparation phase

In this year's Blizzard Challenge (2013) there were two tracks to which the participants could attend: the English (EH) and the Indian (IH) track.

The English track was divided among multiple sub–tracks: the EH1 track which was to build a voice using 19 hours of sentence segmented speech from audiobooks and the EH2 track which enabled participants to build voice using an extended database of more than 300 hours of un–segmented audiobook data (the voice in both tracks belonged to an American actress named Catherine Byers).

The author's main target of participating in the Blizzard Challenge was to assess the performance of the methods and techniques that were described in the previous chapters and to see how they adapt to languages other than Romanian. In the case of English there are several well–established methods and techniques and an abundance of freely–available resources that can be exploited in training and building models, thus making the participation in the EH track an ideal candidate for evaluation purposes.

Several steps were taken in order to adapt the models to English:

(1) The POS tagger was trained using the morpho–syntactic–descriptions (MSD) tagset which is fully described in the MULTEXT EAST specifications (Erjavec, 2004). We used Orwell's "1984" corpus for training our models, which is a MSD tagged corpus with translations available for multiple languages.
(2) The G2P model was trained using the CMUDict (Weide, 2005) lexicon from which we manually edited a few entries and we filtered out all non–English words using Princeton WordNet (PWN) (Fellbaum, 2010).
(3) The syllabification model was trained using Webster's Pocket Dictionary (Amsler, 2010) which was automatically pre–processed according to the numbered ONC procedure.

The biggest challenge in this year's competition was the speech corpora preparation, which according to the organizers contained a number of imperfect prompts that created problems in the automatic segmentation process. Unfortunately we did not have enough time to manually check and correct the corpus and, instead, we used a simple statistical method, which, based on the duration of individual phonemes and their spectral characteristics determined by HTK, attempted to remove incorrect segments. However, the process was not precise and we were forced to use a very low rejection threshold which caused us to waste ~30% of the corpus. Also, some errors still made it through this filtering process and though we detected them while we were synthesizing the sentences for the EH2 track, we did not manually correct these errors because it would have been unfair for the evaluation process.

HTK (Young et al., 2002) was the only external tool used in our voice creation process upon which we depended to:

- Align speech spans to phonemes for each utterance of the corpus;
- Insert short pauses (‘`sp`’) in the utterances that will further refine speech to phoneme alignment;
- Filter out utterances for which the speech to phoneme aligner could not find a probable–enough alignment, mainly due to the fact that the utterance and its prompt were slightly different.

In order to achieve the refined alignments, we performed the following steps:

- Generate (with ‘`HDMan`’) the phonetic transcription dictionary for all words of the corpus using an enriched version of the CMU Dictionary. The OOV words found in the speech corpus were automatically transcribed using the three different G2P approaches that were previously presented in chapter 3: the MIRA approach, the MaxEnt approach and the DLOPS approach. All alternative conversions generated were added to the lexicon and we later relied on `HVite` to choose the most probable one;
- Generate an initial phonetic transcription of the speech corpus (with ‘`HLEd`’) using the first available pronunciation from the dictionary for every word of the corpus;
- Scanning the phonetically–transcribed corpus from the previous step, generate initial 3–state, left–right with no skips HMM models (‘monophone HMMs’ in HTK terminology) for all phonemes in the inventory (not including the short pause ‘`sp`’ “phoneme”) (with ‘`HERest`’) and re–estimated the initial models 4 times. The pruning thresholds specified with the ‘`-t`’ switch of `HERest` were ‘`250.0 150.0 1000.0`’;
- Added the short pause ‘`sp`’ HMM model (initially copied from the silence ‘`sil`’ model at the start/end of corpus utterances) and re–estimated all HMM models another 4 times using the same parameters of `HERest`;
- Re–generate the phonetic transcription (including generation of short pauses) of the speech corpus (with ‘`HVite`’) using the best HMM models from the previous step in order to obtain the pronunciations that best

match the acoustic data (in case that a word has multiple pronunciations in the dictionary);

- Finally, re–estimating (4 times) the monophone HMMs including short pause with the new corpus transcription and generate the alignments with `HVite` giving it the option to output the start/end times for every monophone.

6.2.2 Results

The parameter set used in the unit selection process was chosen so that continuous segments would be preferred over those that met the prosody requirements. Tweaking the parameter set was a subjective process and none of the members of the RACAI Team are native English speakers. Figure 7 shows the Mean Opinion Score (MOS) for the similarity with the original speaker (actual value 2.4) calculated using all listeners and all data. Figures 8 and 9 contain the naturalness results for the news (2.5) and novel (2.3) sections calculated using the scores from all the users.

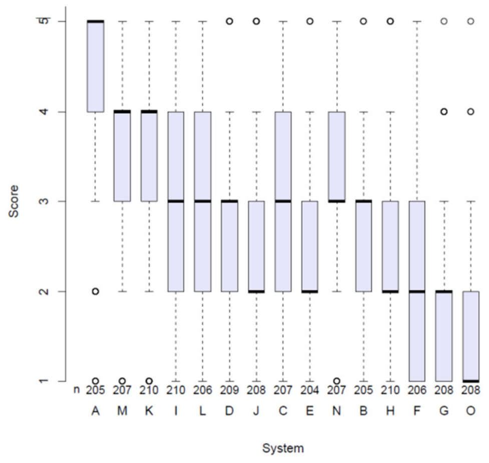


Figure 8 – RACAI TTS (system J) results (similarity to the original speaker – all listeners/all data)

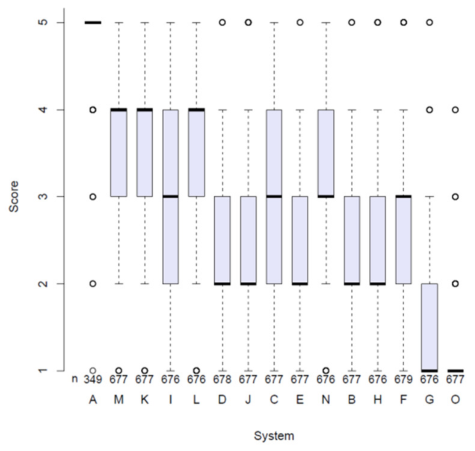


Figure 9 – RACAI TTS (system J) results (naturalness – all listeners/novel)

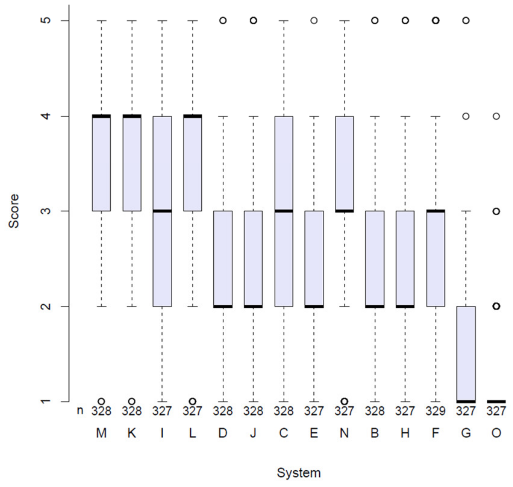


Figure 10 – RACAI TTS (system J) results (naturalness – all listeners/news)

The RACAI entry obtained a high Word Error Rate (WER) of 46%, which was not an unexpected result since in our parameter tweaking process we favored the naturalness and similarity with the original speaker tests. As mentioned earlier, the system supports a decision threshold for the prosodic modification of the selected units. Lowering this parameter increased the synthesis quality for the Semantically Unpredictable Sentences (SUS), but it resulted in an unnatural sounding voice, which was an undesired effect for the other tests in the challenge.

6.3 Conclusions

The participation in the Blizzard Challenge 2013 TTS evaluation campaign showed that the proposed data–driven methods and techniques are scalable and adaptable to other languages. The RACAI entry obtained a MOS score similar to all other unit selection speech synthesis systems with no statistically relevant differences. A general conclusion that was derived from the overall results of all the participating systems, and this is a rephrase from the organizers, is that statistical parametric and unit–selection speech synthesis systems can be as equally good and as equally bad, depending on the feature sets and linguistic analysis provided by the NLP framework but also on other implementation details related to the speech synthesizer. The fact that the RACAI entry placed among the other systems shows that the methods proposed by the author are comparable to state–of–art systems for both English and Romanian. The best performing systems in the Blizzard Challenge 2013 used hybrid speech synthesis methods, proving that by combining natural units with synthetic units one can achieve highly natural and expressive speech and making this a priority for the future development of the RACAI TTS system.

Chapter 7

Applications of text–to–speech synthesis

7.1 An overview of TTS applications

Limited domain speech synthesis has been exploited in numerous applications over the years and one obvious example that almost everyone has come across is the speech synthesis used in the widely–spread interactive voice response systems (IVRs). These systems usually guide the user to navigate through well–defined trees of decisions, a process in which he is presented with a limited number of options at each step (through the use of voice), moving on to the next node incrementally (ex: phone operators' support lines). Another well–established application for this type of speech synthesis is the assistive voice present in the majority of today's navigation systems, a voice which has the important role of giving directions to the user without requiring his full attention on the display screen, thus allowing him to focus on driving. Several other such applications of limited domain speech synthesis exist: the talking clock, voice announcers in public places, busses or subways, etc.

The ability to convert written text into spoken language unlocks the path to a variety of applications that are intended to provide accessibility, assistance or entertainment. In what follows we will briefly address some of them and give a detailed presentation of a prototype speech translation system developed using the previously introduced tools and resources for TTS synthesis:

(1) Probably one of the most common and useful application of TTS synthesis systems is to provide accessibility for the visually impaired or dyslexic people. The first commercial application to provide this type of functionality was the Kurzweil reading machine (Kleiner, 1977). This machine was an integrated system in which an optical scanner was used to digitize a page; optical character recognition (OCR) software was to convert the image into text and a rule–based

speech synthesis system to convert text into speech. This system was an impressive achievement in itself and it is interesting to mention some of the technical details such as the fact that the G2P module used more than 1000 phonetic rules which were used to convert unknown word roots into phonemes and that the system performed a simple type of syntactic analysis in order to add prosodic information to the utterances. Most of today's computerized systems from desktop computers to smart–phones or tablets include such accessibility applications, being able to convert images into speech or provide voice interactive interfaces;

(2) A similar application of TTS is to provide communication means for deaf persons or for any other reason for which one cannot learn of have speaking difficulties. By using TTS systems these people are able to communicate with other individuals that do not understand sign language for example, or it can be used to transmit messages over telephone lines. It must be mentioned that there is a larger class of augmentative communication technologies that are called Speech Generating Devices (SGDs) which use multiple input methods (not just text) in their voice generating process;

(3) Another application of TTS synthesis is in educational software. TTS systems are used to help children learn to read, to improve spelling and pronunciation or to help people learn a foreign language without the supervision of a teacher. Additionally TTS synthesis systems can be used together with ASR and NLP techniques in order to provide a student–computer interface similar to a real teacher. Such a project is DeepTutor (Rus et al., 2012) which uses TTS for presenting lessons and is able to provide an interactive interview–style exam in which speech synthesis is used for asking questions and clarifying ambiguities, while ASR and NLP are used to enable the student to answer questions using natural spoken language;

(4) TTS synthesis finds its usage in any human–computer interaction applications. Today's technological and scientific advances has enabled TTS synthesis to replace limited domain speech synthesis in almost any device: some navigation systems currently use TTS synthesis in order to be able to provide advanced information regarding the route to follow, to speak any address name in conjunction with standard driving directions (e.g.: "take the second exit to Schaffhausen 21913", as opposed to just simple directions

(like ”take the second exit”); any mobile operating system has its own incorporated TTS system and it is able to provide functionalities such as e–book and e–mail reading or HCI interfaces; smart houses and buildings are growing even “smarter” – new TTS modules are designed to provide voice feedback to the user’s commands and in some cases they are integrated with the security system, being able to make phone calls in emergency cases and notify users and/or authorities about occurring or possible future hazards by voice.

Having presented some of the well–known application of TTS, the next section will elaborate around a multi–disciplinary application, which involves methods and techniques used in ASR, MT and TTS, namely speech to speech translation. The work was already introduced in the author’s paper “The RACAI Speech Translation System. Challenges of Morphologically Rich Languages” (Dan Tufiş, Tiberiu Boroş, Ștefan Daniel Dumitrescu) and the following description will focus on the challenges and their solutions involved building a speech translation prototype system designed for the Romanian–English bi–directional language pair.

7.2 Building a speech–to–speech translation prototype

Recent technological advances leading to the popularization and wide–spread of micro–devices with a computational power considerably higher than that of the early 2000’s personal computers accompanied by the standardization of Internet access amplified the need for multimodal and multilingual assistive technologies. A while back, different research groups envisaged the idea of integrating ASR with MT and TTS in order to create what is commonly referred to as speech to speech translation. Through such a system, a text spoken in one language is automatically recognized, translated and synthesized in another language. Back then, there were two main bottlenecks in integrating these technologies: (1) technological – the resources and computational power required by ASR, TTS and MT were prohibitively expensive for mobile devices then, and (2) during the development of the first prototype speech translation system (around 1980’s) (Enkvist, 1982), the state of ASR, TTS and MT was not as advanced as it is now.

Currently there are several projects and systems designed for S2S translation, such as Speechalator (Waibel et al., 2003) or JANUS–III (Lavie et al., 1997), but they mainly centered on English to Arabic, Japanese and Chinese language pairs.

When work was started on the Romanian–English speech translation prototype, further referred to as RACAI S2S, two of the main components had been already developed and thoroughly tested in multiple evaluation campaigns: the TTS system that has been the center of this thesis (methods and techniques are described in chapters 3 and 4 and the system evaluation is presented in chapter 6) and the MT system which will be briefly introduced.

7.2.1 An overview of the adjacent technologies

The RACAI MT system uses a *statistical phrase–based decoder* built using MOSES (Koehn, 2013). One issue with *statistical machine translation* (SMT) systems such as MOSES is that when relying on the classical SMT approach based only on word surface–form analysis, highly inflectional languages such as Romanian are likely to suffer from the *scarcity of training data*. Such systems build a *translation equivalents table* composed on sequences of words in one language (S), their equivalent translation in the target language (T) with an associated set of values that refer to the inverse phrase translation probability $P(T|S)$, inverse lexical weighting $\mathrm{lex}(T|S)$, direct phrase translation probability $P(S|T)$ and direct lexical weighting $\mathrm{lex}(S|T)$. The parameters are well described and explained in the MOSES user manual (Koehn, 2013). *Factored translation models* are designed to reduce the data–scarcity issue for highly inflectional languages by adding to the classical wordform model additional information such as the word's dictionary form (lemma) and part of speech information. Using this improved model it becomes possible to perform various factor–to–factor translations (ex: lemma/POS in language A $\rightarrow$ lemma/POS in language B) as well as integrating in the translation process the so called generation steps in which each lemma/POS pair in the resulting translation is used to generate a number of candidate wordforms, out of which the output combination of wordforms that maximizes a language–model dependent score is chosen.

In all SMT systems, training is performed using *parallel corpora* (pairs of translated sentences in the source and target language) and such corpora are hard to obtain. Some well–known parallel corpora from MT system are DGT (Streinberger et al., 2012) and JRC–Acquis (Streinberger et al., 2006). Manually collecting or generating additional parallel corpora requires and extensive effort in terms of time and resources involved in such a process. However, *comparable corpora* are much easier to obtain, say by automatically crawling web–pages that are available in multiple languages (e.g. Wikipedia). With this in mind, in a recent effort to enhance the collection of parallel corpora from existing freely–available comparable corpora, the RACAI team developed an automatic method for extracting parallel sentences from the later mentioned corpora type. This method was successfully tested and implemented in a freely available tool, called Lucene–based parallel phrase EXtractor from Comparable Corpora (LEXACC) (Ştefănescu et al., 2012). This tool was also used to construct parallel corpora for the English–German, English–Romanian and English–Spanish language pairs, thus leading to the creation of the Parallel Wiki corpus (Ştefănescu and Ion, 2013). The validity of this corpus as well as advanced factor translation tests were conducted and presented in a number of publications (Tufiş et al., 2013, Boroş et al., 2013; Dumitrescu et al., 2013; Dumitrescu et al., 2012)

Having access to the previously mentioned TTS and MT systems, the ASR task was solved by employing the services of an external, freely–available ASR system, namely Google ASR. Though a thorough evaluation of Google ASR is out of scope for this presentation, a brief introduction to the system is required: according to Jaitly et al. (2012), the later mentioned system is trained on a large speech corpus using a Sparse Imputation (SI) Gaussian Mixture Model – Hidden Markov Model (GMM–HMM) composed of context–dependent tri–phone HMMs with a three–state left–to–right topology. The feature set used by the system is obtained by performing Perceptual Linear Predictive (PLP) analysis on the input signal and transforming the results using Linear Discriminant Analysis (LDA). The model is discriminatively trained using boosted maximum–mutual–information (BMMI).

7.2.2 Issues in the design and implementation of a speech translation system

As other research has shown (Ney, 1999; Zhang et al., 2004), S2S architectures composed by independent ASR, MT and TTS components (see figure 11) lacks joint optimality, this is an easy to understand design and it offers a good baseline system for future research. However, linking together these technologies posed a series of challenges among which are divided within the following classes: data–sparseness, module coupling and general processing of OOV words.

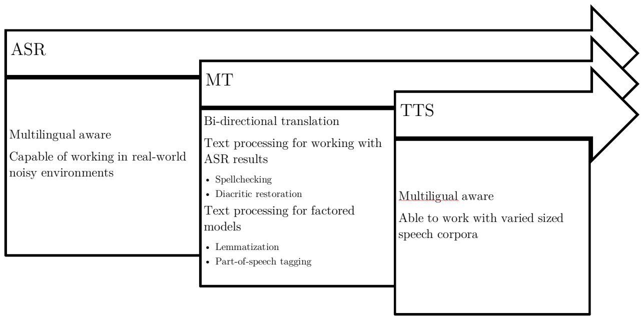

ASR
Multilingual aware
Capable of working in real-world noisy environments
MT
Bi-directional translation
Text processing for working with ASR results
• Spellchecking
• Diacritic restoration
Text processing for factored models
• Lemmatization
• Part-of-speech tagging
TTS
Multiligual aware
Able to work with varied sized speech corpora

Figure 11 – The S2S system architecture

The methods used for handling these issues were introduced in the previous sections and in what follows a brief review of the tasks and solutions will be presented in order to assess the complexity of a S2S translation system:

Morphologically rich languages, such as Romanian, suffer from *data–sparseness* because words have a large number of different forms that are not likely to exist in usual training corpora. This requires careful planning and preparation of the methods and data before constructing models. To reduce the issue of data–sparseness, the RACAI MT system uses the previously discussed factored translation models which rely on the word's lemma and part–of–speech information. Both tasks have been thoroughly addressed in chapter 3: see section 3.3 for challenges and solutions for POS tagging morphologically rich languages and section 3.7 for lemmatization of OOV words.

The processing of OOV words poses a series of challenges for all the sub–systems involved in S2S. Using an external ASR solution makes this issue currently out–of–reach and no solution for OOV words is currently discussable. The issue of OOV words in TTS was thoroughly addressed throughout chapter 3 of the thesis and it is a common practice in MT systems to use transliteration for OOV words (Vogel et al., 2003; Habash, 2008; Hermjackob et al., 2008), a task which was also presented in this thesis (section 3.8).

Module coupling challenges are generated by the fact the each of the MT and TTS synthesis systems have well defined and rather strict expectations from the input data. For example, the MT system requires that the input text is well–

formed, has diacritics and that the words are correctly cased (i.e. “delta dunării” or “delta dunarii” do not have translation equivalents inside the phrase table but “Delta Dunării” has – note the casing and diacritics – and is equivalent with “Danube Delta”). However, the output of the Google ASR system is all lowercased and most words have no diacritics, which involves adding a diacritic restoration and word–casing step before the output text from the ASR is processed by the MT system (diacritic restoration and word–casing is well explained in section 3.4). The MT model is build using the LEXACC Tool to automatically extract parallel corpora from the Wikipedia comparable corpora and a model built from such data is not always correct. However, the TTS system automatically performs diacritic restoration and word–casing on the input text, thus assuring that the impact of such errors is reduced.

7.3 Conclusions

There are many applications that benefit from exploiting the increased accessibility provided by TTS synthesis, which have been thoroughly presented in this chapter. A sensible portion of them were designed to assure assistance to the elderly or disabled people but, as shown in the previous sections, voice interfaces can be used in assisting and providing a better consumer experience to regular users as well.

Currently TTS synthesis systems are part of many platforms and as technology progresses they will become more a requirement than an experience enhancement tool as users get accustomed to using voice interaction as an input method when working with computers.

A number of typical assistive technologies and applications that are based on speech synthesis have been presented in this chapter in order to prove the benefit of TTS synthesis. Additionally, a method for integrating MT, TTS and ASR to build S2S translation system has been investigated and described and an android–based prototype was implemented as proof–of–concept. The working prototype (Romanian–English bi–directional speech translation) was part of a keynote–demo presentation at the 2013 Speech Technology and Human–Computer Dialogue Conference in Cluj, Romania (Tufiș et al., 2013) and offers a solid baseline system for further development.

Chapter 8

Conclusions

The thesis covers interesting aspects related to Romanian speech synthesis domain and it introduces *original approaches* to classical NLP–related issues of TTS synthesis that show *state–of–the art results*.

A fully functional and tested platform for text–to–speech synthesis was described in detail and the *original methods for handling OOV words* in various TTS tasks such as *syllabification*, *grapheme to phoneme conversion* and *stress prediction* were thoroughly tested. Additionally, aspects regarding the data–sparseness issue for *part–of–speech tagging*, the presence of *foreign words* in texts and normalization related issues such as *diacritic restoration* and *spellchecking* have been fully addressed and original or adapted solutions were provided and tested for solving these issues.

Another major contribution to Romanian TTS synthesis was the *construction and evaluation of the RSS–ToBI corpus*[30]. The experimental results showed a significant quality gain by using this corpus for training in TTS synthesis, proving that this corpus is truly an asset for building speech models in corpora based methods.

The *proof–of–concept application for speech translation* shows the viability of the proposed methods and techniques and addresses some module coupling of ASR, MT and TTS and data–sparseness related issues, offering a general purpose assistive technology on mobile platforms.

The tools and resources created during the preparation of this thesis can be applied for other applications such as:

- Spellchecking, diacritic restoration and word–recasing offers a generic functionality as a text–proofing tool;

[30] The corpus is available through the META-SHARE platform http://ws.racai.ro:9191

- The entire TTS system can be used as a module in other applications, similarly to the case of the speech translation system;
- All the tools are freely provided and they can be used as baselines for developing new methods or improving the existing ones for the tasks they were designed;
- The RSS–ToBI corpus can be used in training and testing various methods for predicting prosody starting from text or in the development of expressive speech synthesis systems.

The evaluation of the TTS platform during the Blizzard Challenge campaign showed that *the author's proposed methods and tools are up to par with current state–of–the art systems* and has led to the conclusion that a future research priority is the development of a hybrid speech synthesis module for the TTS platform. Additionally, the creation of the RSS–ToBI corpus will further help with the development, adaptation and testing of a method for automatically extracting prosodic features from text in order to enable Romanian speech synthesis systems provide highly expressive synthetic speech.

8.1 Personal contributions

During the PhD program, as a member of RACAI NLP group, I developed several tools and applications with competitive performances, as shown by the evaluation campaigns in which the NLP group of RACAI participated (Microsoft Speller Challenge, International Workshop on Spoken Language Translation, Blizzard Challenge, etc.).

Software development:

(1) ***RACAI Spellchecker***: HMM Based spellchecking Tool that was evaluated during the International Microsoft Speller Challenge and placing fourth among a number of 300 systems (described in section 3.5).

(2) ***The RACAI Neural MSD POS Tagger*****:** part-of-speech tagging application that uses Feed Forward Neural Networks and genetic optimizations, which is designed to work well on morphologically rich languages, such as Romanian (described in section 3.3)

(3) ***RACAI Diacritic Restoration and Word re-casing tool***: A HMM based diacritic restoration and word-recasing tool designed for text pre-processing in MT and TTS applications (described in section 3.4)

(4) ***RACAI SYL*****:** a word syllabification tool bases on the MIRA classifier and the ONC labelling strategy designed for TTS and general word-processing applications (described in section 3.7.1)

(5) ***RACAI G2P:*** a grapheme to phoneme conversion tool for OOV words that is based on the MIRA classifier and GIZA++ automatically induced alignments (described in section 3.7.3)

(6) ***RACAI STRESS:*** lexical stress prediction tool for OOV words (described in section 3.7.4)

(7) ***RACAI LEMMA:*** a tool designed for the lemmatization of OOV words designed for factored MT (described in section 3.7.2)

(8) ***RACAI TRANSLIT*:** a tool designed for transliterating foreign words in Romanian, which is usable for TTS synthesis of non-native Romanian words (described in section 3.8)

(9) ***RACAI TTS:*** a multilingual TTS synthesis system that was evaluated in the Blizzard Challenge 2013 (chapter 6) and is presented throughout the entire thesis (see chapters 3 and 4 for the description of the methods and tools)

(10) ***RACAI S2S:*** an Android based speech to speech translation system designed for English and Romanian bidirectional conversion (described in section 7.2)

Data collecting and evaluation:

(1) ***RSS-ToBI:*** a prosodically enhanced speech synthesis corpus built by manually labelling the fairy-tale section of the RSS corpus with RO-ToBI labels (described in section 5.3)

(2) ***Evaluation of the RSS-ToBI corpus*:** assessing the contribution of the RSS-ToBI corpus in a crowd-sourcing initiative (described in section 5.4)

(3) ***Crowd-sourced evaluation and data-collection portals:*** the creation of the Romanian Anonymous Speech Corpus (RASC[31]) and the Romanian Spoken Language Processing

[31] http://rasc.racai.ro

portal (where the evaluation of the RSS-ToBI corpus was performed[32])

Published papers during the PhD program:

[1]. **Boroş, T.**, Ion, R. and Dumitrescu, Ș.D. (2013). *The RACAI Text-to-Speech Synthesis System*. In Blizzard Challenge, Speech Synthesis Workshop (SSW), Barcelona, Spain.

[2]. **Boroş, T.** (2013). *A Unified Lexical Processing Framework based on the Margin Infused Relaxed Algorithm. A Case Study on the Romanian Language*. In Proceedings of the Recent Advances in Natural Language Processing (RANLP) Conference 2013, ISSN 1313-8502, pp. 91—97, Hissar, Bulgaria.

[3]. Dumitrescu, Ș.D., **Boroș, T.** (2013). A unified corpora-based approach to Diacritic Restoration and Word Casing. Accepted for publication in Language Technology Conference (LTC) 2013.

[4]. Tufiş, D. and **Boroş, T.**, Dumitrescu, Ș.D. (2013). *The RACAI Speech Translation System. Challenges of Morphologically Rich Languages*. In Proceedings of Speech Technology and Human-Computer Dialogue (SpeD), ISBN: 978-1-4799-1065-6, Cluj-Napoca, Romania.

[5]. **Boroş, T.** and Tufiş, D. (2013). *Romanian-English Speech Translation*. Accepted for publication in Proceedings of the Romanian Academy, ISSN: 1454-9069.

[6]. ZAFIU, A. and **Boroș, T.** (2013). Maximum entropy based machine transliteration. Applications and results. In Proceedings of CONSILR 2013., ISSN 1843-911X, pp.163—174, Miclăușeni, Iași, Romania.

[7]. **Boroş, T.** and Dumitrescu, Ș.D. (2013). *Improving the RACAI Neural MSD Tagger*. In Engineering Applications of Neural Networks, Springer Berlin Heidelberg, ISBN 978-3-642-41012-3, pp. 42—51, Halkidiki, Greece.

[8]. **Boroş, T.**, Ion, R. and Tufiş, D. (2013). *Large tagset labeling using Feed Forward Neural Networks. Case study on Romanian Language*. In Proceedings of the Association of Computational Linguistics, ISBN 978-1-937284-50-3, pp. 692—700, Sofia, Bulgaria.

[9]. **Boroş, T.**, Ion, R. and Ștefănescu, D. (2013). *Data-Driven Methods For Phonetic Transcription Of Out-Of-Vocabulary (OOV) Words*. In

[32] http://rslp.racai.ro/index.php?page=experiment/listening

Proceedings of CONSILR 2013., ISSN 1843-911X, pp.163—174, Miclăușeni, Iași, Romania.

[10]. Tufiş, D., Ion, R. and **Boroş, T.** (2013). *Filtering, Normalizing and Spell-Checking the Romanian N-Grams from Google's Web 1T 5-grams.* Book chapter in Torward a Multilingual Europe. Editura Academiei Române, ISBN 978-973-27-2282-4, pp. 123—142, București, România.

[11]. Dumitrescu, Ș.D., Ion, R., Ștefănescu, D., **Boroş, T.** and Tufiș, D. (2013). *Experiments on language and translation models adaptation for statistical machine translation.* Book chapter in Torward a Multilingual Europe. Editura Academiei Române, ISBN 978-973-27-2282-4, pp. 205—224, București, România.

[12]. **Boroş, T.** (2013). *RACAI NLP Tools for Text-To-Speech Synthesis.* Book chapter in Torward a Multilingual Europe. Editura Academiei Române, ISBN 978-973-27-2282-4, pp. 265—286, București, România.

[13]. **Boroş, T.** and Zafiu, A. (2012). *Transliterare automată din engleză în română. Aplicaţii şi rezultate.* In Revista Română de Interacţiune Om-Calculator, Vol.5, Nr.3, ISSN 1843-4460, pp. 1—14.

[14]. **Boroș, T.**, Ștefănescu, D. and Ion, R. (2013). *Handling Two Difficult Challenges for Text-to-Speech Synthesis Systems: Out-of-Vocabulary Words and Prosody -- A Case Study in Romanian.* Book chapter in Where Humans Meet Machines, ISBN 978-1-4614-6934-6, pp. 137—161 edited by Amy Neustein, Springer.

[15]. **Boroş, T.**, Ion, R. and Ștefănescu, D. (2012). *Bermuda, a data-driven tool for phonetic transcription of words.* In Proceedings of Language Resources and Evaluation Workshops, ISBN 978-2-9517408-7-7, pp. 35—39.

[16]. Dumitrescu, Ș.D., Ion, R., Ştefănescu, D., **Boroş, T.** and Tufiş, D. (2012). *Romanian to English Automatic MT Experiments at IWSLT12 (system description paper).* In Proceedings of IWSLT, Hong Kong, HK, pp. 136—143.

[17]. **Boroș, T.** (2012). *Blind Speech Segmentation applied to the Romanian Language.* In Proceedings of the 8th International Conference "Linguistic Resources and Tools of the Romanian Language", ISSN 1843-911X, pp. 11—16.

[18]. Mititelu, V.B., **Boroş, T.**, Forăscu, C., Ion, R., Irimia, E. and Tufiş, D. (2012). *Laying the Foundation for the Representative Corpus of Contemporary Romanian.* In Proceedings of the 8th International

Conference "Linguistic Resources And Tools For Processing The Romanian Language", ISSN 1843-911X, pp. 39—46.

[19]. Ştefănescu, D., Ion, R. and **Boroş, T.** (2011). *TiradeAI: An Ensemble of Spellcheckers*. In Proceedings of the Spelling Alteration for Web Search Workshop, pp. 20-23, Bellevue, USA.

[20]. **Boroş, T.** (2011). *Speech Segmentation for Collecting Non-Uniform Speech Units*. In Proceedings of Language & Technology Conference, Poznan, Poland., ISBN 978-83-932640-1-8, pp. 456—460.

[21]. Ion, R., Tufiș, D., **Boroș, T.**, Ceaușu, A. and Ștefănescu, D. (2010). On-Line Compilation of Comparable Corpora and their Evaluation. In Proceedings of The 7th International Conference Formal Approaches to South Slavic and Balkan Languages (FASSBL 2010) (Tadic, Marko and Dimitrova-Vulchanova, Mila and Koeva, Svetla). Zagreb, Croația, pp. 29—34.

[22]. **Boroș, T.**, Tufiș, D. and Ceaușu, A. (2010). Automatic collection of comparable corpora (Construcția automată de corpusuri multilingve). In Lucrările celui de-al 5-lea Atelier de Resurse Lingvistice și Instrumente pentru Prelucrarea Limbii Române, ISSN 1843-911X, pp. 103—112.

[23]. Boroş, T. and Ioniţă, A. (2013). *Data mining from unstructured sources for forest fire forecasting applications*, poster presentation at INSPIRE, Florence, Italy.

[24]. Franți, E., Ștefan, G, Șchiopu, P., Plăvițu, A. and Boroș, T. (2011). Modular Software for Artificial Arms Design. In Recent Researches in Automatic Control. Canare Islands, Spain, ISSN: 2223-2907, pp. 387—391.

[25]. Franți, E., Ștefan, G., Șchiopu, P., Boroș, T. and Plăvițu, A. (2011). Intelligent control system for artificial arms configuration. In Proceedings of the European Computing Conference. Paris, France, ISBN: 978-960-474-297-4, pp. 312—316.

Appendixes

10.1 List of abbreviations

AI	-	Artificial Intelligence
ANN	-	Artificial Neural Networks
ASR	-	Automatic Speech Recognition
CART	-	Classification And Regression Trees
CLAWS	-	Constituent Likelihood Automatic Word-Tagging System
CMU	-	Carnegie Mellon University
CRF	-	Conditional Random Field
DCT	-	Discrete Cosine Transform
DE	-	German
DFT	-	Discrete Fourier Transform
DSP	-	Digital Signal Processing
DTFT	-	Discrete Time Fourier Transform
EM	-	Expectation Maximization
EN	-	English
FFN	-	Feed Forward Neural Networks
FFT	-	Fast Fourier Transform
G2P	-	Grapheme To Phoneme
HCI	-	Human Computer Interaction
HMM	-	Hidden Markov Model
ID3	-	Iterative Dichotomiser 3
IPA	-	International Phonetic Alphabet
IT	-	Italian
LM	-	Language Model
LSTM	-	Long–Short Term Memory
LTS	-	Letter To Sound
MARY	-	Modular Architecture For Research On Speech Synthesis
MAXENT	-	Maximum Entropy Classifiers
MFC	-	Mel-Frequency Cepstral/Cepstrum

MFCC	-	Mel Frequency Cepstral Coefficients
MIRA	-	Margin Infused Relaxed Algorithm
ML	-	Machine Learning
MLE	-	Maximum Likelihood Estimation
MSD	-	Morpho-Syntactic Description/Descriptor
MT	-	Machine Translation
NLP	-	Natural Language Processing
ONC	-	Onset Nucleus Coda
OOV	-	Out Of Vocabulary
PLP	-	Perceptual Linear Predictive
POS	-	Part Of Speech
PSOLA	-	Pitch Synchronous Overlap And Add
PT	-	Phonetic Transcription
RACAI	-	Romanian Academy - Center For Artificial Intelligence
RASTA	-	Relative Spectral Transform - Perceptual Linear Prediction
RBM	-	Restricted Boltzmann Machine
RO	-	Romanian
RSS	-	Romanian Speech Synthesis
RU	-	Russian
SAMPA	-	Speech Assessment Methods Phonetic Alphabet
SLP	-	Spoken Language Processing
SVM	-	Support Vector Machine
TL	-	Telugu
TOBI	-	Tones And Break Indices
TR	-	Turkish
TTS	-	Text To Speech

10.2 List of figures

Figure 1 – Basic TTS architecture 2
Figure 2 – Example of a sampled signal 12
Figure 3 – Tiered Tagging methodology 25
Figure 4 – 130 hidden layer network test and train set tagging accuracy as a function of the number of iterations 31
Figure 5 – PSOLA pitch and duration modification (Huang et al., 2001) 72
Figure 6 – Highest occurring ToBI labels inside the RSS–ToBI corpus . 82
Figure 7 – Preference test results for the two systems 83
Figure 8 – RACAI TTS (system J) results (similarity to the original speaker – all listeners/all data) 90
Figure 9 – RACAI TTS (system J) results (naturalness – all listeners/novel) 91
Figure 10 – RACAI TTS (system J) results (naturalness – all listeners/news) 91
Figure 11 – The S2S system architecture 98

10.3 List of tables

Table 1 – Correspondence between the RSS notation standard and SAMPA – adapted from Stan (2011) ... 8
Table 2 – Vowel classification based on articulatory features ... 9
Table 3 – Classification of consonants based on articulatory features ... 11
Table 4: Details on open–source TTS systems NLP modules. Accuracy figures are shown on all / only–OOV words. For phonetic transcription and lexical stress assignment the statistics are computed on CMUDICT ... 23
Table 5 – Execution time vs. number of neurons on the hidden layer ... 31
Table 6 – Train and test accuracy rates for different hidden layer configurations ... 32
Table 7 – Average and maximum accuracy for various hidden layer configuration calculated over 100 training iterations on the test set ... 34
Table 8 – P/N features for various attribute values ... 35
Table 9. Manual and genetic network topology results ... 39
Table 10. Tagging speeds of different network topologies ... 40
Table 11. Corpus composition ... 43
Table 12. Diacritic restoration results ... 46
Table 13 – Word Casing results ... 47
Table 14 – Joint task results ... 48
Table 15 – Comparison with previous experiments ... 48
Table 16 – Lemmatization example for word "îmbrăcați" ... 56
Table 17 – Lemmatization results ... 57
Table 18 – G2P experiments and results using various lexicons and methods ... 62
Table 19 – Lexical stress tagging for the word “îmbrăcaţi” ... 63
Table 20 – Lexical stress accuracy ... 63
Table 21 – Advantages and disadvantages of unit selection and statistical parametric speech synthesis ... 70

References

[1]. Ackley, D. H., Hinton, G. E., & Sejnowski, T. J. (1985). A learning algorithm for Boltzmann machines. Cognitive science, 9(1), 147–169.

[2]. Allen, J., Hunnicutt, M. S., Klatt, D. H., Armstrong, R. C., & Pisoni, D. B. (1987). From text to speech: the MITalk system. Cambridge University Press.

[3]. Baayen, R., Piepenbrock, R., and Gulikers, L. (1995). The CELEX lexical database. Linguistic Data Consortium, University of Pennsylvania, Philadelphia, 1995.

[4]. Bartlett, S., Kondrak, G., & Cherry, C. (2008). Automatic syllabification with structured SVMs for letter-to-phoneme conversion. In Proceedings of ACL: HLT, 568-576.

[5]. Beckman, M., & Pierrehumbert, J. (1986). Intonational structure in Japanese and English. Phonology yearbook, 3(1), 5-70.

[6]. Beckman, M. E., & Hirschberg, J. (1994). The ToBI annotation conventions. Ohio State University.

[7]. Bisani, M., & Ney, H. (2008). Joint-sequence models for grapheme-to-phoneme conversion. Speech Communication, 50(5), 434-451.

[8]. Bisani, M., and Ney, H. (2002). Investigations on joint–multigram models for grapheme–to phoneme conversion. Proceedings of the 7th International Conference on Spoken Language Processing, pages 105–108

[9]. Black, A. W., Lenzo, K., & Pagel, V. 1998. Issues in building general letter to sound rules. In The Third ESCA/COCOSDA Workshop (ETRW) on Speech Synthesis.

[10]. Black, A., & Tokuda, K. (2005). The Blizzard Challenge 2005: Evaluating corpus-based speech synthesis on common databases. In Proceedings of Interspeech.

[11]. Blackman, R. B. and Tukey, J. W. "Particular Pairs of Windows." In The Measurement of Power Spectra, From the Point of View of Communications Engineering. New York: Dover, pp. 98-99, 1959.

[12]. Boroș, T., (2013). A unified lexical processing framework based on the Margin Infused Relaxed Algorithm. A case study on the Romanian Language. In Proceedings of RANLP, Hissar, Bulgaria.

[13]. Boroș, T., Dumitrescu, Ş.D. (2013) Improving the RACAI Neural MSD tagger. In Proceedings of EANN 2013, Halkidiki, Greece.

[14]. Boroș, T., Dumitrescu, S.D., Ion, R., Ștefănescu, D. and Tufiș, D. (2013a). Romanian-English Statistical Translation at RACAI. In E. Mitocariu, M. A.

Moruz, D. Cristea, D. Tufiş, M. Clim (eds.) Proceedings of the 9th LREC, 16-17 mai, 2013,, Miclăușeni, Romania, 2013. „Alexandru Ioan Cuza" University Publishing House. 197 p. ISSN 1843-911X. pp. 81-98.

[15]. Boroș, T., Ion, R., Tufiș, D. (2013). Large tagset labeling using Feed Forward Neural Networks. Case study on Romanian Language. in Proceedings of ACL, Sofia, Bulgaria, August 3-7.

[16]. Boroș, T., Ștefănescu, D. and Ion, R. (2013). Handling Two Difficult Challenges for Text-to-Speech Synthesis Systems: Out-of-Vocabulary Words and Prosody -- A Case Study in Romanian. Book chapter in Where Humans Meet Machines, ISBN 978-1-4614-6934-6, pp. 137—161 edited by Amy Neustein, Springer.

[17]. Boroș, Tiberiu and Dumitrescu, Ștefan Daniel and Ion, Radu and Ștefănescu, Dan and Tufiș, Dan. Romanian–English Statistical Translation at RACAI. In Proceedings of the 9th International Conference "LINGUISTIC RESOURCES AND TOOLS FOR PROCESSING OF THE ROMANIAN LANGUAGE" (Mitocariu, E. and Moruz, M. A. and Cristea, D. and Tufiș, Dan and Clim, M.). "Alexandru Ioan Cuza" University Publishing House, Miclăușeni, Romania, pp. 81–98, May 2013

[18]. Bosch, A., and Canisius, S. (2006). Improved morpho phonological sequence processing with constraint satisfaction inference. Proceedings of the Eighth Meeting of the ACL–SIGPHON at HLT–NAACL, pages 41–49.

[19]. Breen, G., & Pensalfini, R. (1999). Arrernte: A language with no syllable onsets. Linguistic Inquiry, 30(1), 1-25.

[20]. Burileanu, D. (1999). Contributions on Speech Synthesis from Text in Romanian Language, PhD Thesis

[21]. Burileanu, D., Sima, M., & Neagu, A. 1999. A phonetic converter for speech synthesis in Romanian. In Proceedings of ICPhS'99.

[22]. Campbell, N., & Venditti, J. (1995, September). J-ToBI: An intonation labelling system for Japanese. In Proceedings of the Autumn meeting of the Acoustical Society of Japan (Vol. 1, pp. 317-318).

[23]. Carlsson, J., Paiz, C., Wolff. K., Nordin, P. (2002). Interactive Evolution of Speech using VoiceXML Speaking to you GP System. In Proceedings of the 6th World Multiconference on Systemics, Cybernetics and Informatics, VI, page 58--62. IIIS.

[24]. Ceauşu, A.,and Tufiș, D. (2011). Addressing SMT Data Sparseness when Translating into Morphologically-Rich Languages. In Proceedings of the 8th NLPCS workshop. Special theme: Human-machine interaction in translation, pp. 57-68, Copenhagen Business School.

[25]. Chen, H., & Murray, A. F. (2003, June). Continuous restricted Boltzmann machine with an implementable training algorithm. In Vision, Image and Signal Processing, IEE Proceedings– (Vol. 150, No. 3, pp. 153–158). IET.

[26]. CMU (2011). Carnegie Mellon Pronuncing Dictionary. http://www.speech.cs.cmu.edu/cgi–bin/cmudict.

[27]. Content, A., Mousty, P., and Radeau, M. (1990). Une base de données lexicales informatisée pour le français écrit et parlém. L'Année Psychologique, 90:551–566.

[28]. Cortes, C., & Vapnik, V. (1995). Support vector machine. Machine learning, 20(3), 273–297.

[29]. Daelemans, W., Van Den Bosch, A., & Weijters, T. 1997. IGTree: Using trees for compression and classification in lazy learning algorithms. Artificial Intelligence Review, 11(1), 407-423.

[30]. Demberg, V., Schmid, H., & Mohler, G. 2007. Phonological constraints and morphological preprocessing for grapheme-to-phoneme conversion. In Annual Meeting-Association for Computational Linguistics (Vol. 45, No. 1, p. 96).

[31]. DeRose, S. J. (1988). Grammatical category disambiguation by statistical optimization. Computational Linguistics, 14(1), 31-39.

[32]. Divay, M. and Vitale, A. J. (1997). Algorithms for grapheme–phoneme translation for English and French: Applications, Computational Linguistics, 23(4):495–524.

[33]. Dumitrescu, S.D., Ion, R., Ștefănescu, D., Boroș, T. and Tufiș, D. (2012). Romanian to English Automatic MT Experiments at IWSLT12. In Proceedings of IWSLT, Hong Kong 136-143

[34]. Dumitrescu, S.D., Ion, R., Ștefănescu, D., Boroș, T. and Tufiș, D. (2013). Experiments on Language and Translation Models Adaptation for Statistical Machine Translation. In Towards a Multilingual Europe 2020: A Romanian Perspective, pp. 205-224.

[35]. Dumitrescu, Ștefan Daniel and Ion, Radu and Ștefănescu, Dan and Boroș, Tiberiu and Tufiș, Dan. Romanian to English Automatic MT Experiments at IWSLT12. In Proceedings of the International Workshop on Spoken Language Translation. Hong Kong, pp. 136–143, December 2012

[36]. Elman, J. L. (1990). Finding structure in time. Cognitive science, 14(2), 179–211.

[37]. Erjavec, T., & Monachini, M. (1997). Specifications and notation for lexicon encoding. COP Project, 106.

[38]. Enkvist, N. E. (1982). Impromptu Speech: A Symposium. Papers Contributed to a Symposium on Problems in the Linguistic Study of Impromptu Speech

(Abo, Finland, November 20-22, 1981). Meddelanden fran Stiftelsens for Abo Akademi Forskningsinstitut Nr. 78.

[39]. Fant, G. (1970). Acoustic theory of speech production (No. 2). Walter de Gruyter.

[40]. Forney Jr, G. D. (1973). The Viterbi algorithm. Proceedings of the IEEE, 61(3), 268-278.

[41]. Franzén, V., & Horne, M. 2009. Word stress in Romanian. Lund Working Papers in Linguistics, 46, 75-91.

[42]. Habash, N., & Sadat, F. (2006). Arabic preprocessing schemes for statistical machine translation.

[43]. Hermansky, H. (1990). Perceptual linear predictive (PLP) analysis of speech. The Journal of the Acoustical Society of America, 87, 1738.

[44]. Hermansky, H., Morgan, N., Bayya, A., & Kohn, P. (1992, March). RASTA-PLP speech analysis technique. In Acoustics, Speech, and Signal Processing, 1992. ICASSP-92., 1992 IEEE International Conference on (Vol. 1, pp. 121-124). IEEE.

[45]. Hermjakob, U., Knight, K., & Daumé III, H. (2008, June). Name Translation in Statistical Machine Translation–Learning When to Transliterate. In ACL (pp. 389–397).

[46]. Hinton, G. E., & Sejnowski, T. J. (1986). Learning and relearning in Boltzmann machines. MIT Press, Cambridge, Mass, 1, 282–317.

[47]. Hinton, G. E., Osindero, S., & Teh, Y. W. (2006). A fast learning algorithm for deep belief nets. Neural computation, 18(7), 1527–1554.

[48]. Hochreiter, S., & Schmidhuber, J. (1997). Long short–term memory. Neural computation, 9(8), 1735–1780.

[49]. Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. Proceedings of the national academy of sciences, 79(8), 2554–2558.

[50]. Huang, X., Acero, A., & Hon, H. W. (2001). Spoken language processing (Vol. 15). New Jersey: Prentice Hall PTR.

[51]. International Phonetic Association (Ed.). (1999). Handbook of the International Phonetic Association: A guide to the use of the International Phonetic Alphabet. Cambridge University Press.

[52]. Ion, R. (2007). Word Sense Disambiguation Methods Applied to English and Romanian, PhD thesis (in Romanian). Romanian Academy, Bucharest.

[53]. Jaitly, N., Nguyen, P., Senior, A. W., & Vanhoucke, V. (2012). Application of Pretrained Deep Neural Networks to Large Vocabulary Speech Recognition. In Proceedings of INTERSPEECH

[54]. Jiampojamarn, S., Cherry, C. and Kondrak, G. (2008). Joint processing and discriminative training for letter–to–phoneme conversion (2008). Proceedings of ACL–2008: Human Language Technology Conference, pp. 905–913, Columbus, Ohio.

[55]. Jitcă, D., Apopei, V., & Păduraru, O. (2012). The Ro-Tobi Annotation System and the Functional Analysis Perspective of the Romanian Intonation. In Proceedings of CONSILR, ISSN 1843-911x, 3.

[56]. King, S. and Karaiskos, V. (2009). The Blizzard Challenge 2009. In Proc. Blizzard Challenge Workshop, Edinburgh, UK.

[57]. Kleiner, A. (1977). A description of the Kurzweil reading machine and a status report on its testing and dissemination. Bull Prosthet Res, 10(27), 72–81.

[58]. Koehn, P. (2010). MOSES, Statistical Machine Translation System, User Manual and Code Guide.

[59]. Koehn, P., & Hoang, H. (2007). Factored Translation Models. In EMNLP-CoNLL, 868-876.

[60]. Ladd, D. R. (1986). Intonational phrasing: the case for recursive prosodic structure. Phonology Yearbook, 3(311), 40.

[61]. Laurent, A., Deleglise, P., and Meignier, S. (2009. Grapheme to phoneme conversion using an SMT system. Interspeech.

[62]. Ladd, D. R. (1986). Intonational phrasing: the case for recursive prosodic structure. Phonology Yearbook, 3(311), 40.

[63]. Lavie, A., Waibel, A., Levin, L., Finke, M., Gates, D., Gavalda, M., & Zhan, P. (1997). JANUS-III: Speech-to-speech translation in multiple languages. In ICASSP, IEEE International Conference, 99-102.

[64]. Lee, S., & Oh, Y. H. (1999). Tree-based modeling of prosodic phrasing and segmental duration for Korean TTS systems. Speech Communication, 28(4), 283-300.

[65]. Liberman, M., & Prince, A. (1977). On stress and linguistic rhythm. Linguistic inquiry, 8(2), 249-336.

[66]. Lita, L. V., Ittycheriah, A., Roukos, S., & Kambhatla, N. (2003, July). tRuEcasIng. In Proceedings of ACL. Association for Computational Linguistics.

[67]. Marchand, Y. and Damper, R.I. (2000). A multistrategy approach to improving pronunciation by analogy. Computational Linguistics, 26(2):195–219.

[68]. Marchand, Y., & Damper, R. I. (2007). Can syllabification improve pronunciation by analogy of English?. In Natural Language Engineering, 13(1), 1-24.

[69]. Ney, H. (1999). Speech translation: Coupling of recognition and translation. In ASSP, IEEE International Conference, 517-520.

[70]. Ney, H. (1999). Speech translation: Coupling of recognition and translation. In ASSP, IEEE International Conference, 517–520.

[71]. Nespor, M., & Vogel, I. (1983). Prosodic structure above the word. In Prosody: Models and measurements (pp. 123-140). Springer Berlin Heidelberg.

[72]. Nyquist, H. (1928). Certain topics in telegraph transmission theory. American Institute of Electrical Engineers, Transactions of the, 47(2), 617-644.

[73]. Oancea, E., & Bădulescu, A. 2002. Stressed syllable determination for Romanian words within speech synthesis applications. In IJST, 5(3), 237-246.

[74]. Och, F. J., Ney, H. (2003). A Systematic Comparison of Various Statistical Alignment Models. Computational Linguistics, volume 29, number 1, pp. 19–51.

[75]. Pagel, V., Lenzo, K. and Black, A. (1998). Letter to sound rules for accented lexicon compression. International Conference on Spoken Language Processing, Sydney, Australia.

[76]. Papineni, K., Roukos, S., Ward, T., Zhu, W. J. (2002). BLEU: a method for automatic evaluation of machine translation. In Proceedings of ACL. Association for Computational Linguistics, 311–318

[77]. Qin, T., Liu, T. Y., & Li, H. (2007). The TREC Datasets in LETOR. Part of the TREC datasets in LETOR description.

[78]. Rama, T., Singh, A. K., Kolachina, S., (2009). Modeling Letter–to–Phoneme Conversion as a Phrase Based Statistical Machine Translation Problem with Minimum Error Rate Training. Proceedings of the 2009 Named Entities Workshop, ACL–IJCNLP 2009, pages 124–127, Suntec, Singapore.

[79]. Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (2002). Learning representations by back–propagating errors. Cognitive modeling, 1, 213.

[80]. Rus, V., Niraula, N., Lintean, M., & Graesser, A.C. (2013). An Overview of Dialogue and Semantic Processing in Educational Technologies, Chapter in (Eds. Forascu, C., Ionita, A., Tufis, D., Cristea, D. & Rus, V.) Language Technologies in Romanian and Diaspora Research & Development, September 2012, University Al.I. Cuza Publishing House, Iasi, Romania. ISBN 978–973–703–813–5

[81]. Selkirk, E. O. (1984). On the major class features and syllable theory.

[82]. Shannon, C. E. (1949). Communication in the presence of noise. Proceedings of the IRE, 37(1), 10-21.

[83]. Silverman, K. E., Beckman, M. E., Pitrelli, J. F., Ostendorf, M., Wightman, C. W., Price, P., Pierrehumbert, J. & Hirschberg, J. (1992). TOBI: a standard for labeling English prosody. In Proceedings of ICSLP (Vol. 2, pp. 867-870).

[84]. Stan, A. C. (2011). Romanian HMM-based text-to-speech synthesis with interactive intonation optimisation. PhD Thesis.

[85]. Stan, A., Yamagishi, J., King, S., & Aylett, M. (2011). The Romanian Speech Synthesis (RSS) corpus: building a high quality HMM-based speech synthesis system using a high sampling rate. In Speech Communication, 53(3), 442-450.

[86]. Stan, A., Yamagishi, J., King, S., Aylett, M. (2011). The Romanian Speech Synthesis (RSS) corpus: building a high quality HMM–based speech synthesis system using a high sampling rate. Speech Communication, vol.53, issue 3, pp.442–450.

[87]. Ştefănescu, D., & Ion, R. (2013). Parallel–Wiki: A Collection of Parallel Sentences Extracted from Wikipedia. In Proceedings of the 14 th International Conference on Computational Linguistics.

[88]. Ştefănescu, D., Ion, R., & Boroş, T. (2011). TiradeAI: An Ensemble of Spellcheckers. In Proceedings of SAWSW, 20-23.

[89]. Ștefănescu, D., Ion, R., & Hunsicker, S. (2012). Hybrid parallel sentence mining from comparable corpora. In Proceedings of the 16th Conference of the European Association for Machine Translation (EAMT 2012), pp. 137—144, Trento, Italy.

[90]. Steinberger, R., Eisele, A., Klocek, S., Pilos, S., & Schlüter, P. (2012). DGT–TM: A freely available Translation Memory in 22 languages. In LREC (pp. 454–459).

[91]. Steinberger, R., Pouliquen, B., Widiger, A., Ignat, C., Erjavec, T., Tufis, D., & Varga, D. (2006). The JRC–Acquis: A multilingual aligned parallel corpus with 20+ languages. arXiv preprint cs/0609058.

[92]. Sun, X., & Applebaum, T. H. (2001, September). Intonational phrase break prediction using decision tree and n-gram model. In INTERSPEECH (pp. 537-540).

[93]. Syrdal, A. K., Hirschberg, J., McGory, J. T., Beckman, M. E. (2001). Automatic ToBI prediction and alignment to speed manual labeling of prosody, Speech Communication 33(1-2):135-151.

[94]. Taylor, P. (2005). Hidden Markov Models for grapheme to phoneme conversion. Proceedings of the 9th European Conference on Speech Communication and Technology.)

[95]. Taylor, P. (2009). Text-to-speech synthesis, Cambridge University Press.

[96]. Tieleman, T. (2008, July). Training restricted Boltzmann machines using approximations to the likelihood gradient. In Proceedings of the 25th international conference on Machine learning (pp. 1064–1071). ACM.

[97]. Tokuda, K., Yoshimura, T., Masuko, T., Kobayashi, T., & Kitamura, T. (2000). Speech parameter generation algorithms for HMM-based speech synthesis. In Acoustics, Speech, and Signal Processing, 2000. ICASSP'00. Proceedings. 2000 IEEE International Conference on (Vol. 3, pp. 1315-1318). IEEE.

[98]. Tremain, T. E. (1982). The government standard linear predictive coding algorithm: LPC-10. Speech Technology, 1(2), 40-49.

[99]. Tufiş, D. (1999). Tiered tagging and combined language models classifiers. In TSD, Springer Berlin Heidelberg, 28-33.

[100]. Tufiş, D., & Ceauşu, A. (2008). DIAC+: A professional diacritics recovering system. In Proceedings of LREC.

[101]. Tufiş, D., & Dragomirescu, L. (2004). Tiered tagging revisited. In Proceedings of LREC, 39-42.

[102]. Tufiş, D., & Dumitrescu, S. D. (2012). Cascaded Phrase-Based Statistical Machine Translation Systems. In Proceedings of the 16th Conference of the EAMT, 129-136.

[103]. Tufiș, D., Boroș, T. and Dumitrescu, Ș. D. (2013). The RACAI Speech Translation System. In Proceedings of the 7th International Conference on Speech Technology and Human–Computer Dialogue (SPED 2013). Cluj–Napoca.

[104]. Tufiş, D., Ion, R., Ceauşu, A., & Ştefănescu, D. (2008). RACAI's Linguistic Web Services. In Proceedings of the 6th Language Resources and Evaluation Conference-LREC.

[105]. Tufiș, D., Ion, R., Dumitrescu, S. and Ștefănescu, D. (2013). Wikipedia as an SMT Training Corpus. In Proceedings of RANLP, Hissar, Bulgaria, September 10-13, 2013

[106]. Ungurean, C., Burileanu, D., & Dervis, A. (2009). A statistical approach to lexical stress assignment for TTS synthesis. In IJST, 12(2-3), 63-73.

[107]. Ungurean, C., Burileanu, D., Popescu, V. and Derviş, A. (2011). Hybrid Syllabification and Letter-To-Phone Conversion For TTS Synthesis. In U.P.B. Sci. Bull., Series C, Vol. 73, Iss. 3, 2011, ISSN 1454-234x

[108]. Vogel, S., Zhang, Y., Huang, F., Tribble, A., Venugopal, A., Zhao, B., & Waibel, A. (2003, September). The CMU statistical machine translation system. In Proceedings of MT Summit (Vol. 9, p. 54).

[109]. Waibel, A., Ahmed Bdran, A., Black, A.W., Frederking, R.E., Gates, D., Lavie, A., Levin, L.S., Lenzo, K., Tomokiyo, L.M., Reichert, J., Schultz, T., Wallace, D., Woszczyna, M., Zhang, J. (2003) Speechalator: two-way speech-to-speech translation on a consumer PDA. In Proceedings of Interspeech.

[110]. Weijters, A. 1991. A simple look-up procedure superior to NETtalk?. In Proceedings of ICANN, Espoo, Finland

[111]. Xipeng, S,, and Bo, X. (2000). A CART based hierarchical stochastic model for prosodic phrasing in Chinese." In Proc. of ISCSLP'00, pp. 105-108. 2000.

[112]. Zhang, R., Kikui, G., Yamamoto, H., Watanabe, T., Soong, F., & Lo, W. K. (2004). A unified approach in speech-to-speech translation: integrating features of speech recognition and machine translation. In Proceedings of COLING. ACL, Geneva, 1168-1174

[113]. Zhang, R., Kikui, G., Yamamoto, H., Watanabe, T., Soong, F., & Lo, W. K. (2004). A unified approach in speech–to–speech translation: integrating features of speech recognition and machine translation. In Proceedings of COLING. ACL, Geneva, 1168–1174