UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Università degli Studi di Padova

Dipartimento di Studi Linguistici e Letterari

Corso di Laurea Magistrale in Linguistica
Classe LM-39

Tesi di Laurea

Data preparation and improvement of NLP software modules for parametric speech synthesis

Relatrice
Prof.ssa Cecilia Poletto

Correlatori
Ing. Giacomo Sommavilla
Ing. Piero Cosi

Laureanda
Silvia Gardini
n° matr.1131198 / LMLIN

Anno Accademico 2017 / 2018

Abstract

Natural Language Processing (NLP) is a subfield of artificial intelligence and linguistics whose area of research and application covers computer processing, understanding and production of natural language. One of NLP applications is TTS (text-to-speech) synthesis, which is employed in healthcare, educational and commercial purposes, for example intelligent assistants such as Amazon Alexa or Apple Siri. The purposes of our work are two. The first is to demonstrate the relevance of LibriVox as speech corpus for improving the naturalness of the output synthetic speech. The second purpose is to arrange the generation of a feminine and a masculine synthetic voice for Spanish and Portuguese languages, in order to improve the NLP software modules for Mivoq parametric synthesiser. Our research consisted in the preparation and elaboration of data from LibriVox, the research of Portuguese corpora for POS tagging, the implementation of POS tagging module for Spanish by machine learning and the implementation of rule-based phonemisation module for Spanish. An actual perceptive assessment to prove the efficacy of LibriVox as data source did not take place, but our preparatory work laid the foundation for the future work.

Acknowledgements

Mivoq [1] srl is a spin-off of CNR-ISTC (Istituto di Scienze e Tecnologie della Cognizione [2]) of Padua, which develops a speech synthesis technology implementing personality, intended as similarity of the digital speech to the authentic voice of the speaker.

Heartfelt thanks to Piero Cosi, Giacomo Sommavilla, Giulio Paci and Fabio Tesser for the unvaluable opportunity to work with them as a team.

I would like to address my special thanks to Giacomo Sommavilla. He has instilled me his personal and professional approach and has been open to encounter mine, promoting a mutual exchange of ideas and aptitudes which has encouraged me to give my best.

Sincere thanks to Giulio Paci for his constructive and pragmatic contribution.

Finally, I express my gratitude to my friend Simone, for his accurate and stimulating suggestions whenever I needed, and to all the ones who supported me throughout my research.

[1] MImic VOice Quest

[2] Institute of Cognition Sciences and Technologies

Contents

Chapter 1

Introduction

"It is more important to teach an approach and a style of use than just details."
"The best way to learn something is by doing it. [...] you can experiment, verify or contradict what we say, explore the limits and the variations."

[22]

Natural Language Processing (NLP) is a field of research and application of computer processing, understanding and production of natural language. It is considered as a subfield of artificial intelligence and its methods are employed in applications where engineering techniques have to be underpinned by scientific understanding of language, such as machine translation, speech recognition and speech synthesis. NLP researchers, unlike computational linguists which focus on the development of formalisms, are interested in more practical engineering issues, such as finding efficient algorithms for language computing programs [37].

Among the NLP applications, we focus on Text-to-Speech (TTS) synthesis, a system which generates natural and intelligible human-like speech. It is engaged in healthcare, educational and commercial purposes, for instance in the form of aids for the visually or vocally impaired, railway station announcements and car navigation. Intelligent personal assistants are spreading rapidly worldwide, such as Amazon Alexa, Google assistant or Apple Siri. In the near future people may be able to interactively talk with all kind of devices, in and outside home, which is the direction taken by the emerging network of Internet of Things.

Concerning its functioning, the most prominent TTS synthesisers relies on recordings of authentic human speech, which are stored in a database and aligned to the correspondent text. After that multiple software modules are

implemented, given any text as input, the synthesiser produces as output any required speech. Such modules first convert the input text into a machine-readable representation which includes word sequences, their pronunciation and the relations among them, expressed through tags and labels next to the words or tokens. Then such text-based abstract representation is turned into a parameter-based representation. Finally, it is converted into a sequence of waveforms which are sent to an audio device to produce spoken language [6].

The TTS system employed in our research uses the statistical parametric method, which involves the concatenation of statistical models of the speech units [12] and is mostly implemented by machine learning techniques. It is "parametric" because it describes speech through parameters, such as the values for fundamental frequency (F0), it is "statistical" because parameters are described statistically, capturing the distribution of parameter values found in the training data. In this way, starting from the finite amount of training data available, any required sequence of context-dependent models can be retrieved to generate human-like speech [23].

Each module which composes a synthesiser may be implemented by rules or by machine learning techniques. The first approach is based on rules written by the developer, which apply systematically when patterns are matched, namely with the occurrence of given conditions. The latter consists in the development of systems able to process large amounts of data in order to extract and exploit information [41]. This is possible thanks to the availability of large corpora of linguistic data which are representative of the written and/or oral use and are necessary to train an algorithm. The training is the first step of machine learning, where a model which learns from the data is created. In the second step, the testing phase, such model is evaluated on the basis of how well the algorithm accomplishes a given task. In the third and last step, the model is applied to real-world data. Both rule-based and machine learning techniques feature advantages and drawbacks, therefore the choice between them is driven by factors including the specific application and the language which is implemented in the synthesis system.

The overall quality of a TTS system is heavily bound to the quality and quantity of the speech data. This is the reason why large speech corpora made up of recordings from single expert speakers are highly valuable. A huge source of such kind of data to freely draw from is LibriVox, a worldwide group of volunteers which create audiobooks and make them available on their web site [1].

[1] `https://librivox.org/`

The goals of our research are two. The first is to demonstrate the relevance of LibriVox as data source. It is adequate to be employed as corpus for two main reasons. Firstly, the material it contains is huge, diverse and readily available. Secondly, the production of a natural human-like voice takes advantage of the high cohesion of the read and recorded texts. This is highly convenient, compared to the relatively short, untied and incoherent collection of sentences which are generally used to create digital speech. Being such sentences chosen in order to cover all the phonetic contexts of a language, they tend to be complex, contrived and different from everyday speech. Since the synthesis process is heavily data-driven, the unnaturalness of the base-text necessarily affects the resulting synthetic speech. In support of our thesis we explored and handled LibriVox by asking for permissions, designing both objective and subjective criteria for the selection of data and elaborating such data in order to make them available for further software processing.

The second purpose of our research is to arrange the generation of a feminine and a masculine synthetic voice for Spanish and Portuguese languages, by improving and making prototypes of some NLP software modules for their implementation in our TTS system.

Thus, our research work consisted in five main steps. First, we chose two audiobooks from LibriVox as corpus speech for the implementation of a feminine and a masculine Spanish voice. The selection was made according to subjective factors such as the voice quality, the absence of noise and reverberation and the level of the signal, and objective factors such as the presence of standard Castilian accent and of several file specifications.

Secondly, we elaborated the above-said speech corpora. We wrote a shell script for automatise the downloading of high quality audio files from LibriVox, then, by means of a software, we created files containing segmented and aligned audiobook chapters, with adequate corrections of text errors and mismatches between audio waveforms and text.

Thirdly, we searched for corpora for the training of machine learning NLP modules for Portuguese POS tagging.

The four step of our work consisted in training and testing the POS tagger for Spanish.

The last step was getting hold of a NLP task which was based on rules rather than machine learning. It included the grapheme-to-phoneme (G2P) conversion module, syllabification and stress assigment for Spanish and it was handled by creating adequate shell script prototypes.

Therefore our work is organised as follows. Chapter 2 reviews, from the general to the particular, NLP, dwelling on its area of research, history

and applications, moving towards the description of text-to-speech synthesis from several perspectives, such as its criteria of evaluation, classification and implementations.

Chapter 3 deals with the steps of our research method, focusing on the software tools employed and the ones produced in order to automatise processes. The topics include the preparation of LibriVox audiobooks and their elaboration. Finally, the arrangement by means of shell scripting of POS tagging and phonemisation modules for Spanish and Portuguese is presented.

Chapter 4 and chapter 5 show, respectively, the results of our work and the analysis of data.

Chapter 5 outlines the conclusions and the intents for future work.

Chapter 2

Literature review

"Solving the natural language problem is as difficult as solving the AI problem."

[2]

2.1 Natural Language Processing (NLP)

The aim of this section is certainly far from the purpose to be exhaustive; it wants to give some brushstrokes concerning the area of research, history and applications of NLP, in order for text-to-speech synthesis to be placed in context, as a branch of NLP research.

2.1.1 Area of research

NLP is a field of computer science, artificial intelligence and linguistics whose focus is the interactions between humans and computers [2] and the development of tools and techniques to make computer systems understand and manipulate language to perform determined tasks (see Figure 2.1). Thus, NLP researchers aim to examine in depth how human beings understand and use language, so that there are three major problems emerging in the process of building language processing computer programs [9]:

1. the thought process
2. the representation and meaning of the linguistic input and output
3. the world knowledge

While humans produce and comprehend language through all its levels simultaneously, NLP analyses and represents language at one or more levels of linguistic analysis, not necessarily all of them. Nevertheless, the most explanatory method for presenting the functioning of a NLP system, is by means of the "levels of language" approach. It is also referred to as "the synchronic model", in order to distinguish it from the earlier "sequential model". A NLP system to be capable utilises as more levels as possible [25], thus it comprises the word level (involving phonetic, phonological, morphological, lexical and semantic level), the sentence level (syntax and prosody) and the context or extra-linguistic environment (pragmatics) [9]. Currently, NLP systems implement modules to accomplish mainly the lower levels of processing for several reasons. First, the specific application may not require interpretation at the higher levels. Secondly, the lower levels have been more researched and implemented. Thirdly, the lower levels deal with smaller units of analysis, e.g. morphemes, words, and sentences, which are rule-governed, versus the higher levels of language processing which deal with texts and world knowledge [25].

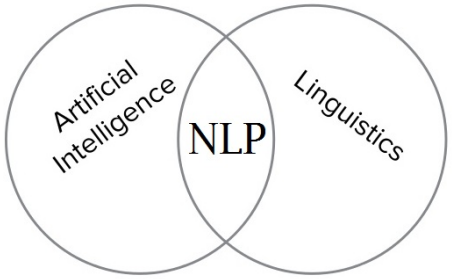

Figure 2.1: Natural Language Processing is placed where Language and Computer Sciences overlap.

2.1.2 History

Machine Translation (MT) was the early attempt of integration between natural language and computer-based applications, dating in the late 1940s with projects based on expertise in breaking enemy codes during World War II. The enthusiasm due to the developments of the syntactic theory of language and parsing algorithms [25] led to the birth of NLP, whose history conventionally starts in 1950, with the publication of "Computing machinery and intelligence" by Alan Turing [2]. Turing proposes to consider the question: "Can machines think?" [38], which, assuming the interdependence between thought and language, lays the groundwork to the conceive of NLP

as a subfield of artificial intelligence. In the 1960s early natural language systems were developed. Researchers at the MIT AI Laboratory developed programs capable of intelligent behaviour in artificially simple situations, known as *micro-worlds*. An example of micro-world approach is SHRDLU, a natural language understanding computer program developed by Terry Winograd in 1968-1970, whose micro-world consisted of coloured blocks of various shapes and sizes arrayed on a flat surface. SHRDLU could respond to commands typed in English by answering or planning out a sequence of actions, for example: "Had you touched any pyramid before you put one on the green block?" "Yes, the green one" [11]. In 1972 PARRY, an early example of a chatterbot, was implemented by psychiatrist Kenneth Colby. It was an attempt to understand the paranoiac illness, in fact the robot was built in order to seem hostile, defensive and unpredictable [10]. In 1975 Robert Schank and his group at Yale applied the micro-world approach to language processing. His program was SAM and its task consisted in answering questions about simple stories concerning stereotypical situations, such as dining in a restaurant, by inferring implicit information in the stories. For example, when asked "What did John order?", SAM replied "John ordered lasagne". Three years later the same group built the program FRUMP, which was able to produce summaries in three languages of wire-service news reports.

Programs like these were criticised because disembodied and unable to actually understand and learn language and situations and they were reduced to machines with the ability to produce strings of symbols in response to other strings of symbols [11]. Nevertheless, they represented the first attempts of language processing in the field of artificial intelligence.

2.1.3 Applications

In the last thirty years the NLP applications grew rapidly thanks to the increased availability of large amounts of electronic texts and of computers with increased speed and memory [25]. We give here a picture of the main tasks and relative fields of research in which natural, both textual and spoken, language processing is employed [2] [9].

Machine translation. It is the automatic translation of text or speech from one natural language to another. Speech translation recognises incoming speech from the source language, translates the text and then synthesises it into audible speech in the target language [34].

Automatic abstracting and summarisation. On the basis of linguistic and/or statistical criteria, parts of the text are automatically selected

to generate abstracts or summaries of texts. They are used in the web and digital library environments.

Automatic Speech Recognition. It consists in the recognition of speech and conversion into text. The scientist Hervé Boulard once said, pressing an open tube of toothpaste (whose brand was Signal): "Speech recognition is the art of pushing the toothpaste back into the tube" [12]. Examples of ASR applications are Speaker Authentication for voice access systems, to secure access to facilities or resources such as buildings or bank accounts, and Access Control, which serves to check if the speaker belongs to a particular group of people [34].

Named Entity Recognition. This system describes a stream of text, determines which items in the text relate to given names (such as people or places) and the type of reference.

Sentiment Analysis. Also known as opinion mining, it is often used for marketing purposes, costumer service or healthcare to determine the attitude of people with respect to a topic or a product.

Voice Transformation. It consists in the changing of some aspects of the speech signal while preserving linguistic information. A subfield of VT is Voice Conversion, which converts the source voice into a target voice, in such a way that the output is perceived as uttered by a different speaker.

Optical Character Recognition. Given an image representing printed or handwritten text, it determines the corresponding or related text. It is widely used to decode passport documents, invoices, receipts and bank statements.

Speech Synthesis. It is a system which produces human-like speech. The input may be text (text-to-speech system) or a symbolic linguistic representation, such as phonetic transcriptions. The output is intelligible speech. It uses recorded authentic speech stored in a database and produces any required speech (see 2.2).

2.2 Text-to-speech synthesis

The purpose of this section is to give a framework to the heterogeneous and quickly-moving field of text-to-speech synthesis, which involves engineering, computer science, multiple areas of linguistics, mathematical psychology and

acoustics [21], in order to provide a general and comparative overview about this system which produces artificially human speech.

2.2.1 Criteria of evaluation

For a Text-to-Speech (TTS) system to be efficient it will feature intelligibility of the output, namely the comprehensibility and clarity of the speech chain, and, in the second place, naturalness or "believability" [6], which consists in the closeness of the output of speech synthesis system to authentic human speech [1]. In order to gain a better understanding of intelligibility and naturalness of the synthetic speech, a typical comparison with the synthesis of human faces can be made. A cartoon-like drawing may be unanimously recognised as a human face even being stylised (intelligibility), but it is certainly more arduous to paint a realistic human-like face which could be mistaken for a photograph of a real human being (naturalness). This example helps to highlight how even substantial alterations, both in the synthesis of human face and speech, may be done without affecting intelligibility, whereas even slightly changes will immediately make the output look artificial. Delivering intelligibility and naturalness has been the focus of speech synthesis research for the past 40 years [12]. There are some other criteria of evaluation which are increasingly considered the cutting edge and which, together with intelligibility and naturalness, contribute to affect the quality of a text-to-speech system in an interdependent manner. Among such criteria we include expressivity, likeability and speaker similarity. Expressivity relates to the ability to change the speaking style. For example, the IBM Expressive TTS Engine offers five different speaking styles: neutral declarative, conveying good news, conveying bad news, asking a question and contrastive emphasis. Furthermore, this TTS system is able to generate para-linguistic events, such as breaths, sighs and pauses [14]. Likeability concerns the appeal perceived by the listener and fitted to the image or projected image of the company or product whose voice is to become [6]. When the output from a speech synthesiser is listened for the first time, it may sound pleasant, but during longer listening period, single clicks or other weak points in the system may make the so-called "annoying effect" arise. Thus the feedback from long-term users is sometimes essential. Speaker similarity concerns the fact that the speech signal conveys information about the speaker's anatomy, physiology, linguistic experience and mental state, so that speaker's individuality is heavily expressed through speech, at all levels of the message generation process [40].

Criteria of evaluation	
Criterion	Description
Intelligibility	Comprehensibility of the speech chain
Naturalness	Believability or closeness to human speech
Expressivity	Ability to change speaking style for conveying emotions
Likeability	Pleasantness versus "annoying effect"
Speaker similarity	Expression of individuality

Table 2.1: Description of the criteria to evaluate the overall quality of a TTS system.

2.2.2 Methods for the evaluation

> "In a perfect world, there would be a simple metric that we could apply to any TTS system to reveal some sort of magic number that describes its overall quality. That metric does not exist." [1]

There are two main argues to explain the impossibility to achieve an objective and absolute evaluation for TTS systems.

1. The performance of a TTS system cannot be assessed in isolation, but rather within a specific criterion, application and context. For example it is difficult to completely isolate the several criteria, such as naturalness from speaker similarity, when evaluating a system. In addition, one could state that naturalness of the output is an unavoidable factor to implement in a synthesiser, but, for example, in the case of a TTS system whose aim is to assist a blind, it could be advantageous for the user to be able to easily discern between the synthetic voice and an authentic human voice (see 2.2.3). Moreover, a likeable synthetic voice which may be appropriate for a system intended to sell computer games to teenagers, probably won't be the best for a banking application.

2. When assessing a TTS there are subjective factors coming into play, which are independent of TTS quality. For example, the evaluation of intelligibility may be affected by the accent of the listener, or by many other individual factors, such as gender, age or level of education [1].

For these reasons, when assessing a TTS system the researcher has to control all of the factors which could affect the results of the evaluation. We briefly present some examples of experimental conditions to consider when designing an evaluation test, after having focused on the specific application

of the system to be assessed, the final users and the criterion which is going to be evaluated.

1. **Task and test type**

   Phonetic task. Listeners are asked to identify a specific sound within a word. It is suitable to assess intelligibility.

   SUS test. Subjects are asked to transcribe Semantically Unpredictable Sentences (SUS), in order to minimise the possibility to derive information from any source but the speech signal itself.

   Mean Opinion Score (MOS). Listeners are asked to rate the speech quality of different systems, usually synthesising the same sentences, with multiple choice. It is suitable to assess naturalness or speaker similarity [1].

   ABX. Consists in comparing two versions to a target or reference and indicating the closer one [6].

2. **Experimental subjects**

   Recruitment. Each way of recruitment features advantages and drawbacks. For example, crowdsourcing is a valuable tool because it allows to find a large amount of people in short time. Moreover it is possible to set up filters in order to reach only a selected part of population. Its drawback is the possible lack of motivation of the participants.

   Features. There are many features to be controlled when choosing the subjects, such as gender, age, level of education, mother tongue, accent, level of expertise in speech technology, familiarity with the background of the experiment.

3. **Text and speech corpus.** Being the output of a speech synthesiser heavily bound to the speech database, the selection of the corpus data is an essential factor to be controlled. For example, in order to better understand and compare research techniques in building speech synthesisers, the Blizzard Challenge, namely a challenge devised to compare TTS systems, is made on a common speech corpus. The challenge consists in taking the released common speech database, building a synthetic voice from the data and synthesise a predefined set of test sentences. Then, the sentences from each synthesiser will be evaluated through listening tests [3].

2.2.3 Fields of application

Despite being interdependent, the criteria which affect a speech synthesiser quality (see 2.2.1) have more or less priority both on the basis of the area of application of the synthesiser and according to the kind of the required system, which can be hardware or software based. Diachronically, first commercial speech synthesis systems were mostly hardware based. Since computers have become more powerful, most synthesisers today are software based, which are easier to configure and to update, and usually they are less expensive than hardware based. However, the choice between a hardware device or a software program for speech synthesis depends on the type of application needed; for example an hardware synthesiser may be the best solution when a portable system is required.

Each module involved in the process of synthesis contributes to degrade the speech signal, thus if to be synthesised is a limited vocabulary, it is advantageous to store natural speech samples of large units, as words or sentences, and synthesise the output message by concatenating such units. This kind of system is used in applications as railway-station announcement, interactive guides, car navigation directions and weather forecasts [18] [6]. For all the applications requiring unlimited vocabulary, a TTS system is needed.

Applications for the blind and visually impaired. This field includes reading and communication aids for the blind. Before speech synthesis, specific audiobooks were used for reading purposes, and to get information from computers special bliss symbol keyboards were employed, whereas currently a voice output system (or reading machine) is provided. The first commercial TTS applications were introduced in the late 70s and, because of their expensiveness, they was mostly used in public libraries. Current systems are mostly software based, thus it is easy to equip any computer environment with a reading machine with tolerable expenses. The most crucial factor with reading machines is intelligibility; as regards naturalness, in some cases it may be desirable for the listener to be able to identify that the speech is coming from a machine.

Applications for the deaf and vocally impaired. Deaf or people which are losing their capability to speak obtained the opportunity to communicate with people who do not understand the sign language. Furthermore, they may feel frustrated by the inability to convey their emotions and personality, therefore expressivity and similarity to the

speaker voice may be implemented in communication devices (see 2.2.1).

Educational field. A computer with speech synthesiser can be used in many educational situations, for example language learning and other interactive educational tasks. Children affected with dislexia, with proper computer software, may avail themselves of unsupervised training.

Telecommunication and multimedia. The field of multimedia is the newest speech synthesis application. It has been used for decades in telephone enquiry systems, and nowadays TTS systems have improved their quality to such extent that e-mails, sms and chat messages may be listened in mobile phones. Man-machine dialogue system is used in applications as commercial assistants and mobile-phone virtual assistants. An emerging service is the smart room environment, in which visual and acoustic technologies are combined to recognise and interpret people's actions and interactions inside the room. It is a useful tool for work, educational, didactic and ludic purposes [1].

2.2.4 Classification

2.2.4.1 Rule-based and Machine Learning (ML) approaches

Speech synthesis systems can be firstly distinguished between technologies which generate digital speech from authentic recordings of natural speech by using arbitrary rules and by machine learning techniques.

Rule-based approach. In the 1970s TTS tools have followed the development of phonological theory, especially the work on generative phonology by Chomsky and Halle [8], leading to a synthesis system based on context-sensitive rewrite rules [7]. Chomsky and Halle generative theory tries to account for the phonologic competence of speakers by underlying phonological rules. A phonological rule is a process which applies systematically with the occurrence of specific conditions, namely the environment which triggers the change. In other words, when the input A is in context X, it realises as B; for example in Spanish whenever the plosive phonemes /b/, /d/ and /g/ are in intervocalic context, they are articulated as their equivalent fricative form, respectively [β], [ð] and [ɣ]. Phonological rules are represented using formal rewrite

[1] `http://research.spa.aalto.fi/publications/theses/lemmetty_mst/chap6.html`

rules in the most general way possible. Rule-based technique operates this way: the researcher creates rules, then the computer builds a rule engine whose behaviour follows the lines of: search for a rule that matches the pattern of data, execute that rule, go to top [2].

Among the benefits provided by handcrafted rule-based technique there are analytic controllability [30] and repeatabilty. On the other hand, being arbitrary, the implementations of such system may vary depending on the developer's inclination [7]. Moreover, people are often prone to making mistakes.

Machine learning approach. Since 2002, speech synthesis has been revolutionised by machine learning or data-driven techniques for natural language processing [30]. They consist in automatised systems able to process large amounts of data in order to extract and exploit useful information [41]. This is possible thanks to the manual labelling of a part of data in the speech corpus [5] which serves for the training; then the tool is able to automatically label data which were not previously seen [41]. This technology launched key elements for designing multilingual and embedded systems, thanks to the features of genericity and scalability, respectively the quality of being generic and easily modified and adapted to even wide changes in the input data [30]. The quality of this method is heavily bound to the speech corpus, and this unavoidably leads to its drawbacks. In fact the database requires a large corpus of natural speech from one speaker, and it has to be designed, developed and balanced in order to ensure coverage and diversity of the units or contexts that are minimally sufficient to describe the perceptually relevant variation in acoustics and prosody. Even the speech style needs to be controlled, because the style of the corpus speech determines the style of the synthetic speech [31] [5].

2.2.4.2 Front-end/back-end and integrated models

The second proposed classification relates to the model design. Most synthesisers feature an internal division in two macro-modules, front-end and back-end (see Figure 2.2), but there are models which combine these modules into one integrated model.

Front-end and back-end model. The conversion from written language (the input) to spoken language (the output) is divided into two main

[2] `https://msdn.microsoft.com/en-us/library/aa480020.aspx#aj1rules_topic10`

steps. The first one, the front-end or Natural Language Processor (NLP), converts the input into linguistic specification. This occurs first by means of a language-processing stage producing a machine-readable representation of the input including word sequences, pronunciations and relations among them. Then a prosody-processing stage is involved, converting the abstract text-based representation into a sequence of parameters. The second step, the back-end or Digital Signal Processor (DSP), converts the linguistic specification (or parameter-based representation) into a sequence of waveform segments which are sent to an audio device to produce the speech sound [6]. The first step is largely language-specific, while the latter is more independent of the language, except for the data it contains or it is trained on [23].

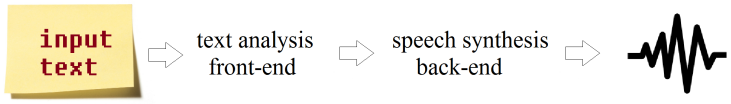

Figure 2.2: General architecture of a TTS synthesiser.

Integrated model. Standard TTS systems consist of two independent modules: text analysis and speech synthesis (i.e., front-end and back-end), where the first is trained using text corpora and the latter with a labelled speech database. Instead, in integrated models, linguistic and acoustic models are trained and performed simultaneously, thanks to the optimisation and integration of model parameters. The main advantage of this model, as proposed by Oura et al. [15], is that the labelling phase of phrases and prosody is not required for neither linguistic nor acoustic model training, because labels are regarded as latent variables in the model.

A specific application of such integrated model approach is deep generative model, the one implemented in WaveNet synthesiser [39]. It is a neural network model which generates speech miming human voice with high naturalness by directly modelling the raw waveform of the audio signal, one sample at a time. This technique is inspired by recent advances in neural autoregressive generative models, which model complex distributions such as images and texts. In addition to naturalness, the advantage of such method is its flexibility, in fact a single model manages to generate different voices.

2.2.4.3 Concatenative and generative systems

The distinction between concatenative and generative TTS systems lies mainly in the nature and dimensions of the considered speech segments.

Concatenative synthesis. Concatenative synthesis is a system which concatenates segments of recorded speech or statistical models of such speech segments. The nature and duration of segments, which are stored in a database, is bound to the implementations. There are three main types of concatenative systems.

1. The first is synthesis based on a fixed inventory, whose speech database is made up of all the transitions between one sound and the following which occur in a given language (see 2.2.5.3).
2. The second is based on a variable inventory, which draws from a larger database containing segments (or units) of changeable length (see 2.2.5.4).

   These two systems try to deliver intelligibility and naturalness of speech by automatically sampling speech chunks and gluing them together, assuming that it does not imply a mere playback of a sequence of pre-recorded words or phonemes. To give an idea of the complexity of this method we may take into account that speech chunks require to embody natural coarticulation, jitter, shimmer and inharmonicity. Coarticulation is the result of the continuous movement of each articulator form the realisation of one phoneme to the next one, therefore transients in speech are crucial more than stable segments (e.g., the centre of a vowel); and even vocalic segments feature aperiodicity, exhibiting small variations in frequency and amplitude (respectively jitter and shimmer). To face the issue of inharmonicity, namely the presence of noise-like components, it does not suffice to add noise to the speech signal because inharmonicity is not pure randomness.
3. The third type is statistical parametric synthesis, which is based on the Hidden Markov Model (HMM) and concatenates parameters rather than speech units. This technique might be described as generating the average of some sets of similarly sounding speech segments [17](see 2.2.5.5). There can also be hybrid approaches.

Generative synthesis. Generative or granular synthesis works by generating a rapid succession of speech bursts or grains, whose duration

Classification			
Type	Description	Advantages	Drawbacks
Rule-based	Arbitrary hand-written rules	Controllability, repeatability	Exposed to subjectivity and mistakes
Machine learning	Automatic processing and extraction of data	Genericity, scalability	Corpus-dependency
Front-end/back-end	Independent modules for text analysis and speech synthesis		
Integrated model	Linguistic and acoustic modules are integrated	Optimisation of parameters; no labelling required	
Concatenative synthesis	Concatenation of speech segments or statistical parameters		
Generative synthesis	Succession of speech bursts (1-50 ms)		

Table 2.2: Summary of the classification of TTS systems.

may be of 1 to 50 milliseconds. Despite the simplicity of a granular instrument, its operation requires the control of large sets of parameters. A sound designer is therefore needed to provide global grain parameters and to control them; otherwise it would be impracticable to specify thousands of parameters manually. The functioning of this technique can be compared to the one of motion pictures: the more fine-grained the sequence of movements/sounds is, the more the output emerges continuous and natural. There are several approaches to generative synthesis. Granular sampling approach extracts small portions of a sampled sound and applies an envelope to them. The envelope phase is crucial because it prevents the glitches that would otherwise be produced by possible discontinuities between the grains [26].

2.2.5 Applied implementations

When designing a speech synthesiser, the strengths and weaknesses of each method are to be considered and linked to the actual application of the synthesiser. There are standard implementations which typically involve deter-

ministic choices; nevertheless, each designer can arbitrary choose to deviate from the standard implementations, and also designing hybrid systems.

2.2.5.1 Formant synthesis

Formant synthesis typically features a rule-based, integrated and generative model. The theory of formant synthesis arises with the first attempt to model the acoustics of the speech waveform, starting from the assumption about the inter-relationship of articulatory models, acoustic manifestations and perception of spoken communication (each of which is analysed by a different branch of phonetics). This TTS system uses a terminal analogue as a device for generating speech sound. The internal structure follows the source-filter model [7], which assumes that the human vocal folds are a source of sound energy and the vocal tract is a frequency filter altering the timbre of the vocal fold sound thanks to the interaction of the sound waves in it [24]. It involves a buzz generator for voiced sounds, a hiss generator for unvoiced sounds and resonators to simulate the resonances in the vocal tract. The advantages of this system are the possibility to automatically set the formant amplitudes and to model both linguistic and extralinguistic processes. Its drawback is due to the unsophistication of the model, which make difficult the matching between natural and artificial spectra [7].

2.2.5.2 Articulatory synthesis

The first articulatory synthesiser was developed in the 1970s at Haskins Laboratories on the basis of the vocal tract model developed at Bell Laboratories in the 1960s. As formant synthesis, it is an example of generative synthesis by rules, with the difference that articulatory system directly modifies the speech spectrum. It, in fact, generates acoustic signals on the basis of articulatory-acoustic models of speech production, including yielding and vibrating walls, noise generation in turbolent conditions and time-varying lengths of the speech apparatus regions [4].

2.2.5.3 Diphone synthesis

Diphone synthesis employs a concatenative system based on a fixed inventory, from which speech chunks are picked up and glued. Such inventory is made up of basic units named *diphones*. A diphone is a speech segment which starts in the middle of the stablest part of a phoneme and ends in the one of the following phoneme.

The main disadvantage of this approach consists in its limited-sized unit inventory. In fact, if a language has N phonemes, it has about N^2 diphones (being them combinations between two phonemes), or slightly fewer because not all the diphones are possible in a natural language. This leads to a typical diphone database size of 1500 units (about 3 minutes of speech). Since speech units are originated from different words or phonetic contexts, their concatenation may produce audible clicks due to mismatches in phase, pitch or spectral envelopment between successive units. This is hard to cope even though the corpus is read by the same speaker, a professional reader which avoids speaking-styles and pitch variations [12].

2.2.5.4 Unit selection synthesis

Unit selection (see Table 2.3) provides an example of rule-based concatenative synthesis including front-end and back-end. This method concatenates waveforms of units selected from a single-speaker database, but it use a larger units inventory than the one used for concatenative synthesis based on a fixed inventory [12]. A larger set of units which is diverse in terms of prosody and spectral features takes to the achieving of a more natural synthetic speech. The selection of units takes place in order to minimise the acoustic distortion between the units and the target spectrum, both in prosodic and phonetic terms [20]. As in diphone synthesis, its task is reduced to retrieval rather than replication, thanks to the employment of original waveform segments, which is possible without the need for a perceptually damaging signal processing [5]. Since acoustic modifications tend to reduce the speech quality, the reduction of the extent of required signal processing to possibly correct the acoustic features of the units, lead to an increasing of the quality of the speech output [20].

The two main challenges to face are one related to the mechanism of selecting the units (target cost), the other to their combination (concatenation cost). To reduce target cost, the clustering method is used, where similar speech units are put in one cluster, thereby all units are available at the necessary situation. To reduce concatenation cost, the largest sequence of originally consecutive units is selected, so that fewer connection points lead to a more continuous and, thus, natural synthesised speech [32]. For example, if a considerable part of a target sentence is available in the speech corpus, the corresponding consecutive units are selected. Using originally consecutive units avoids the presence of concatenation points, setting the concatenation cost to zero [12].

Concatenative systems based on stored exemplars (i.e., diphone and unit-

selection synthesisers) have improved the quality of synthetic speech, producing a high degree of naturalness. These systems represent an enhancement compared to the previous formant and articulatory synthesis systems, but the lack of an explicit and sophisticated speech model limits their usefulness to the restricted task of neutral speech, being inadequate in conveying emotional speech. Another critical aspect is the lack of flexibility. Being the nature of the database a crucial factor for the quality of the synthetic speech, the system should use a database which is sufficiently large to contain all the speech styles and a sophisticated speech model within a specific procedure for speech unit selection which identifies the speech styles in the database [13].

2.2.5.5 Statistical parametric synthesis

Unlike exemplar-based concatenation systems (i.e., diphone and unit-selection synthesisers), the statistical parametric is typically implemented by machine learning, and it is a model-based system [23] involving the concatenation of statistical models of speech units [12] (see Table 2.3). It is "parametric" because it describes speech through parameters (rather than stored exemplars), such as the values for fundamental frequency (F0) and the spectral envelope of the aperiodic excitacion. It is "statistical" because parameters are described statistically, capturing the distribution of parameter values found in the training data. By sharing parameters with similar models it is possible to create a model for any required linguistic specification. In this way, starting from the finite amount of training data available, any needed sequence of context-dependent models can be retrieved to generate speech. There are two ways for the models to be indexed by linguistic specification. The first is manual labelling, but it is often too expensive and inconvenient. The other is the prediction of linguistic specification based on the text corresponding to the speech corpus. To improve the accuracy of this kind of labelling, techniques based on forced alignment methods (which are borrowed from speech recognition) can be applied.

There is not a stored speech corpus, but rather a model fitted to the speech corpus during the training phase. Such model is constructed in terms of individual speech units, namely context-dependent phonemes [23]. The first and most important advantage is high flexibility, thanks to the possibility to modify the characteristics of speaker voice, as speech style and emotional state, by changing the parameter models. The second advantage is phonetic space coverage, which allows the production of continuous speech. Another advantage is the support of multilinguality, due to the small

amount of required training data, by determining language-specific factors (e.g., contextual factors) and collecting appropriate speech data [32] (see 2.2.7.2). The disadvantage of statistical parametric speech synthesis concerns three factors which degrade speech quality. The first is the presence of buzz in the synthesised speech due to the vocoder. The second consists in the fact that since speech parameters are produced from the acoustic models, which in turn are influenced by the speech data, their accuracy affects the quality of the synthesised speech. The third concerns over-smoothing, namely the characteristics of speech parameters are removed in the modelling part and cannot be retrieved in the synthesis phase because the synthesis algorithm does not explicitly include a recovery mechanism, producing a muffled speech sound [17].

2.2.5.6 Hybrid approaches

The essential difference between unit-selection and statistical parametric approaches is that in the first each cluster is represented by multi-templates of speech units and in the latter each cluster is represented by the statistics of the cluster. As a consequence hybrid approaches between the two methods are possible. As examples, without claiming to be exhaustive, some of these approaches use parameters as "targets" for unit-selection synthesis. Another type of hybrid uses statistical models to smooth segment sequences obtained by unit selection. Yet another is the mixing of natural and generated speech units, where if there are not enough candidate units for the required diphone, a segmental waveform is synthesised, and, as a result, the waveform of an utterance is synthesised by concatenating the sequences of segmental waveforms obtained from unit selection or parameter generation. The prospect for the future is to fuse unit-selection and statistical parametric methods into an optimal complementary system of corpus-based speech synthesis that can solve the respective drawbacks while retaining the advantages of both of them [17].

2.2.6 Metrics for the assessment

A metric is a statistical method which measures the proportion of positive and negative results, concerning, for example, the extent of a certain quality in a system or the distribution of samples in a set. In order to assess an algorithm several metrics can be used, namely recall, precision and accuracy, according to the required task (e.g. pattern recognition, information retrieval, word alignment) and the expected results.

Applied implementations	
Unit Selection	Statistical Parametric
Mainly implemented by rules	Mainly implemented by machine learning
Exemplar-based system	Model-based system
Selection of speech units from the database	Capture of statistical distribution of parameters in the training data
Lack of flexibility	Flexibility
High degree of naturalness	High phonetic space coverage (continuous speech)
Only neutral speech	Supports multilinguality
Target cost/combination cost	Buzzy and muffled speech sound

Table 2.3: Comparison between the two main TTS implementations

In order to understand how such metrics work, we assume a theoretical dichotomous classification task, where we have a binary set of elements A and B. The required task is the retrieval of all the A elements (relevant elements) and the refuse of the B elements. We suppose the algorithm retrieves some of the A elements (True Positives) and some B (False Positives). The A elements which are not retrieved are called False Negatives and the not retrieved B elements True Negatives.

Recall. Recall or sensitivity is the ratio between the correctly retrieved TP and the total amount of relevant elements, or:

$\frac{TP}{TP+FN}$

Precision. Precision or confidence is the ratio between TP and the total amount of retrieved elements, or:

$\frac{TP}{TP+FP}$

Both recall and precision focus only on the positive elements, without taking into account how the model handles negative cases.

It is possible to interchange positive and negative in order to predict the opposite case. Thus, inverse recall (or True Negative Rate) is the proportion of correctly predicted negative cases. Conversely, inverse precision (or True Negative Accuracy) is the proportion of predicted negative cases which are indeed real negatives.

Accuracy. Accuracy is the ratio between the correctly selected elements (both TP and TN) and the total amount of the cases examined, thus it explicitly takes into account the classification of negatives. It is expressible as both a weighted average of recall and inverse recall, and as a weighted average of precision and inverse precision [29].

In the recent years the use of ROC (receiver operating characteristics) graphs in machine learning has increased, thanks to the realisation that precision, recall and accuracy are often poor metrics for measuring performance [16]. ROC graph is a technique for visualising, organising and selecting classifiers (namely algorithms for statistical classification) based on their performance, in order to compare them and choose the best one. The ROC curves come from the integration between TPR and FPR. Thus, the best classifier is the one which is closest to the minimum score for False Positive Rate and the maximum score for True Positive Rate [29], so that:

- Perfect classifier $\rightarrow FPR = 0\%,\ TPR = 100\%$
- Worst classifier $\rightarrow FPR = 100\%,\ TPR = 0\%$
- Random classifiers $\rightarrow FPR = TPR$

ROC curves have been used in all kinds of evaluation practices for long, when Spackman [35] first adopted them in machine learning, demonstrating their value in evaluating and comparing algorithms [16].

2.2.7 Challenges

2.2.7.1 Generalisation: the challenge of unseen contexts

The key problem in speech synthesis is generating utterances that we do not have natural recordings of. Utterances must be built starting from smaller units (phonemes-in-context). For a better understanding of the relevance of the problem there are two facts to consider, which converge into the statistical phenomenon known as the "Large Number of Rare Events" (LNRE).

1. Even a large speech corpus has a very sparse coverage of the language, so most unit types have no examples in the corpus and it does not include all of the commonly-occurring contexts. This makes it impossible to model rare or unobserved types because there are not enough examples to learn anything (see 2.2.4.1).

2. Since the context spans the whole utterance, a unit type cannot occur twice in the same context, making them rare in the corpus.

As a consequence we have to generalise from a limited set of contexts observable in the training data, to the large amount of unseen contexts we inevitably encounter when synthesising novel utterances. The solution for the problem of generalising from limited training data is the *model complexity control*. We could reduce the number of types to label the data, namely reducing the contextual factors considered depending on their effects on the realisation of the phonemes, but we do not know *a priori* which context factors are significant. Moreover, the relevance of a context will vary depending on its peculiar interaction with complex combinations. A better and more elegant solution is to maintain a large number of types to label the data and to control the complexity of the model rather than the complexity of the labelling. The model complexity control used in HMM-based systems involves sharing parameters among similar models. In this way it is possible to control for the data available the number of free parameters and of parameters for which we have only a few or no examples at all. Being the complexity of the model (i.e., how much or how little parameter tying there is) data-driven, the amount of context-dependency may vary from model to model, for example more data will lead to a more complex model. In order to choose which contextual distinctions are worth making and when we should use the same model or separate models for two different contexts, a widely used technique involves clustering together similar models for best fitting the model to the training data. After the models have been clustered, the number of models is lower than the number of distinct contexts. A larger training data set allows us to use a larger amount of models and makes more fine-grained distinctions [23].

2.2.7.2 Multilinguality

Since the early 1980s TTS systems have been developed to be multilingual, meaning that they allow to synthesising several languages. Problems and challenges in the different modules are heavily language-oriented. For example:

- in English there are many names and verbs which are homographs (i.e., they have the same written form) but they differ in the part of speech (name versus verb), the role played in the sentence and the stress assignment (e.g., "a rébel" versus "to rebél");
- Italian has many polimorphemic homographs containing clitics, which share the same written form, root and part of speech but differ in the stress assignment and number. For example, "ricordatelo" (meaning

the second person of the imperative "remember it") may be referred to the second person singular ("ricórdatelo") or plural ("ricordátelo").

Thus, it is fundamental for such systems to have separated language-dependent and language-independent modules and sources. A good devised architecture makes it easy for a system which is already operational with a language X, to add a new language Y. Concerning multilinguality, the ultimate goal is to provide a single set of programs which can load language-specific tables and therefore function as a synthesiser for multiple languages, considered the crucial fact that any module involved in the process of synthesising speech contributes to the overall degradation of the synthetic speech signal [18]. Once language-specific tables and parameters are appropriately set up, some of the language-specific modules for the TTS system of language X can simply be taken and used in the TTS pipeline for the language Y. Modules which are already language-independent are kept as they are. However, it has been developed a set of "vanilla" (i.e. default, minimal) modules in order to provide for the lack of the information to construct the necessary tables to drive all the modules, for example in the early phases of the development of a system for a new language [27]

2.3 MaryTTS/Mivoq synthesiser

This section presents Mivoq synthesiser, which is a modified version of the German MaryTTS system (see Figure 2.4) [33]. Mivoq synthesiser is implemented with statistical parametric method and machine learning techniques, although some of the modules it contains are implemented by rules (see Figure 2.3). For convenience, from now on we will refer just to Mivoq synthesiser, even if the two systems share most features. Their main characteristics are made explicit below, and then the single modules which constitute the front-end are featured.

2.3.1 Characteristics

2.3.1.1 A modular architecture

Mivoq synthesiser is a flexible system, whose flexibility is given by its modular architecture. This entails a step-by-step processing, which allows the access to partial processing results, so that each intermediate process cannot only be displayed but it can be modified by the user. Thereby the effects of a precise piece of information on the output of a given processing step can be interactively explored [33]. This modular design allows arbitrary

Figure 2.3: Mivoq web client interface. It is possible to select language, voice, audio effects and to display the output of the single modules.

system architectures, so that each module needs certain data type as input and others as output, so that two modules can be connected just having the output type of one module identical to the input of the other. Thus, such modular pipeline design facilitates revision and replacement of modules and the insertion of tools into the pipeline in order to modify parameters [27]. By means of such modular structure it is possible to extend the system to other languages. It suffices to define new data types and to provide modules connecting such data types (see 2.2.7.2). These advantages highlight the reasons why it is more convenient a modular approach than a monolithic one.

2.3.1.2 Markup language

Such architecture is made flexible also by the use of an internal XML language (eXtensible Markup Language), which codify the data representation format inside the TTS system. Beyond superficial syntactic differences, XML-based markup languages related to speech synthesis share the possibility for non-expert users to add information to a text to improve the quality of the speech. These languages are, in principle, independent of any particular TTS system: a system parses the markup language in its input

and converts it into an internal data representation format (often not an XML-based language). Through an operation called *serialisation*, at any stage of processing the document can be externally visible in the form of a textual XML document. The inverse process of *deserialisation* is also is possible [33]. These are useful tools for an expert to control all the modules and the intermediate processes of the system.

2.3.2 The front-end

As we have seen, the front-end converts the input text into a linguistic specification using a sequence of processes and intermediate representations. The particular ordering of the processes or modules is to some degree determined by the logical structure of the task, and to some degree arbitrary. The same is for the division of modules, which follows the assumption that each module faces a certain problem. Being the choice, at least partially, arbitrary, there can be systems where two modules collapse into one and vice-versa [27]. Here are, therefore, presented the mostly used modules in non-integrated models implemented with statistical parametric techniques, with the adequate underlines when the specific module or technique is a peculiar implementation of MaryTTS/Mivoq synthesiser and an example of Mivoq XML tag for each module, as summarised in Table 2.4.

2.3.2.1 Sentence splitting

The first aim of the front-end, at least in Mivoq synthesiser, is to segment the text into a list of sentences, enclosed by <s>...</s> XML tags. Whereas most linguistic units are largely influenced by their neighbours, sentences do not heavily interact with each other; thus they can be processed quite independently, making the process of splitting easier. Even so, it is essential for a high quality TTS system to put the sentence boundaries in the right place because it determines the sentence-final prosody, which is a phenomenon which listeners are very sensitive to.

It is therefore required to identify the written sentence boundaries. Since we cannot rely on any deep linguistic analysis to access the internal coherence, syntax, semantics and pragmatics of a sentence, we have to employ a more formal definition of sentence, merely based on writing. A sentence can hence be defined as "the words between a capital letter and a full stop". The challenge consists in discerning full stops and upper case characters delimiting a sentence and the ones serving other purposes. It is faced controlling the immediate context of these characters (see 2.3.2.2 for non-conventional meanings of dots and how disambiguation works) [36].

The following is an example of Mivoq XML tag for the sentence "Benvenuto nel mondo della sintesi vocale" (meaning "Welcome to the world of speech synthesis"):

```
<s> Benvenuto nel mondo della sintesi vocale.  </s>
```

2.3.2.2 Tokenisation

The second task of the front-end consists in extracting words from the sentences. The process needed for this step is *tokenisation*, meaning the segmentation of sentences into tokens. Such segmentation can be intuitively carried out using blank spaces and punctuation marks as delimiters, as they indicate the presence of word boundaries. But there are cases where this approach turns out to be simplistic and ambiguous, e.g. dealing with acronyms, abbreviations or some particular use of punctuation marks. As an example, the word "IBM" may be split into three tokens (<t> i </t>; <t> b </t>; <t> m </t>), but a form as "NATO" does not follow the same convention, being considered a single token [36]. In the case of acronyms, Mivoq synthesiser segments words as "IBM" and "NATO" in the same way for convenience, namely they both are considered single tokens. As well as acronyms, the meaning and the relative pronunciation of a dot can be ambiguous: it can function as a sentence-final full stop (which implies a speech pause), a decimal number delimiter, an abbreviation point; or it can form part of an e-mail address (where it is pronounced as /dɒt/, as in "yahoo.com"). In order for disambiguation the tokeniser uses a set of rules to label the different meanings (of a dot, in this case) [33]. Such rules are determined through corpus analysis on the basis of the surrounding context, collecting evidences of the roles fulfilled by the dots in the different contexts. For example a dot followed by whitespace and a lower-case character is not a sentence-final period.

A token is therefore a potential and provisional written form [36], enclosed by a <t>...</t> XML tag. Each token will be examined in turn in the *normalisation* phase, whereby any ambiguity will be resolved. The advantage of the tokenisation module is that it makes it easier to process the input of the front-end.

The following is an example of Mivoq XML tag for the token "Benvenuto" (meaning "Welcome"):

```
<t end="9" start="0">
Benvenuto
</t>
```

2.3.2.3 Normalisation

Text normalisation, or expansion, converts each token for which the spoken form does not entirely correspond to the written form, such as numbers and abbreviations, to its pronounceable form (e.g. a form like "Dr." is expanded into /dɒktə/). Here we need to discern between natural and non-natural language classes.

Non-natural language tokens. Non-natural language is any token for which a normalisation process is required, in order to decode such types into their underlying, pronounceable form. Therefore numbers, dates, times, urls or addresses may form part of non-natural language classes, which need to be converted into natural language forms. Concerning numbers, for example, there is not a one-to-one correspondence between a digit and its spoken form. There are different number types, whose pronunciation largely depends on the peculiar use. In fact the way a telephone number is spoken differs from the way a date or a time are. For example, in telephone numbers the digit "0" is pronounced as /əʊ/ and numbers are often spelled one by one, while a year is usually pronounced two digits at a time, but with some exceptions (e.g. "2010" is spoken /twɛnti tɛn/ and "2009" is /tuː θaʊzənd naɪn/). Moreover there can be ambiguity between cardinal and ordinal number: in a numbered list it could be more correct to pronounce the digit "1" as /fɜːst/, rather than /wʌn/.

Natural language tokens. Generally natural language text does not involve a normalisation process because of the clear correspondence between the written and the spoken form of a word. But there are natural language types which need to be normalised; three main types are outlined below.

1. The first type is abbreviation. Two main groups are distinguished. The first group includes the abbreviations which are spelled out (such as /juː-ɛs-eɪ/ for "USA"). They simply suffice spelling rules to be correctly pronounced. The second group abbreviations need expansion. They use an expansion table containing a graphemic and optionally a phonemic expansion. The graphemic expansion is needed for abbreviations which can be treated by the default letter-to-sound (L2S) conversion algorithm (see 2.3.2.7), for example "St." and "Dr." are the abbreviated forms of, respectively, /striːt/ and /dɒktə/, and they can be nor-

malised through graphemic expansion. For non-standard pronunciations (such as for foreign abbreviations) the phonemic expansion is required. For tokens identified as abbreviations for which no entry in the expansion table is found (when, for example, the writer introduces a new abbreviation which is not listed in the lexicon) two procedures can be applied. They are either spelled out, if they consist of less than six characters, or they are pronounced as normal words if they are longer.

2. The second type is mis-spelling. Some TTS systems manage to cope with mis-spelled text form of words, as human readers do. If the synthesiser has a spell correction system, it is able to discern a mis-spelling from a genuine unknown word (which will be labelled as "unknown", and added to the run-time lexicon), and to correct it. Otherwise the grapheme-to-phoneme (G2P) converter is applied to the mis-spelled word, as it occurs in Mivoq synthesiser.
3. The third particular type of natural language is alternative spelling. Where the differences are systematic it is possible to include alternative spelling in the lexicon. For example, in English language there are different and systematic conventions related to American versus non-American, so that surface forms as "organize" and "organise" encode the same underlying form of the suffix /aɪz/, which is present in the lexicon with the two alternative spellings [33] [36].

2.3.2.4 POS tagging

POS (Part-of-Speech) tagging operates a morphosyntactic disambiguation (or homograph disambiguation) and a semiotic classification (or POS assignment), assigning a part-of-speech (POS) to each token.

Morphosyntatic disambiguation. Ambiguity affects those tokens for which a surface form (i.e. the writing) matches with more underlying forms. A form can be ambiguous concerning its grammatical value, meaning and sound produced, but for the purposes of TTS synthesis all the ambiguities which do not affect phonetics can be ignored. For example the different meanings of the word "bank", namely the institution, a storage place (e.g. blood bank), a collection of data or a slope of land, do not involve differences in the word pronunciation /bæŋk/. Conversely, the word "record" entails two different meanings accord-

ing to the word interpretation as a noun (a rècord) or as a verb (to recòrd). In this instance a morphosyntactic disambiguation over the surface forms is required.

Semiotic classification. Semiotic classification means determining the semiotic class for each token.

Both homograph disambiguation and semiotic classification involve assigning a category to a token and both can be solved through similar algorithms. The process of assigning a label to each token is a classification task. Each label l_i is drawn from a pre-defined set of labels:

$$L = l_1, l_2, ...l_N$$

The choice depends on the analysis of a number of features, each of which has a set of legal values. For example the token "read" can encode a past or a present tense, with the respective pronunciations /ɹɛd/ and /ɹiːd/. In this case the label set comprises two labels such that:

$$L = l_1\texttt{=READ-PAST, } l_2\texttt{=READ-PRESENT}$$

In order to decide which is the correct label, the context, namely the other tokens and their interaction, is analysed. The factors which guide the choice can be formulated as features, like F_1=time token, F_2=is the preceding word "to"? The legal values of F_1 are "yesterday", "now" and "tomorrow"; the legal values of F_2 are "yes" and "no". This process can be defined as a function C and the classification algorithm (which is not accomplished in Mivoq synthesiser) can be defined as:

$$C(F_1, F_2) \mapsto l_1 \text{ or } l_2$$

The following is an example of Mivoq XML POS tag for the token "mondo" (meaning "world"):

```
pos="Sms"
mondo
```

There are two approaches for extracting the features from a text (which are common to other NLP tasks and otherwise, see 2.2.4.1): by rules and machine learning.

1. The first is also named "hand-written", meaning that a human designer evaluates the effectiveness of the estimated tags and creates or improves the algorithm designing rules. A good working hand-written algorithm needs good working rules, for which reason such method is toilsome because of the vast number of exceptions to the rules. Such approach is therefore efficient with those languages whose morphosyntax is enough straightforward.

2. The second approach is also named "data driven", where the mapping is learnt by the machine through the analysis of a database of examples. This method is more flexible, and comprises three operative phases: training, testing and development.

   Training. The training phase consists of providing the input data and tuning the algorithm. The input data have to be numerous, diverse and similar to real cases, even if it may depend on the task assigned to the synthesiser. The tuning is made on the basis of several factors: subjectivity of the researcher, input data, expected output and required task. Taking into account such factors, may lead to the decision to prefer, for example, quality of the output than speed, or vice versa. The training can be supervised or unsupervised. In a supervised training the user provides a reference or gold-standard, meaning labelled examples which match the input data with the respective expected output. The idea is that the algorithm can be trained to reproduce the correspondence, given a set of pairs of input data and labels. In the unsupervised training the user only provides the input data; the algorithm looks at the inherent patterns in the data to come up with a mapping. Such kind of training may be chosen for being less laborious for the researcher, but it tends to provide worse results than a supervised training.

   Test. The testing phase serves to verify the effectiveness of the tuned algorithm, and involves a combination of automatic processing and human correction. The user provides a percentage (round about the 90%) of labelled data, which are used for the training, and the remaining unlabelled input data are used for testing the algorithm, which is assessed through the calculation of metrics (see 2.2.6). In order to achieve significant results, the test is repeated many more times with the same corpus of input data but changing in each test-session the data which are labelled and

the ones which are not. The average of the test results gives a significant percentage of the algorithm effectiveness.

Application. In the application phase the model developed is executed and its results are achieved. They cannot be evaluated such as in the testing phase, because in this phase the model is applied to real-world data, without a reference.

2.3.2.5 Chunk parser or syntactic analysis

The chunk parser provides information about syntactic phrasing, by grouping words and determining the boundaries of noun phrases, adjective phrases, etc. This process is necessary because the sequence of tagged tokens achieved at this stage of the processing in not enough, thus a further level of analysis is required to determine the underlying structure of the sentences. There are several justifications to perform syntactic analysis. In the first place it is useful for determining the prosodic phrase structure of utterances, namely prominence and intonation. Secondly, it serves for delimiting the domain for morphological unification, for example for assigning the correct inflection ending to expanded abbreviations. Moreover, such level of analysis can resolve part-of-speech ambiguity problems. Consider the following examples:

1. The/*det* project/*noun* was/*aux* behind/*prep* schedule/*noun*.
2. The/*det* problem/*noun* is/*aux* how/*wrb* to/*inf* project/*verb* the/*noun* missile/*noun* accurately/*adv*.
3. The new images project well on to the wall.

In sentences `1` and `2` the POS-tagger manages to resolve the noun/verb homographs ambiguity (see 2.3.2.4) for the token "project". In sentence `3` the POS-tagger alone is not sufficient to operate the disambiguation, thus the syntactic analysis is required [36].

In some TTS systems a dependency parser tree is create; in other systems, such as in Mivoq synthesiser, chunking information is added to the token tags <t>...</t> [33]. It would be a more logical and satisfactory solution to insert syntactic information into an XML tree structure, but in such way there could be conflicts between syntactic and prosodic tree, for two reasons. First, XML language does not allow overlapping between edges of different tree structures (even if it could be linguistically more appropriate to partially overlap syntactic and prosodic phenomena). The second reason, which is directly dependent from the previous one, is that syntactic and prosodic structures do not entirely coincide in all cases.

2.3.2.6 Semantic analysis

Besides the syntactic analysis, sometimes (not in the case of Mivoq synthesiser) a semantic analysis of is required. For example, the Italian written form "osservatori" has two underlying forms which do not differ for the part-of-speech of the tokens, since they both are nouns. Their pronunciation differs for the height of the vowel "o", in fact /osserva'tori/ means "observers", and /osserva'tɔri/ means "observatories". In such cases a semantic disambiguation is needed.

2.3.2.7 Phonemisation

The phonemisation module works through two different ways.

1. **Lexicon.** The first is lexicon, which deals with known words of a language or with languages with a complex phonological system. It contains the graphemic forms and phonetic transcriptions of the lexical entries of a language. As regards compounds, namely the concatenation of two or more lexical entries, the corresponding phonetic forms are concatenated. The accentuation rules for the compounds differ from language to language. In German the primary word stress is reduced to secondary stress for all the parts of the compounds except the first, which is considered the dominant one. For example the German compound "geshäftsführer", meaning "managing director", has the stress on the first element. Instead in Spanish with-dash compounds each part of the compound is independent and follows the general accentuation rules, as in "físico-químico", meaning "physical-chemical"; in the without-dash compounds to keep the primary stress is only the last part of the compound, as in "sacapuntas", meaning "pencil sharpener", where the primary stress is kept in the second element of the compound.

2. **Grapheme-to-phoneme (G2P)**. The second way for phonemisation is a conversion letter-to-sound algorithm. The letter-to-sound (L2S) or grapheme-to-phoneme (G2P) conversion rules are applied for unknown words, namely such words for which there is no entry in the lexicon, or with languages which have a straightforward phonological system, such as Spanish. In order for a good working G2P algorithm it does not suffice a mere letter-to-sound conversion, but rather morpheme boundaries, syllabification and words stress information are to be added. First, a statistical morpheme parser makes a morphological decomposition (this step lacks in Mivoq synthesiser). The resulting

morpheme chain is compared to a list of affixes which have a predictable effect on word stress position. In fact each affix can attract or shift the stress, or can have no effect on it. Then the transcribed morphemes are syllabified following the standard phonological principles of the language, such as the sonority hierarchy of phonemes, the maximal onset principle, the obligatory coda principle and the phonotactic restrictions. Finally, a syllable receive the primary stress, according to a word stress assignment algorithm [33].

The following is an example of Mivoq XML tag for the phonemisation of the token "Benvenuto" (meaning "welcome"):

```
g2p_method="lexicon" ph="beNF-e-'nu1-to"
Benvenuto
```

Where `"lexicon"` indicates that the entry was taken from the lexicon rather than from the algorithm for G2P conversion; `NF` is the phonetic transcription for the labiodental nasal /ɱ/ and `u1` indicates that such vowel receives the tonic stress.

2.3.2.8 Prosody rules

The tool used for modelling prosody in Mivoq synthesiser is ToBi (Tones and Break Indices). Prosody is described in terms of fundamental frequency (F0) target points and phrase breaks, and the prosody rules module assigns symbolic labels. For example, label "2" denotes a potential boundary location, "3" an intermediate phrase break, "4" an intra-sentential phrase break and "5" a sentence-final boundary. In a later step labels are converted into concrete F0 targets and pause duration [33].

An example of Mivoq XML tag for the prosody of the token "sintesi" (meaning "synthesis") is:

```
accent="L+H*"
sintesi
```

Prosody rules are derived through corpus analysis, and in particular from POS tagging and syntactic analysis modules. For example, the punctuation signs, some conjunctions and the output of the chunk parser serve for locate prosodic boundaries. After determining prosodic boundaries and pitch accents, the tones are assigned on the basis of the sentence type, namely declarative, interrogative -wh, interrogative yes-no or exclamatory.

2.3.2.9 Post-lexical phonological processes

In the post-lexical module we have as input the words transcribed in a standard phonemic string, including syllable boundaries, lexical stress, prosody labels and prosodic phrase boundaries. Such contents can be re-structured by phonological rules, which operate on the basis of phonological context information. Currently only segment-based rules apply in, such as co-articulation and glottal stops. For the future it is planned to apply suprasegmental rules, in order to achieve a reduced number of pitch accents and phrase boundaries for fast speech MaryTTS [33]. The output of this module is the richer XML structure, including all the information added to the structure by all the previous modules.

2.3.2.10 From linguistic specification to acoustic parameters

In Mivoq synthesiser the output linguistic specification comprises elements which might affect the acoustic realisation of the speech sounds, such as phoneme sequences (segmental information) and prosody (suprasegmental information). Linguistic specification can be considered as a summary of all the contextual factors in which a phoneme appears. It will therefore include the surrounding phonemes, the prosodic pattern, the position of the segment in the syllable, word and utterance, the part-of-speech (POS), and it could also include paralinguistic factors, such as the mood of the speaker or the identity of the listener. To reduce the vast number of effectively different contexts we have to consider that not all the factors always produce significant effects: which contextual factors are taken into account depends on the HMM configuration chosen, that affects the aspects of parametrisation of the speech signal and the choice of modelling unit [23](see 2.2.7.1).

Thus, the last module converts the symbolic domain to the parametrical one. For each tone symbol a set of target points are positioned on the nucleus of the syllable they are attached to. Regarding frequency, the target points are related to a pair of topline and baseline representing the highest and lowest possible frequency at a given moment. This allows the calculation of concrete frequency target values. Some values need to be set appropriately for the voice used for the synthesis, for example according to the gender and age of the speaker.

Mivoq interface for the acoustic parametrisation module of the word "benvenuto" from the sentence "Benvenuto nel mondo della sintesi vocale" (meaning "welcome to the world of speech synthesis") is showed in Figure 2.5.

Front-end of Mivoq TTS		
Module	Task	XML tag
Sentence splitting	Text segmentation into sentences	<s>...</s>
Tokenisation	Extraction of words (tokens) from the sentences	<t...</t>
Normalisation	Expansion of tokens to their pronounceable form	<t...</t>
POS tagging	Homograph disambiguation and part-of-speech assignment	pos="..."
Syntactic analysis	Definition of phrase boundaries	<phrase>
Phonemisation	Conversion into phonetic alphabet	ph="..."
Prosody rules	Modelling in terms of F0 target points and phrase breaks	accent="..."
Post-lexical phonology	Application of segmental and suprasegmental rules	
Acoustic parameters	Translation from linguistic specification to acoustic parameters	<ph d="..." end="..." f0="..." />

Table 2.4: Summary of the modules constituting the front-end of Mivoq synthesiser.

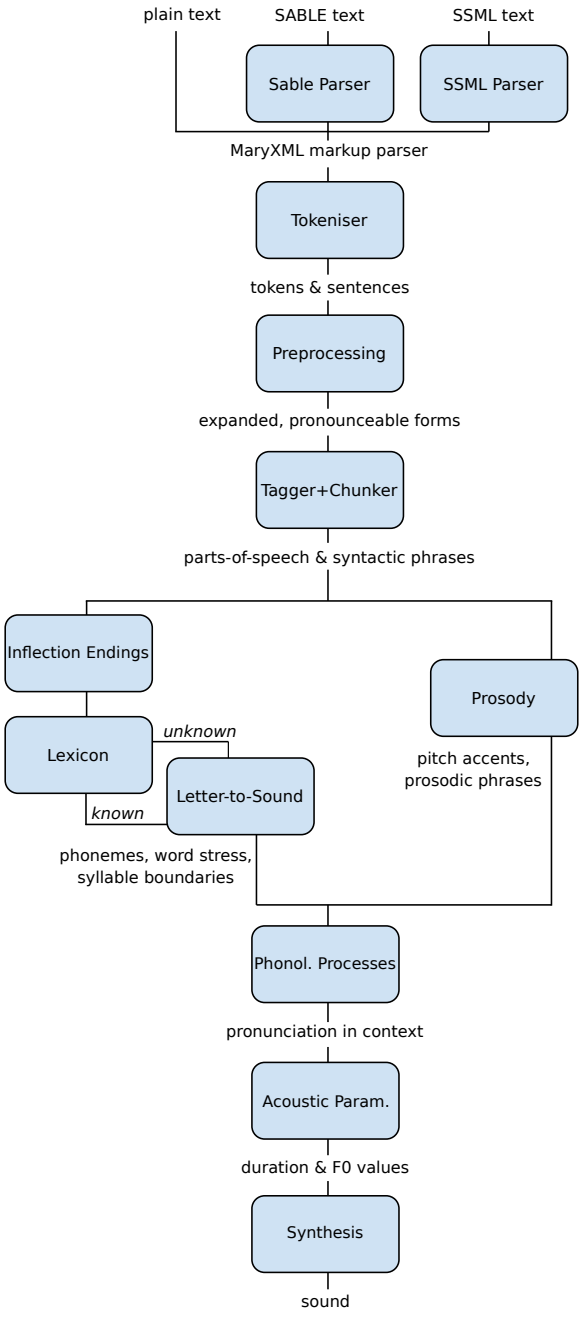

Figure 2.4: Architecture of MaryTTS synthesiser. [33]

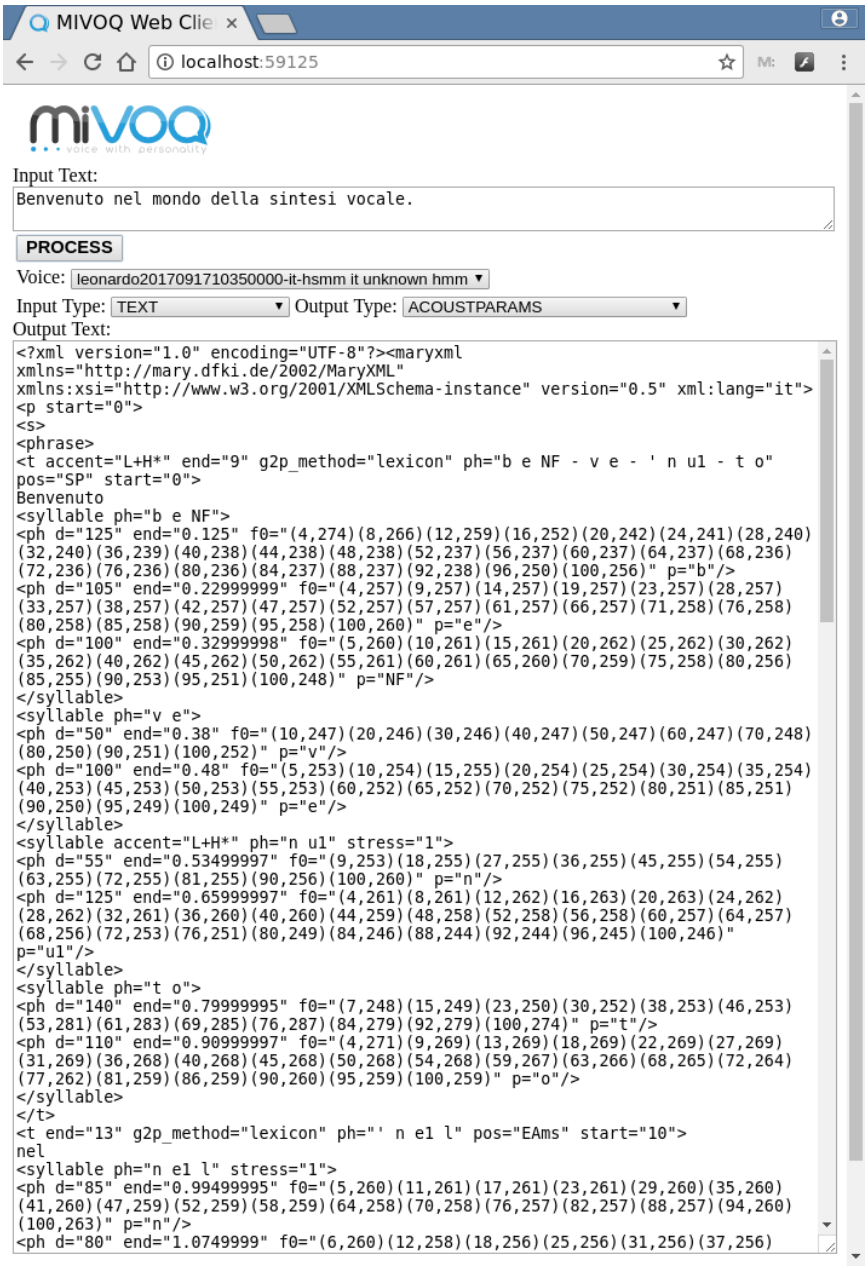

Figure 2.5: Mivoq interface displaying the output of the acoustic parametrisation module for the word "Benvenuto" ("welcome").

Chapter 3

Method

"When one impatient swain asked the UNIX, 'Echo nothing', the UNIX obligingly opened her mouth, echoed nothing, and closed it again."

[22]

"Sed is the ultimate stream editor. If that sounds strange, picture a stream flowing through a pipe. Okay, you can't see a stream if it's inside a pipe. That's what I get for attempting a flowing analogy. You want literature, read James Joyce."

[1]

3.1 Data preparation

3.1.1 LibriVox

LibriVox is a worldwide non-profit group founded by Hugh McGuire in 2005, where volunteers record their voice while reading a book. Audiobooks are made available for free in the public domain in audio format on the web site librivox.org. The catalogue features novels, poems, plays and religious texts in more than 30 languages. LibriVox has got an open structure (see Figure 3.1); the only and loose conditions for allowing a recording to be published is substantial clarity and faithfulness to the source text. Even being in the public domain, the consent form to use LibriVox audiobooks was asked for and obtained for research purposes.

LibriVox was chosen as data source because the material it contains is huge, diverse and readily available. In fact LibriVox data help achieving a

[1] `http://www.grymoire.com/Unix/sed.html`

natural synthetic voice, thanks to the high cohesion of the read and recorded text, compared to the relatively short, untied and incoherent collection of sentences which is generally used as a base-text to be read and recorded to create digital speech. Moreover, normally such sentences are chosen so as to best cover the phonetic contexts of language, and, as a consequence, they often turn out to be complex and contrived. Thus the prosody and the lexicon prove to be far from everyday speech. Being the synthesis process heavily data-driven, the unnaturalness of the base-text necessarily affects the resulting synthetic speech (see 2.2.4.1).

Besides the advantages, LibriVox features some drawbacks in regard to the purposes of our research. Because of the open structure of the web site, some recordings have low audio quality, background noise or non-native accent. Such issues were faced or avoided by designing specific criteria of selection, as illustrated below.

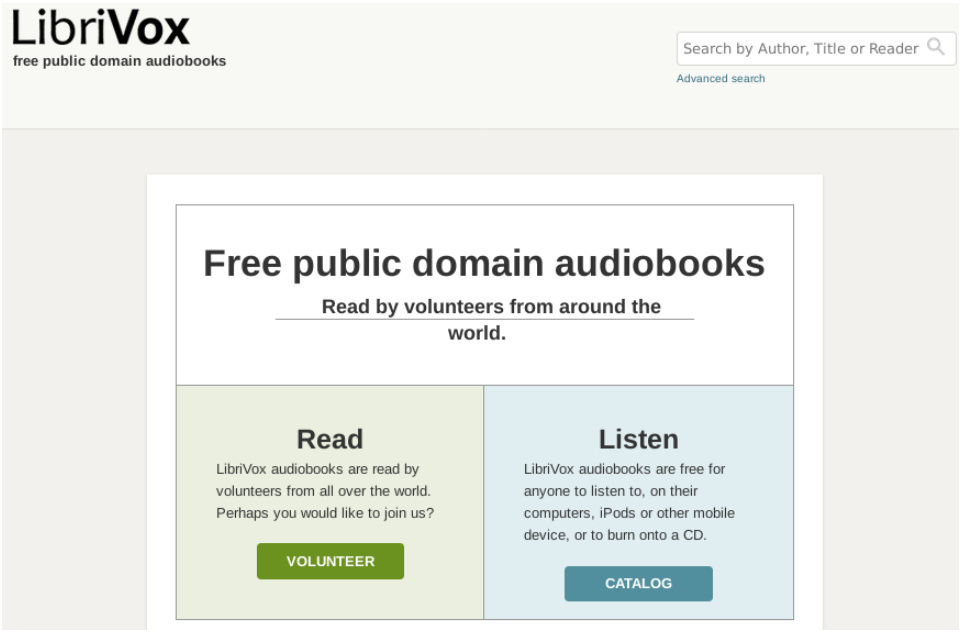

Figure 3.1: LibriVox homepage.

3.1.2 Criteria for data selection

Two audiobooks from LibriVox catalogue, a masculine and a feminine voice, were chosen.

1. Masculine voice: speaker Tux [2] reading "La Batalla de los Arapiles"

[2] `https://librivox.org/reader/3946?primary_key=3946&search_category=`

by Benito Pérez Galdós (1843-1929).

2. Feminine voice: speaker Mongope [3] reading "Juanita la larga" by Juan Valera (1824-1905).

The choice was driven by the following criteria.

Language. Librivox.org allows to easily select Spanish language audiobooks. Standard Castilian accent was purposely needed, thus local varieties, namely the once spoken in the various regions of Spain and throughout Latin America, were excluded.

File specifications. The required audio files are endowed with some objective characteristics, some of which were to some extent unavoidable choices, depending on how librivox.org organises and stores its files and data.

- Format. LibriVox web site stores and makes available for downloading audio files in `.mp3` format. Files were converted in `.wav` format at a later stage (see 3.3.1).
- Bitrate. The web site provides 64 and 128 kb/s `.mp3` files but only for the 64 kb/s files it allows the downloading in `.zip` format, granting the downloading of the entire audiobook, whereas for the 128 kb/s files the chapters are to be downloaded one by one. Thus, first the 64 kb/s files were downloaded. Despite their lower definition compared to the respective 128 kb/s files, they permit a faster downloading of the audiobooks. Once evaluated the relative quality of voice and audio (see below) and chosen the two audiobooks, the respective 128 kb/s files were downloaded (see 3.3.1).
- Duration of the speech. We looked for `.mp3` files whose `.zip` was at least 200 MB because such size accomplishes about 4 hours of speech, which was our final target. If ever more data had been required, other material read by the same speaker and endowed with the same features would have been downloaded.

Voice and audio features. The audiobook had to feature several characteristics which were subjectively evaluated by the researchers. We

reader&search_page=1&search_form=get_results

[3] https://librivox.org/reader/10246?primary_key=10246&search_category=reader&search_page=1&search_form=get_results

chose to make a subjective assessment rather than an objective calculation with automatic tools for two reasons. First, because the features to assess were all perceptive (see below); secondly because handling only few recordings belonging to few speakers we found more economic and effective making a subjective evaluation of them. The voice and audio features needed were:

- a good timbre (or voice quality);
- the absence of reverberation and background noise;
- a high level of the signal (namely the volume of the speech sound) in comparison with the one of background noise.

Single/multiple reader. On librivox.org there are books read by a single reader and others with chapters read by several readers. Our work required audio-files containing single reader voice (see Figure 3.2), with the total of at least 100 minutes of speech per speaker.

Date of publication. Spanish language has kept on changing its phonology until the sixteenth century, therefore books written and published earlier were discarded.

Textual genre. Among the several textual genres in the catalogue, we only selected narrative because it is structurally cohesive and it is, concerning style, lexicon and prosody, the genre more similar to everyday speech.

3.2 Software tools

For our research we used Debian operating system, based on the Linux kernel and GNU programs. The advantages of such Unix-like and open-source operating system are:

1. free access to the largest online repositories of software packages;
2. easy source for natural language processing software;
3. brings to fruition the shell command-line interpreter to create and execute commands and scripts. It allows, for instance, to process text files, by selectively adding, deleting, substituting parts of the text (see 3.3.1). This is made by a terminal application which provides a text-based user interface (TUI). The significance of using a powerful tool as bash as shell interpreter, lies on the possibility to automatise even toilsome and complex tasks.

Figure 3.2: Spanish catalogue in librivox.org.
By selecting "solo", only single-reader audiobooks were displayed.

3.2.1 Shell scripting

The power of a system like shell scripting does not lie in the programs themselves, but in the relationship among programs, by combining programs to build programs [22].

The purpose of this subsection is far from being exhaustive; it wants to give some preparatory concept to better understand the syntax of the scripts presented below.

Script. A script is a text file which is made executable in order to run as a program. To make a file executable you have to type on the command line:

```
$ chmod +x    filename
```

Shell. It is the command interpreter. It allows to write programs and to run them.

Bash (Bourne Again SHell.) It is a modified clone of the shell.

Variable. A variable is a container including the values you set. A variable is set by assigning it a name, in this way:

```
variable_name="variable_value"
```

It can be retrieved by prefixing its name with the symbol `$`, as follows:

```
$variable_name
```

Pattern matching. It is the occurring of given conditions, such as a pattern found.

Shebang. It is a sequence of characters at the beginning of a script which specifies that it has to be executed using the bash (or shell) interpreter.

```
#!/bin/bash
```

Prompt. It is a symbol (typically a `$`) indicating that the PC is ready for the following command, thus it is found first of each command line.

Statements. There are looping statements and conditional statements.

- `for...in...do...done`. It is a shell control-flow statement which loops over a set of arguments (e.g. filenames or urls).
- `if...then...else...fi`. `if` a condition occurs (e.g., there is pattern matching), `then` execute command x, otherwise (`else`) execute command y. The `then` and `else` parts are optional.

Redirection. By default, the output of a program is concatenated on the terminal, but it may be redirected in another file (by the > symbol) or to another program (by the pipe symbol |).

Programs. Each program has its peculiar use and syntax.

- `echo`. Is a program which echoes its argument, printing it in the terminal. For example:

  ```
  $ echo "Nothing"
  Nothing
  $
  ```

- `cat`. Concatenates the content of a file, as follows:

  ```
  $ cat filename
  This is the content of the file
  $
  ```

- Filters. Filters are a class of programs that read some input data, perform transformations of them and write the correspondent output.
  - `grep`. It is an unvaluable tool for selecting parts of the output of a program, by searching for a given pattern. Its basic syntax is:

`$ grep` *pattern filename*

And the output is the content of the line(s) containing the pattern.

– `sed`. Its name stands for "Stream EDitor", in fact it streams data and applies the list of commands of the program given by the user (see Figure 3.11), in order, on every line of the input, and writes its edited form as output [22]. Its basic syntax when taking the input from a file is:

`$ sed` *'list of commands' filename*

Its syntax when the input is taken from another program includes the pipe symbol, meaning that the output of the command performed by the program preceding the pipe is the input of the program which follows the pipe:

`$ echo` *'input'* | `sed` *'list of commands'*

Special characters. They are characters with special meanings.

- `*` (asterisk) Matches with any character. For example:

```
$ echo D*
$ Desktop Documents Downloads
```

The shell interprets the special meaning of `*` by echoing all the filenames in the current directory which start with "D" followed by any character.

- \ (backslash) Removes the special meaning to a character. For example:

```
$ echo \*
$ *
```

The shell interprets `*` without its special meaning and echoes `*` itself.

3.3 Data elaboration

3.3.1 Automatic audiobooks downloading and conversion

Once chosen the two audiobooks on LibriVox web site, we had to handle the single chapters of each of them, thus we had to download them one by

one. In order to accelerate the downloading phase, it were automatised by creating a shell script (see Figure 3.3). Given the link to the right librivox.org page (see Figure 3.4 above), the script, making use of a pipeline, extracts the respective HTML code source, then filters its content and keeps only the precise links to the `.mp3` files containing the required audiobook chapters. Such links are finally sent to a program for the downloading (see Figure 3.4 below). The advantage of this script is its genericity: in fact, it may be used for the automatic downloading of whichever corpus data taken from LibriVox (and, with few changes, from any web site). Once downloaded, another shell script converts each `.mp3` to `.wav` file (see Figure 3.5). The necessity of the conversion step lies in the fact that both the software involved in the next phase of elaboration and our speech synthesis system require `.wav` files as input. The output, namely a list of `.wav` files containing a chapter of an audiobook each, was then processed in the following phase.

```
#!/bin/bash

if [ -z "$1" ]
then
    echo "$0: missing URL"
    echo "Usage: $0 URLFROMLIBRIVOX"
    echo "For example: $0 https://librivox.org/la-batalla-de-los-arapiles-by-benito-perez-galdos/"
    exit
fi

if [ ! `echo "$1" | grep 'https:\/\/librivox.org\/.*\/' -c ` = "1" ]
then
    echo "$0: wrong URL"
    echo "Usage: $0 URLFROMLIBRIVOX"
    echo "For example: $0 https://librivox.org/la-batalla-de-los-arapiles-by-benito-perez-galdos/"
exit
fi

for i in $(wget "$1" -q -O -)
do
    mp3_url=$(echo $i | grep "s.mp3" | grep -P -o "(?<=(href\=\"))(.*?)(?=(\"))")

    if [ -n "$mp3_url" ]
    then
        wget "$mp3_url"
    fi
done
```

Figure 3.3: Shell script in bash for the automatic downloading of audiobooks in `.mp3` format (see 3.2.1).

3.3.2 Audio segmentation and alignment

To facilitate the processing of the input data, the `.wav` files were divided into segments shorter than a minute and aligned with the corresponding text, in turn segmented. This was made with Transcriber software, a tool for manual segmentation, labelling and speech transcription [4]. It is useful for linguistic research thanks to its user-friendly graphical interface which allows to fulfill in a quick and easy manner tasks as:

[4] `http://trans.sourceforge.net/en/presentation.php`

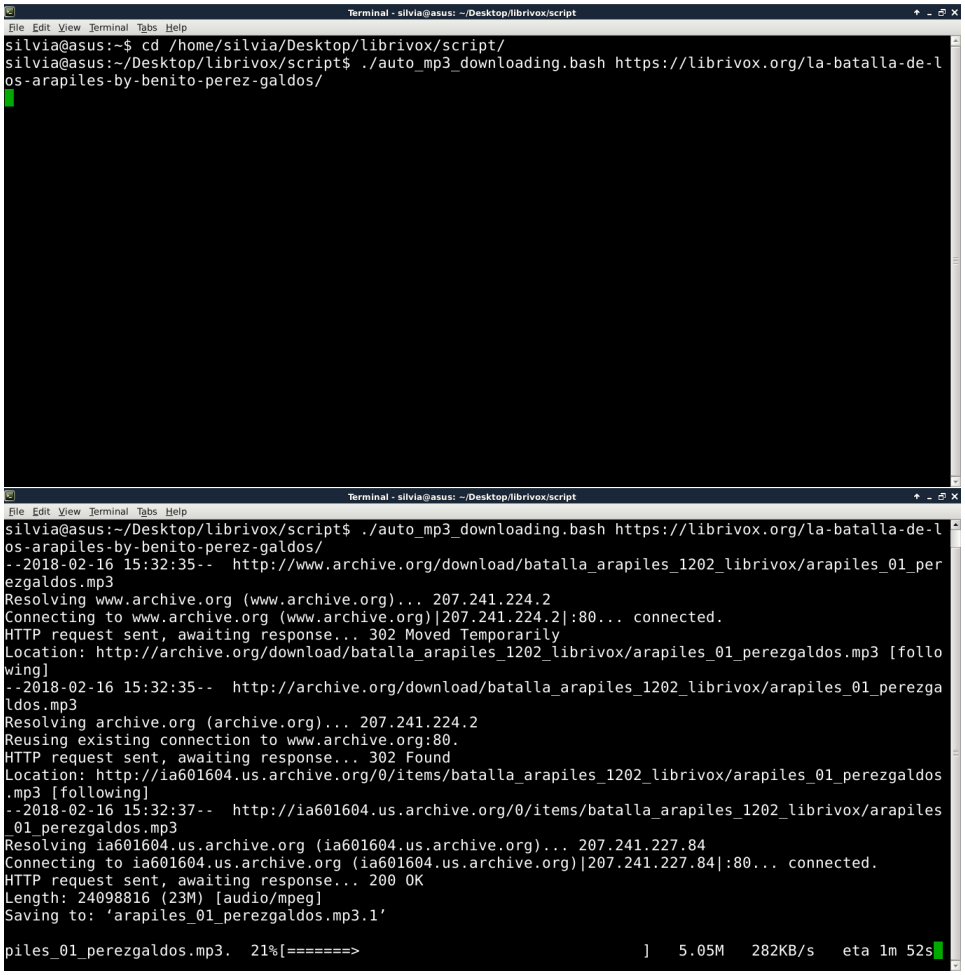

Figure 3.4: What happens on the terminal when running the script for the automatic downloading.

1. showing the segmentations under the signal and in the text editor;
2. synchronising the text editor and the audio signal;
3. easily moving segment boundaries;
4. putting to use standard text edition features like cut/copy/paste and find/replace;
5. keeping previous versions in backup files.

Concerning the latter bullet point, we also used Git, a version control system. It records changes to a file over time so that you can recall specific versions later. Therefore this tool allows to revert specific files or the entire

```
#!/bin/bash
for filename_mp3 in *.mp3
do
    filename_wav=${filename_mp3%.mp3}.wav
    sox $filename_mp3 $filename_wav
done
```

Figure 3.5: Shell script in bash for the automatic conversion of files from `.mp3` to `.wav` format (see 3.2.1).

project back to a previous state, compare changes over time, recover lost files, see changes which might be causing a problem, and more.

Therefore each audio file and the respective text were uploaded on Transcriber, then both were segmented and aligned (see Figures 3.6 and 3.7). The output of such process is a list of files with `.trs` extension, each containing a segmented and aligned audiobook chapter.

In the following phase such output will be elaborated to create pairs of files, where each pair is made up of an audio file in `.wav` format and a text in `.txt` format for each segment. These pairs will be the input of the training phase.

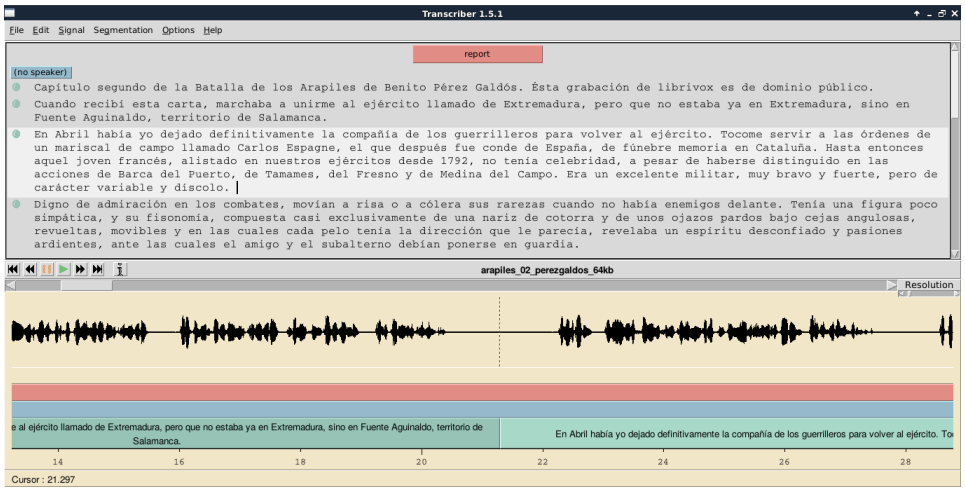

Figure 3.6: Text and audio of chapter 2 of Tux audiobook, both segmented and aligned. The light blue bullet points indicate the boundaries between text segments, the vertical red line on the soundwave indicates a boundary between audio segments. Text chunks were manually aligned to the correspondent waveform segments.

arapiles_02_pe...galdos_64kb.trs

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE Trans SYSTEM "trans-14.dtd">
<Trans scribe="silvia" audio_filename="arapiles_02_perezgaldos_64kb" version="3" version_date="170707">
<Episode>
<Section type="report" startTime="0" endTime="260.389">
<Turn startTime="0" endTime="260.389">
<Sync time="0"/>
Capítulo segundo de la Batalla de los Arapiles de Benito Pérez Galdós. Ésta grabación de librivox es de dc
<Sync time="9.978"/>
Cuando recibí esta carta, marchaba a unirme al ejército llamado de Extremadura, pero que no estaba ya en E
<Sync time="21.297"/>
En Abril había yo dejado definitivamente la compañía de los guerrilleros para volver al ejército. Tocome s
<Sync time="59.992"/>
Digno de admiración en los combates, movían a risa o a cólera sus rarezas cuando no había enemigos delante
<Sync time="92.587"/>
Muchas de sus acciones revelaban lamentable vaciedad en los aposentos cerebrales, y si no peleamos algunas
<Sync time="122.942"/>
Mandaba este hombre una división perteneciente al ejército de que era comandante general D. Carlos O'Donne
<Sync time="171.769"/>
Yo había recibido el empleo de comandante en Febrero de aquel mismo año. Por mi ventura mandé durante algu
<Sync time="214.841"/>
Obedecí, y como en los mismos días recibiese la última carta de las que puntualmente he copiado, juzgué fa
<Sync time="243.299"/>
Mientras con zozobra y desesperación revolvía en mi mente distintos proyectos, ocurrieron sucesos que no c
<Sync time="252.58"/>
Fin del capítulo segundo.
<Sync time="256.192"/>



</Turn>
</Section>
</Episode>
</Trans>
```

Figure 3.7: Text and audio of chapter 2 of Tux audiobook, both segmented and aligned, in xml version.

3.4 POS tagging

3.4.1 Portuguese Corpus

For selecting an adequate POS tagged Portuguese corpus, a web research was made. In order to compare the corpora, their main features were summarised, as illustrated in Figure 3.8.

The corpora were considered on the basis of the following criteria:

Corpus size. The corpus had to be wide, and its size was measured in number of words (at least 100.000). When the corpus size was not explicit in its web site, an estimate of the number of words was made by downloading the corpus and observing its features.

Linguistic variety. Diatopic variations were controlled. Brazilian and European Portuguese differ on phonology, syntax and lexicon, and the extent to which such differences are significant depends on the researcher's choice.

Further variations. There can be diachronic variation in historical corpora or other variations regarding textual genre or topic. They may be functionally employed or discarded.

Tag set. There is not a standard international tag set, thus the employed

tags can be more or less fine-grained and may give or give not additional morphological information on the words. For example, depending on the POS tagger, the same verb may bring the tag `VB.PRET.ACT` (indicating past tense, active form), `Vip` (indicating main verb, indicative, present), or `VMI` (indicating verb, main, indicative).

Tag format. The POS tagged corpus may have the tags next to each token in the same line, as:

```
word1_TAG1 word2_TAG2 word3_TAG3
```

or one token with the correspondent tag per line, as:

```
word1 TAG1
word2 TAG2
word3 TAG3
```

Licence. It is necessary to understand the terms or conditions under which each corpus is distributed. For example, the granted baseline right for the works protected by the Creative Commons licence (CC) is to distribute them worldwide without commercial purposes. For the purposes of our work, we need the permit to make derivative works and distribute them for, primarily, research and, secondly, commercial purposes.

For the corpora for which the kind of licence was not specified, we got in touch with the developers asking for permission.

3.4.2 Spanish corpus

For Spanish language, the selected corpus was AnCora-ES [5]. AnCora (ANnotated CORporA) are multilingual and multilevel corpora developed by the CLiC-UB researchers for linguistic analysis and NLP applications. They are multilingual because there are one for Catalan (AnCora-CAT) and one for Spanish (AnCora-ES) and they are multilevel because they are annotated at several levels of linguistic analysis (e.g. morphological, syntactic, semantic and pragmatic level).

Corpus size. AnCora Spanish corpus includes 500.000 words.

[5] `http://clic.ub.edu/corpus/`

Corpus	Linguistic variety	Tag Format	Provided subdivision	Size (words)	Licence
1. Mac-Morpho	Brazilian Portuguese	In a row*	Training Test Development	700.000 100.000 30.000	Commercial (CC BY 4.0)
2. NILC's	Brazilian Portuguese; literary, journalistic, didactic texts.	In a row*	Training Tagset Evaluation	100.000	
3. COPLE2	L2 Portuguese	One token per line**	L1 Topic Task	Complex to estimate	
4. Bosque		One token per line**	Training Test Development	140.000	Commercial (CC BY-SA 4.0)
5. Tycho Brahe	Historical Portuguese (1380-1881)	In a row*	38 different text-files	2.000.000	Research purposes

* w1_TAG1 w2_TAG2 w3_TAG3

** w1 TAG1
w2 TAG2
w3 TAG3

Figure 3.8: Comparison between POS tagged Portuguese corpora.

Linguistic variety. AnCora corpora are mainly based on journalistic texts.

Tag set. AnCora tag set provides a tag (or terminal node) for each part of speech. Each node has several attributes, each attribute has, in turn, several attribute values, which may have secondary attributes and so on, as illustrated in Figure 3.9.

Node / *description*	Attributes / *description*	Attribute values / *description*
<n> / *noun*	**<gen>** / *gender*	**"c"** / *common* **"f"** / *feminine* **"m"** / *masculine*
	<num> / *number*	**"c"** / *common* **"p"** / *plural* **"s"** / *singular*
	<postype> / *PoS subclassification*	**"common"** **"proper"**

Figure 3.9: AnCora tag set for the node "noun".

Tag format. The tags are in the format "one token with the correspondent tag per line", as:

```
word1 TAG1
```

```
word2 TAG2
word3 TAG3
```

Besides the word (<wd>) and the correspondent POS tag (<pos>), AnCora corpus provides other morphological information, such as attributes, attribute values (see above) and lemma (<lem>), which denotes the unmarked form of the word. For example, for the word "reservas", AnCora provides the following annotations:

```
<n gen="f" lem="reserva" lexicalid="noun.reserva.2.default"
num="p" origin="deverbal" pos="ncfp000" postype="common"
sense="16:09626584" wd="reservas"/>
```

Licence. AnCora corpora are distributed under GNU GPL (General Public Licence) 3.0 licence, according to which they may be modified and distributed for both research and commercial use.

3.4.3 POS tagger

HunPos tagger. We chose HunPos (Hungarian POS tagger) as a software for POS tagging. It is, roughly, a tool for estimating lexical probabilities, implemented in OCaml language. Its main advantages are:

- It is free and open source (under LGPL licence), thus the details for the tuning are accessible to the user, in order for him/her to implement the training and tagging process across languages [28] even for commercial purposes. Is it a re-implementation of TnT tagger, which is free but only in executable form (being not open source) and it is, in turn, an implementation of the Viterbi algorithm for second-order Markov chain.
- OCaml is a high-level language supporting a succinct, well maintainable coding style and whose compiler produces native code with speed comparable to a C/C++ implementation.
- Its development has had precision of tagging on unknown and unseen words as priority.
- It is reliable with large amounts of tag sets. For example, an inflected (or fusional) language as Hungarian, whose single inflectional morphemes convey multiple features, both semantic and syntactic, needs to preserve detailed morphological information in the POS tags, leading to significantly larger tag sets, compared

to, for example, the one required for analytic languages such as English.

- The context window for the contextualised lexical probabilities as an undefined size, thus, unlike traditional HMM models, probability estimates are based on the current as well the previous tags.

Corpus and tagger elaboration. HunPos tagger was downloaded from the web [6] and, then, compiled by means of mivoq_nlp_scripts. Such scripts were modified in order to accept AnCora-ES as input. AnCora-ES follows the format illustrated in Figure 3.10.

```
<article lng="es">
  <sentence>
 <S arg="arg1" clausetype="completive" coord="yes" func="cd" tem="pat">
      <S clausetype="completive">
        <sn arg="arg1" func="suj" tem="tem">
          <spec gen="f" num="p">
     <d gen="f" lem="el" num="p" pos="da0fp0" postype="article" wd="Las"/>
          </spec>
          <grup.nom gen="f" num="p">
     <n gen="f" lem="reserva" lexicalid="noun.reserva.2.default" num="p"
          origin="deverbal" pos="ncfp000" postype="common" sense="16:09626584" wd="reservas"/>
     <sp arg="arg1" func="cn" tem="pat">
               <prep>
                 <s lem="de" pos="sps00" postype="preposition" wd="de"/>
               </prep>
               <sn complex="yes">
                 <grup.nom coord="yes">
                   <grup.nom gen="m" num="s">
         <n gen="m" lem="oro" num="s" pos="ncms000" postype="common" sense="16:10487505" wd="oro"/>
                   </grup.nom>
                   <conj conjunctiontype="coordinating">
         <c lem="y" pos="cc" postype="coordinating" wd="y"/>
                   </conj>
                   <grup.nom gen="f" num="p">
         <n gen="f" lem="divisa" num="p" pos="ncfp000" postype="common" sense="16:05149395" wd="divisas"/>
                   </grup.nom>
                 </grup.nom>
               </sn>
     </sp>
     <sp arg="arg0" func="cn" tem="agt">
               <prep>
                 <s lem="de" pos="sps00" postype="preposition" wd="de"/>
               </prep>
               <sn entityref="ne" ne="location">
                 <grup.nom>
                   <n lem="Rusia" ne="location" pos="np00001" postype="proper" wd="Rusia"/>
```

Figure 3.10: AnCora POS tagging of the sentence "Las reservas de oro y divisas de Rusia" (meaning "The reserve of gold and the currency of Russia")

To make the corpus usable as input for the POS tagger, the morphological annotations were filtered by means of a script created by Mivoq which parses the XML input structure, in order to keep only the words and the correspondent POS tags, such as:

[6] `https://code.google.com/archive/p/hunpos/`

```
Las da0fp0
reservas ncfp000
de sps00
oro ncms000
y cc
divisas ncfp000
de sps00
Rusia np00001
```

Training and test. The script for the calculation of the POS tagger metrics was run.

Established that:

- A is the entire AnCora-ES corpus;
- X is one tenth of A.

The script:

- Splits A in ten parts ($X_1, X_2, \ldots X_{10}$).
- In turn considers each X for the test, and the rest (thus, A-X) for the training.
- After that the training is made, employs the model for tagging the test set (X).
- Calculates the metrics of recognition of the POS tags, namely how well the tags have been put correctly by the tagger.
- Writes down the results for each metric.
- Repeats the process for each X.

Thus, the corpus A is split into parts, the process is repeated 10 times and, for each repetition, it is used a different set, both for the training and for the test (see 2.3.2.4). This was made in order to avoid the randomness and to achieve a significant and representative result.

3.5 Rule-based phonemisation module

The phonemisation module, namely the grapheme-to-phoneme (G2P) conversion, is necessary and preparatory for the phoneme-to-speech processing. Besides the G2P conversion module, syllabification and stress assignment rules were added (see 2.3.2.7 and 2.2.4.1).

3.5.1 G2P conversion rules

The G2P module was implemented by rules. For a language as Spanish, whose phonology is straightforward, it is possible to make a list of rules of conversion from grapheme to phoneme (G2P), relying upon a one-to-one grapheme-to-phoneme correspondence and relatively few exceptions to the rules. Such rules were implemented through `sed` scripting language (see 3.2.1), a tool for making changes in a text, so that for each grapheme we obtained as output the correspondent phonetic transcription, according to Spanish phonological rules and coarticulation processes.

From phonological to G2P rules. In order to facilitate the transition from the linguistic knowledge, i.e. the phonological rules, to the programming language which implements such knowledge, a special language was used. It was borrowed from the terminology of formal phonology, which is involved in the description of phonological changes in languages. The basic structure of a phonological rule is:

/A/ → [B] / C_D
CAD → CBD

Where:

- A is the input to which the rule is applied;
- the arrow represents that the sound on its left changes into the one on its right;
- B is the structural change;
- the slash is a shorthand which stands for "in the environment where...";
- the underscore indicates the location of the sound which is going to be changed;
- C and D represent the environment or phonological context.

Thus, this notation can be read as follows: any /A/ becomes a [B] in the environment where it is preceded by C and followed by D.

The structure of the rule may be different. Either C or D may be empty; both C and D empty define a context-free rule; an unspecified A may define an insertion and an unspecified B a deletion [19].

We transposed such terminology into the description of grapheme-to-phoneme rules, so that, for example, the rule stating that the Spanish

grapheme <d> in intervocalic context is pronounced as its correspondent fricative phone [D], was written as follows:

<d> → [D] / V_V

We found convenient to discriminate between basic rules, as the deletion of <h>, which is systematic and context-free, and contextually-dependent rules, which are more complex to define.

Phonetic alphabet. The chosen phonetic alphabet was not the IPA (International Phonetic Alphabet), but rather SAMPA (Speech Assessment Methods Phonetic Alphabet). Such alphabet uses printable ASCII characters, in order to facilitate computational processing of transcriptions in speech technologies. Its purpose is to form the basis of an international standard machine-readable phonetic alphabet. SAMPA is language-dependent, so that there is a SAMPA table accounting for the phonological inventory of each of the principal European Union languages [42]. In SAMPA, the IPA lower-case alphabet symbols remain the same, and the upper-case symbols are largely employed. Thus, for example, to mean the unvoiced interdental phoneme we do not need to type "θ" but simply "T", so that the Spanish word <zero> is transcribed [Tero].

From G2P to `sed` rules. Concerning basic context-free rules, as <h> deletion, the change is straightforward and systematic, so that the grapheme <h> undergoes deletion in all cases, and an input as <hola> will give the output [ola]. Concerning contextually-dependent rules, there are alternations which are more complex to convey, because the same grapheme may give a different output according to the phonological context. For example, the grapheme <c> will be converted into its correspondent phonetic transcription [k] in most cases, but when followed by <e> or <i> it is converted into [T] (the equivalent of [θ] in IPA) and when followed by <h> it is converted into [tS] (the equivalent of [tʃ] in IPA).

All the rules were first transposed into our G2P terminology and then into `sed` syntax. For example the above-mentioned rules accounting for the graphemes <h> and <c> were written as indicated in Table 3.1.

`sed` script. The input is a string of graphemes, words or sentences in Spanish, which pass through the filter (see Figure 3.11) and undergo sub-

From G2P to `sed` rules	
G2P rule	`sed` syntax
<h> → [ ]	`s/h//g`
<c> → [tS] / _h	`s/ch/tS/g`
<c> → [T] / _e,i	`s/c\([ei]\)/T\1/g`
<c> → [k]	`s/c\([^ei]\)/k\1/g`

Table 3.1: Comparison between some G2P rules written with phonology-like terminology and the correspondent `sed` syntax.

stitutions, deletion or other changes, producing as output the same string converted into phonetic alphabet. For example:

```
$ echo "Ejemplo de frase en español" | ./g2p_spa.sed
eXemplo De frase en espaNol
```

As you can see in Figure 3.11, the script consisted of a shebang (see 3.2.1) meaning that `sed`, thanks to the option `-f`, will be the interpreter of the following commands.

```
#!/bin/sed -f
```

The commands of the list, in fact, are all written with `sed` syntax. In order to clarify such syntax, a global picture of the aspects which are common to all the rules is provided, and then some of the more complex of such rules are extracted in order to explain their meaning and function.

`s`/*target*/*replacement*/`g`

- The initial `s` means that the change to apply is a substitution.
- *target* is the string of characters to which the rule is applied.
- *replacement* is the string of characters which have to be put in place of the target.
- The final `g` means that the rule has to be applied to all the occurrences of the target (otherwise it would be applied only to the first occurrence).

1. `s/./\L&/g`

- Global meaning: convert any input to lower case. This was the first of `sed` commands and it was useful for two reasons. First, for avoiding to have to handle both upper and lower case in the input graphemes. Secondly, for avoiding conflicts between the input graphemes and the SAMPA upper case letters used for the replacements.
- The dot matches with any character.
- The `\L` turns the replacement to lower case.
- The `&` corresponds to the pattern found.
- Thus, for example:

```
$ echo "Hola, me llamo Silvia" | ./g2p_spa.sed
ola, me JJamo silBJa
```

2. `s/c\([^ei]\)/k\1/g`

- Global meaning: substitute each <c> which is not followed by <e> or <i> with /k/.
- `c` is the target character to substitute.
- The symbol `^` is a negation of what follows, thus `[^ei]` stands for "any single character which is not 'e' nor 'i'".
- The round brackets preceded by backslashes are required to retrieve the characters they enclose in the second part of the substitution, being such characters `[^ei]` underspecified.
- `k` is the character to put in place of `c`.
- `\1` is the variable which stands for the underspecified characters inside the brackets, thus `k\1` means "replace with k followed by the unvaried character which followed the target "c".
- Thus, for example, for an input as <cocer> the first <c> is replaced by <k>, as follows :

```
$ echo "cocer" | ./g2p_spa.sed
koTer
```

3. `s/\([aeiou]\)b\([aeiou]\)/\1B\2/g`

- Global meaning: substitute each <b> in intervocalic context with /B/.
- `b` is the target character to substitute.
- The square brackets mean "only one of the characters we enclose", thus `[aeiou]` stands for "one vowel".

Conflict SAMPA-grapheme		
Word	Grapheme	SAMPA
<gente>	<g>	/**x**/
<éxito>	<**x**>	/ks/

Table 3.2: Conflict between SAMPA transcription and Spanish grapheme (in bold).

- The round brackets preceded by backslashes are required to retrieve the characters they enclose in the second part of the substitution, being such characters underspecified.
- `B` is the character to put in place of `b`.
- `\1` and `\2` are variables which stand for, respectively, the first and the second underspecified character inside the brackets. Thus `\1B\2` means "replace with B preceded by and followed by, respectively, the the same unvaried characters which preceded and followed the target "b".
- Thus, for example, for an input as <beber> the second <b>, being preceded and followed by vowels, is replaced by <B>, as follows:

```
$ echo "beber" | ./g2p_spa.sed
beBer
```

For convenience, in some cases we moved away from SAMPA inventory because some of its letters conflicted with some input graphemes. For example, the SAMPA transcription for the unvoiced fricative velar phoneme is /x/ (the same as IPA) conflicts with the Spanish grapheme <x>, whose pronunciation is /ks/ (both in IPA and SAMPA), as illustrated in Table 3.2.

Thus, we changed the SAMPA /x/ into /X/. The two following rules should be considered:

1. `s/g\([ei]\)/X\1/g`
   (substitute all the <g> followed by <e> or <i> with /X/)
2. `s/x/ks/g`
   (substitute all the <x> with /ks/)

Since `sed` applies the changes in order, if we had kept SAMPA /x/ instead of /X/, then `sed` would have changed each <g> in /x/ and consequently in /ks/. Thus, an input as <gente> would have wrongly given /ksente/ as output.

3.5.2 Syllabification

The syllabification module was implemented by rules through bash programming language. As for the G2P module, we made a list of rules on the basis of the combinations between Spanish consonants and vowels, according to the standard phonological principles of the language.

Syllabification rules. Being Spanish phonological system rigid, it is possible to state rules for dividing words into syllables with none ore few exceptions. In the first place, as for the G2P modules, we transcribed the rules of syllabification in a schematic manner, in order to outline the combinations of vowels and consonants. Thus, for example, the rule stating that a consonant between two vowels belongs to the syllable with the second vowel, was sketched as follows:

VCV → V-CV

Where, intuitively, "V" denotes a vowel, "C" a consonant and "-" the syllable boundary.

Bash script. The syllabification rules were implemented in bash, through a pipeline of `sed` commands, each one facing the development of one or more rules of syllable division (see Figure 3.12).

First, for convenience, a variable was created (see 3.2.1). The variable name is "vowel", and we set its value in order for it to contain all the Spanish vowels, including the stressed ones.

```
vowel=‘‘aeiouáéíóú"
```

In this way, by retrieving the variable `$vowel` we can easily refer to whichever vowel.

On the contrary, to mean a consonant it suffices to prefix the variable `$vowel` with the symbol `^`, meaning "everything except what follows". Thus, `^$vowel` matches a consonant.

For example, in the following `sed` rule:

```
sed "s/\([$vowel]\)\([^$vowel^h]\)\([lr$vowel]\)/\1-\2\3/g"
```

- General meaning: VCV → V-CV
- The pattern `s`/*target*/*replacement*/`g` in `sed` means that the change to apply is a substitution (`s`) and that the rule has to be applied to all the occurrences of the target (`g`) and not only to the first occurrence.

- The square brackets mean "only one of the characters we enclose", thus `[$vowel]` stands for "one vowel".
- The round brackets preceded by backslashes are required to retrieve the characters they enclose in the second part of the substitution, being such characters underspecified.
- The `^` negates what follows, thus `[^$vowel^h]` means "any character which is not a vowel nor 'h'", in other words "any consonant except 'h'".
- The `-` indicates itself, standing for a syllable boundary.
- `\1`, `\2` and `\3` are variables which stand for, respectively, the first, the second and the third underspecified character inside the brackets.
- Thus, the rule literally means: substitute any vowel followed by a consonant (except "h"), followed by "l", "r" or a vowel with the first, followed by `-`, followed by the other two characters.
- Thus, for example:

```
$ echo "sílaba" | ./sp_syllabification.sh
sí-la-ba
```

In order to build and test our script, we created a text file containing 29 Spanish words which cover most phonological contexts which triggers the application of a syllabification rule. For example, diphthongs always form a single syllable, even if separated by a "h", thus we included the word "desahijar" (meaning "to wean") in our test-file, whose syllabification is "de-sahi-jar".

Therefore, we used the test-file as input for the script, and as output we got the 29 test-words divided into syllables, as showed in Figure 3.13.

Finally, the script for syllabification and the one for G2P conversion (see 3.5.1) were run together in a pipeline. The join between them was interesting but not perfect. In fact the scripts were both designed in order to receive graphemes as input, while, in the pipeline, the input of G2P script is the output of the syllabification script, thus the presence of the symbols `-` affects the G2P conversion, as showed in Figure 3.14.

3.5.3 Stress assignment

The word stress position may be easily predicted according to the number of syllables which a Spanish word is made up of.

The rules for the implementation of phonetic stress assignment are essentially two:

1. The stress is on the penultimate syllable when the word finishes with "s", "n" or a vowel.

2. The stress is on the last syllable when the word finishes with a consonant, except "s" or "n".

All the other cases are regulated by the graphic stress position, whose implementation is not required being already present in the input graphemes.

Thus, we may give an idea of the operating principle of a future program implementing the stress assignment providing the following pseudocode (namely an informal high-level description of the functioning of the algorithm):

```
if      graphic stress if present
then    stress is on that syllable

else if     word finishes with "s", "n" or a vowel
then    stress is on the penultimate syllable

else if     word finishes with a consonant (except "s" or "n")
then    stress is on the last syllable
```

```
#!/bin/sed -f
s/./\L&/g
s/ch/tS/g
s/c\([ei]\)/T\1/g
s/c\([^ei]\)/k\1/g
s/h//g
s/z/T/g
s/v/B/g
s/ñ/N/g
s/j/X/g
s/qu/k/g
s/ll/JJ/g
s/\([aeiou]\)b\([aeiou]\)/\1B\2/g
s/\([aeiou]\) \<b\([aeiou]\)/\1 B\2/g
s/d\>/D/g
s/\([aeiou]\)d\([aeiou]\)/\1D\2/g
s/\([aeiou]\) \<d\([aeiou]\)/\1 D\2/g
s/g\([ei]\)/X\1/g
s/gu\([ei]\)/g\1/g
s/\([aeiou]\)g\([aou]\)/\1G\2/g
s/\([aeiou]\) \<g\([aeiou]\)/\1 G\2/g
s/i\([aeou]\)/J\1/g
s/u\([aeiyo]\)/W\1/g
s/\<p\([nst]\)/\1/g
s/\<x/s/g
s/x/ks/g
```

Figure 3.11: Shell script in `sed` for G2P conversion (see 3.2.1).

```
#/bin/bash

vowel="aeiouáéíóú"

sed "s/en\([lr]\)/en-\1/g" \
 | sed "s/\([$vowel]\)\([^$vowel^h]\)\([lr$vowel]\)/\1-\2\3/g" \
    | sed "s/\([$vowel]\)\([^$vowel^-h]\)\([lr$vowel]\)/\1-\2\3/g" \
      | sed "s/\([^$vowel^-]\)\([^$vowel^lrs-]\)/\1-\2/g" \
           | sed "s/\([aeoáéíóú]\)\([aeoáéíóú]\)/\1-\2/g" \
              | sed "s/\([s]\)\([^$vowel^-]\)/\1-\2/g" \
                 | sed "s/\([^$vowel]\)\([-]\)\([ ]\)/\1/g" \
                    | sed "s/en\([lr]\)/en-\1/g" \
                       | sed "s/\([$vowel]\)\([^$vowel^h]\)\([lr$vowel]\)/\1-\2\3/g" \
                         | sed "s/\([$vowel]\)\([^$vowel^-h]\)\([lr$vowel]\)/\1-\2\3/g" \
                            | sed "s/\([^$vowel^-]\)\([^$vowel^lrs-]\)/\1-\2/g" \
                                | sed "s/\([aeoáéíóú]\)\([aeoáéíóú]\)/\1-\2/g" \
                                  | sed "s/\([s]\)\([^$vowel^-]\)/\1-\2/g" \
                                     | sed "s/\([^$vowel]\)\([-]\)\([ ]\)/\1/g" \
                                        | sed -e 's/-\+/-/g' -e 's/^-\+//' -e 's/-\+$//'
```

Figure 3.12: Bash script for Spanish syllabification (see 3.2.1).

```
$ cat sp_words_test.txt | ./sp_syllabification.sh
el                las                 fui
a-pli-ca-do       or-na-men-tal       i-gua-na
en                no                  rei-na
a-que-llas        es-drú-ju-la        cre-er
sí-la-ba          su                  fe-o
u-ti-li-ta-ria    ce-rá-mi-ca         tí-o
tó-ni-ca          en-lo-que-cer       vio
me-no-res         som-bri-lla         trans-por-te
des-ti-na-das     me-ren-gue          de-sahi-jar
son               ciu-dad
```

Figure 3.13: The script for syllabification was tested by using as input 29 Spanish words. The output is the list of words divided into syllables (see 3.2.1).

```
$ cat sp_words_test.txt | ./sp_syllabification.sh | ./g2p_spa.sh
el                las                 fWi
a-pli-ka-do       or-na-men-tal       i-gWa-na
en                no                  rei-na
a-ke-JJas         es-drú-Xu-la        kre-er
sí-la-ba          su                  fe-o
u-ti-li-ta-rJa    Te-rá-mi-ka         tí-o
tó-ni-ka          en-lo-ke-Ter        BJo
me-no-res         som-bri-JJa         trans-por-te
des-ti-na-das     me-ren-ge           de-sai-Xar
son               TJu-daD
```

Figure 3.14: The scripts for syllabification and for G2P conversion were run together in a pipeline. The output is a list of words converted into SAMPA phonetic alphabet and divided into syllables (see 3.2.1).

Chapter 4

Results

In this chapter the results achieved in each step of our research work are presented, as it has been made throughout the work.

Data preparation and elaboration. Two audiobooks from LibriVox catalogue were selected as data source for Spanish speech corpus, a masculine and a feminine voice. We automatised the downloading of the single chapters in a high quality format and the conversion into `.wav` files. Then, we divided the waveforms and the relative texts into segments and aligned one another. The resulting output was a list of XML files with `.trs` extension, each containing an audiobook chapter segmented and aligned to the correspondent text chunk.

POS tagged Portuguese corpus. In order to focus the choice for an adequate POS tagged corpora for the training, a web research was made. A set of five POS tagged corpora was found and a table illustrating their main features was compiled (see 3.8), in order to facilitate the comparison and selection.

Machine learning POS tagging for Spanish. After selecting AnCora-ES as text corpus and HunPos as software tagger (see 3.4.3 for a review of their main features), we elaborated them in order to adjust and facilitate the integration between the one's output and the other's input. Then, the training and test phases were accomplished in order to implement the POS tagging module through machine learning.

In order to evaluate how well the model has been trained, its accuracy metrics against the test sets have been calculated. Table 4.1 shows the performance of the model trained. Each line represents the results of calculation of several metrics, such as accuracy, precision and recall

test	WCR	WER	WSR	ACC	PRC	RCL	F1
test10.ref	0.9608	0.0392	0.0392	0.9608	0.9608	0.9608	0.9608
test09.ref	0.9606	0.0395	0.0394	0.9605	0.9606	0.9606	0.9606
test08.ref	0.9605	0.0395	0.0395	0.9605	0.9605	0.9605	0.9605
test07.ref	0.9622	0.0378	0.0378	0.9622	0.9622	0.9622	0.9622
test06.ref	0.9613	0.0387	0.0387	0.9613	0.9613	0.9613	0.9613
test05.ref	0.9606	0.0394	0.0394	0.9606	0.9606	0.9606	0.9606
test04.ref	0.9613	0.0387	0.0387	0.9613	0.9613	0.9613	0.9613
test03.ref	0.9601	0.0399	0.0399	0.9601	0.9601	0.9601	0.9601
test02.ref	0.9602	0.0398	0.0398	0.9602	0.9602	0.9602	0.9602
test01.ref	0.9601	0.0399	0.0399	0.9601	0.9601	0.9601	0.9601
Average	0.9608	0.0392	0.0392	0.9608	0.9608	0.9608	0.9608

Table 4.1: Accuracy of the trained model for POS tagging.

(see 2.2.6) for each set used as test set (test.ref). The last line shows the average results for each metric.

Among the metrics, WIR (Word Insertion Rate) and WDR (Word Deletion Rate) are not present because for this specific task and the way it has been designed, insertions and deletions do not occur.

Rule-based phonemisation module for Spanish. The phonemisation module was subdivided into grapheme-to-phoneme (G2P) conversion rules, syllabification and stress assignment. They all were implemented in three steps. First, by preliminarily compiling a list of rules to account for all the facets of the tasks. Secondly, ad hoc shell scripts were generated. Finally, the scripts were tested by using Spanish word and sentences as input.

Chapter 5

Discussion

In this chapter the results achieved in each step of our research work are analysed and discussed.

Data preparation and elaboration. The criteria employed for the selection of corpora from LibriVox proved to be adequately designed by the researchers in order to exclude the ones which do not satisfied the criteria of research and to select the ones which best suited them. If ever more speech data would be required in addition to the approximately four hour of speech included in the chosen corpora, other material provided with analogous features would be selected and elaborated.

Concerning the subjective criteria of selection, such as voice quality, high level of the signal and absence of reverberation and background noise, for their nature they could not be evaluated in a manner to ensure their repeatability.

POS tagged Portuguese corpus. Being machine learning techniques heavily bound to the quality and quantity of the text and speech corpus, the research of an adequate corpus for the purposes of the specific task to be accomplished by each module of a TTS synthesiser is an essential step.

Machine learning POS tagging for Spanish. The performance of the POS tagger is satisfactory. In particular, 0.9608 is high, even if its performance does not amount to the level of the more innovative technologies implementing POS tagging module in the current state of art. It is to take into account the fact that POS tagging module individually is not crucial for the overall quality of a TTS system. Therefore, such result may be considered adequate and sufficient for the fulfil-

ment of the task required for the POS tagging module. For example, a potential increase by few percent points for the POS tagging task would improve the perceived overall quality in a non significant manner. Therefore, it would not be economic because an expansive improvement in terms of effort and resources employed for a single module would not be reflected in a robust improvement in the system efficacy.

Rule-based phonemisation module for Spanish. Our modules for phonemisation represent a robust improvement compared to the ones currently employed in Mivoq synthesiser. In fact, Mivoq phonemisation module for Spanish is highly basic. It is based on G2P conversion rules and machine learning trained with a tiny speech corpus, thus it proves to be a weak module. The phonemisation modules elaborated throughout our work, on the contrary, allow to make the G2P conversion of each word, except some exceptions such as foreign words and Spanish words preserving their antique pronunciation, for example the ones derived from Mexican words. An example is the word "México" itself, where the grapheme <x> is pronounced /x/ without following the Spanish rule which phonemises such grapheme /ks/.

Chapter 6

Conclusion

The purpose of our work was far to be definitive, but rather to lay the foundation for a possible integration and adjustment for future work, in order to expand the NLP software modules to join new languages in Mivoq synthesiser catalogue, starting from Spanish and Portuguese.

Thus, we outlined the relevance of LibriVox as data source for TTS systems and the arrangement for the generation of a feminine and a masculine synthetic voice for Spanish and Portuguese.

LibriVox is a free source of huge, diverse and readily available recordings of audiobooks in several languages. Being the output of a TTS synthesiser heavily dependent from the corpus data, and being the read and recorded text on LibriVox cohesive, such audiobooks feature closeness to everyday-speech, thus LibriVox data are suitable to be used as speech corpus in order to generate synthetic speech with high level of naturalness.

The criteria employed for the selection of two Spanish audiobooks from LibriVox, a masculine and a feminine voice, were part objective and part subjective. As regards the subjective ones, we plan for the future to design adequate perceptive test for their quality assessment, in order to guarantee the control over the experimental environment and repeatability of the method as much as possible.

Moreover, we asked for and achieved permissions from LibriVox volunteers for our research purposes. For the future it could be helpful to obtain even commercial permit, in order to allow the progress of this area of research on all fronts and even making it more competitive.

Then, such text and speech corpora were elaborated through Transcriber software and shell scripts. We did not manage to effectively demonstrate by means of perceptive evaluation tests that employing LibriVox as data source contributes to improve the quality of the TTS output, above all assessing

the naturalness of the artificial speech, but we made the preparatory work for the future verification of our thesis.

The advantage of the preliminary selection of POS tagged Portuguese corpora following specific criteria lays the foundation for implementing Portuguese language in our TTS system, starting from the POS tagging module.

The actual implementation of such module for Spanish was satisfactory, making it potentially competitive with the average performance of the systems implementing POS tagging in the current state of the art.

Concerning the rule-based phonemisation module for Spanish, it features an influential enhancement in the context of Mivoq current state of the art.

To conclude, as the future work we intend to further improve and integrate the NLP software modules for both expanding Mivoq multilingual speech synthesiser, and for ultimately demonstrate the advantages bound to the employment of LibriVox as source of speech data in order to improve the naturalness of the resulting synthetic speech.

Bibliography

[1] Assessing text-to-speech system quality. White paper, ScanSoft, 2004.

[2] Abhinav Prasha Abhimanyu Chopra and Chandresh Sain. Natural language processing. In *International Journal of Technology Enhancements and Emerging Engineering Research, vol 1, issue 4, ISSN 2347-4289*, pages 131–134, 2013.

[3] Alan W Black and Keiichi Tokuda. The blizzard challenge-2005: Evaluating corpus-based speech synthesis on common databases. In *Interspeech 2005: 6th Annual Conference of the International Speech Communication Associations*, pages 77–80, 2005.

[4] Paul Boersma. An articulatory synthesizer for the simulation of consonants. In *EUROSPEECH 93*, pages 1907–1910, 1993.

[5] Nick Campbell. Data driven speech synthesis. In *The Journal of the Acoustical Society of America 105, 1029*, 1999.

[6] Nick Campbell. Evaluation of speech synthesis. From Reading Machines to Talking Machines. In *Evaluation of Text and Speech Systems*, pages 29–64, 2007.

[7] Rolf Carlson and Björn Granström. Rule-based speech synthesis. In *Handbook of Speech Processing*, pages 429–436, 2008.

[8] Noam Chomsky and Morris Halle. *The Sound Pattern of English.* Harper and Row, 1968.

[9] Gobinda G Chowdhury. Natural language processing. In *Annual Review of Information Science and Technology, 37*, pages 51–89, 2003.

[10] Jack Copeland. In *Artificial Intelligence. A Philosofical Introduction*, 1993.

[11] Jack Copeland. What is artificial intelligence? In *AlanTuring.net. Reference Articles on Turing*, 2000.

[12] Thierry Dutoit. Corpus-based speech synthesis. In *Handbook of Speech Processing*, pages 437–456, 2008.

[13] Mike Edgington. Investigating the limitations of concatenative synthesis. In *Eurospeech*, pages 1–4, 1997.

[14] Ellen M Eide et al. A corpus-based approach to expressive speech synthesis. In *SSW5-2004*, pages 79–84, 2004.

[15] Keiichiro Oura et al. Simultaneous acoustic, prosodic, and phrasing model training for TTS conversion systems. In *Proceedings - 2008 6th International Symposium on Chinese Spoken Language Processing, ISCSLP 2008*, pages 1–4, 2008.

[16] Tom Fawcett. An introduction to ROC analysis. In *Pattern Recognition Letters 27*, pages 861–874, 2006.

[17] Keiichi Tokuda Heiga Zen and Alan W Black. Statistical parametric speech synthesis. In *Speech Communication*, 2009.

[18] Wolfgang J Hess. Section Introduction. A Brief History of Applications. In *Progress in Speech Synthesis*, pages 563–564, 1997.

[19] S Hunnicutt. Grapheme-to-phoneme rules: A review. In *Speech Transmission Laboratory, QPSR 2-3*, pages 38–60, 1980.

[20] Andrew J Hunt and Alan W Black. Unit selection in a concatenative speech synthesis system using a large speech database. In *Proc. ICASSP-96*, pages 373–376, 1996.

[21] Joseph P Olive Jan P H van Santen, Richard W Sproat and Julia Hirschberg. Preface. In *Progress in Speech Synthesis*, page V, 1997.

[22] Brian W Kernighan and Rob Pike. *The UNIX Programming Environment*. Prentice Hall, 1984.

[23] Simon King. A beginners guide to statistical parametric speech synthesis. In *A tutorial on HMM speech synthesis*, 2010.

[24] Peter Ladefoged and Keith Johnson. Acoustic phonetics. In *A course in phonetics*, pages 187–216, 2010.

[25] Elizabeth D Liddy. Natural language processing. In *Encyclopedia of Library and Information Science*, 2001.

[26] Eduardo Reck Miranda. *Computer Sound Design. Synthesis Techniques and Programming.* Focal Press, 2002.

[27] Richard W Sproat Joseph P Olive. A modular architecture for multilingual text-to-speech. In *Progress in Speech Synthesis*, pages 565–573, 1997.

[28] András Kornai Péter Halácsy and Csaba Oravecz. HunPos - an open source trigram tagger. In *Proceeding of the ACL 2007 Demo and Poster Sessions*, pages 209–212, 2007.

[29] David M W Powers. Evaluation: from precision, recall and f measure to ROC, informedness, markedness and correlation. In *Journal of Machine Learning Technologies ISSN: 2229-3981 and ISSN: 2229-399X, Volume 2, Issue 1*, pages 37–63, 2011.

[30] Thierry Dutoit (reviewed by). Data-driven techniques in speech synthesis. pages 570–572, 2001.

[31] Rohit Kumar S P Kishore and Rajeev Sangal. A data-driven synthesis approach for indian languages using syllables as basic unit. In *Proceedings of International Conference on Natural Language Processing (ICON)*, 2002.

[32] Mohammad Savargiv and Azam Bastanfard. Study on unit-selection and statistical parametric speech synthesis techniques. In *Journal of Computers and Robotics 7*, pages 19–25, 2014.

[33] Marc Schröder and Jürgen Trouvain. The German Text-to-Speech Synthesis System MARY: A tool for research, development and teaching. 2003.

[34] Tanja Schultz. Speaker characteristics. In *Speaker Classification I*, pages 47–74, 2007.

[35] Kent A Spackman. Signal detection theory: valuable tools for evaluating inductive learning. In *Proceedings of the sixth international workshop on Machine learning*, pages 160–163, 1989.

[36] Paul Taylor. *Text-to-Speech Synthesis.* Cambridge University Press, 2009.

[37] Jun'ichi Tsujii. Computational linguistics and natural language processing. In *Computational Linguistics and Intelligent Text Processing. CICLing 2011, Part I, LNCS 6608*, pages 52–67, 2011.

[38] Alan M Turing. Computing machinery and intelligence. In *Mind, 59*, pages 433–460, 1950.

[39] Aäron van den Oord et al. Wavenet: a generative model for raw audio. 2016.

[40] Mark Huckvale Volker Dellwo and Michael Ashby. How Is Individuality Expressed in Voice? An Introduction to Speech Production and Description for Speaker Classification. In *Speaker Classification I*, pages 1–20, 2007.

[41] Manolis Wallace. General Purpose Applications of AI. In *Emerging Artificial Intelligence Applications in Computer Engigneering*, 2007.

[42] John C Wells. Computer-coding the IPA: a proposed extension of SAMPA. In *University College*, 1995.