COPPE UFRJ
Instituto Alberto Luiz Coimbra de Pós-Graduação e Pesquisa de Engenharia

COMPARISON BETWEEN RULE-BASED AND DATA-DRIVEN NATURAL LANGUAGE PROCESSING ALGORITHMS FOR BRAZILIAN PORTUGUESE SPEECH SYNTHESIS

Luiz Felipe Santos Vecchietti

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia Elétrica, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia Elétrica.

Orientador: Fernando Gil Vianna Resende Junior

Rio de Janeiro
Abril de 2017

COMPARISON BETWEEN RULE-BASED AND DATA-DRIVEN NATURAL LANGUAGE PROCESSING ALGORITHMS FOR BRAZILIAN PORTUGUESE SPEECH SYNTHESIS

Luiz Felipe Santos Vecchietti

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA ELÉTRICA.

Examinada por:

RIO DE JANEIRO, RJ – BRASIL
ABRIL DE 2017

Vecchietti, Luiz Felipe Santos

Comparison between rule-based and data-driven natural language processing algorithms for Brazilian Portuguese speech synthesis/Luiz Felipe Santos Vecchietti. – Rio de Janeiro: UFRJ/COPPE, 2017.

XIII, 60 p.: il.; 29, 7cm.

Orientador: Fernando Gil Vianna Resende Junior

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia Elétrica, 2017.

Referências Bibliográficas: p. 55 – 60.

1. Natural Language Processing. 2. Pronunciation Modeling. 3. Speech Synthesis. 4. Grapheme-to-Phoneme Conversion. I. Resende Junior, Fernando Gil Vianna. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia Elétrica. III. Título.

*A meu pai Cláudio, minha vó
Anna Lucia e meu amigo
Matheus.*

Agradecimentos

À minha família pelo incansável apoio e felicidade compartilhada em todos os momentos.

Ao meu orientador Fernando Gil pela confiança, atenção e por todo o conhecimento compartilhado diariamente.

Aos professores Abraham e Mariane que prontamente aceitaram o convite para participar da banca de defesa desta dissertação.

A todos os meus amigos pelo companheirismo, pelo apoio mesmo nos momentos mais difíceis e pelas risadas compartilhadas.

À minha namorada Kyunga, por sempre me incentivar e ser um exemplo de determinação.

A todos os professores do Programa de Engenharia Elétrica da Coppe por seus valiosos ensinamentos.

Ao povo brasileiro, por custear meus estudos com o seu trabalho.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

COMPARAÇÃO ENTRE TÉCNICAS DATA-DRIVEN E BASEADAS EM REGRAS APLICADAS A ALGORITMOS DE PROCESSAMENTO DA LINGUAGEM NATURAL PARA SÍNTESE DE VOZ EM PORTUGUÊS BRASILEIRO

Luiz Felipe Santos Vecchietti

Abril/2017

Orientador: Fernando Gil Vianna Resende Junior

Programa: Engenharia Elétrica

Nos últimos anos, devido ao grande crescimento no uso de computadores, assistentes pessoais e smartphones, o desenvolvimento de sistemas capazes de converter texto em fala tem sido bastante demandado. O bloco de análise de texto, onde o texto de entrada é convertido em especificações linguísticas usadas para gerar a onda sonora final é uma parte importante destes sistemas. O desempenho dos algoritmos de Processamento de Linguagem Natural (NLP) presentes neste bloco são cruciais para a qualidade dos sintetizadores de voz. Conversão Grafema-Fonema, separação silábica e determinação da sílaba tônica são algumas das tarefas executadas por estes algoritmos.

Para o Português Brasileiro (BP), os algoritmos baseados em regras têm sido o foco na solução destes problemas. Estes algoritmos atingem boa performance para o BP, contudo apresentam diversas desvantagens. Por outro lado, ainda não há pesquisa no intuito de avaliar o desempenho de algoritmos *data-driven*, largamente utilizados para línguas complexas, como o inglês. Desta forma, expõe-se neste trabalho uma comparação entre diferentes técnicas *data-driven* e baseadas em regras para algoritmos de NLP utilizados em um sintetizador de voz. Além disso, propõe-se o uso de Sequence-to-Sequence models para a separação silábica e a determinação da tonicidade.

Em suma, o presente trabalho demonstra que o uso de algoritmos *data-driven* atinge o estado-da-arte na performance dos algoritmos de Processamento de Linguagem Natural de um sintetizador de voz para o Português Brasileiro.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

COMPARISON BETWEEN RULE-BASED AND DATA-DRIVEN NATURAL LANGUAGE PROCESSING ALGORITHMS FOR BRAZILIAN PORTUGUESE SPEECH SYNTHESIS

Luiz Felipe Santos Vecchietti

April/2017

Advisor: Fernando Gil Vianna Resende Junior

Department: Electrical Engineering

Due to the exponential growth in the use of computers, personal digital assistants and smartphones, the development of Text-to-Speech (TTS) systems have become highly demanded during the last years. An important part of these systems is the Text Analysis block, that converts the input text into linguistic specifications that are going to be used to generate the final speech waveform. The Natural Language Processing algorithms presented in this block are crucial to the quality of the speech generated by synthesizers. These algorithms are responsible for important tasks such as Grapheme-to-Phoneme Conversion, Syllabification and Stress Determination.

For Brazilian Portuguese (BP), solutions for the algorithms presented in the Text Analysis block have been focused in rule-based approaches. These algorithms perform well for BP but have many disadvantages. On the other hand, there is still no research to evaluate and analyze the performance of data-driven approaches that reach state-of-the-art results for complex languages, such as English. So, in this work, we compare different data-driven approaches and rule-based approaches for NLP algorithms presented in a TTS system. Moreover, we propose, as a novel application, the use of Sequence-to-Sequence models as solution for the Syllabification and Stress Determination problems.

As a brief summary of the results obtained, we show that data-driven algorithms can achieve state-of-the-art performance for the NLP algorithms presented in the Text Analysis block of a BP TTS system.

Contents

List of Figures

List of Tables

List of Symbols

C_x	Cell State, p. 19
W_x	Weight vector, p. 19
η	Number of the occurrences for the M-gram, p. 16
ψ	Word included in the training lexicon, p. 15
φ	Phonemic Transcription, p. 14
ϑ	Model parameters, p. 15
b_x	Bias vector, p. 19
e	Expected number of occurrences, p. 16
h_x	Previous state, p. 19
x_t	Input vector, p. 19
G	Set of graphemes, p. 14
P	Set of phonemes, p. 14
Q	Set of Graphones, p. 14
S	Set for all the possible segmentations, p. 15
g	Word, p. 14
h	Sequence of preceding graphones, p. 16
q	Graphone, p. 14

List of Abbreviations

BPTT	Backpropagation Through Time, p. 20
BP	Brazilian Portuguese, p. 1
DNN	Deep Neural Network, p. 18
EM	Expectation-Maximization, p. 15
EP	European Portuguese, p. 12
FSA	Finite-State Acceptor, p. 17
FST	Finite-State Transducer, p. 12
G2P	Grapheme-to-Phoneme Conversion, p. 1
GRUs	Gated Recurrent Units, p. 19
HMM	Hidden Markov Model, p. 17
LSTM	Long Short Term Memory, p. 12
NLP	Natural Language Processing, p. 1
PER	Phoneme Error Rate, p. 40
RNN	Recurrent Neural Networks, p. 12
SPSS	Statistical Parametric Speech Synthesis, p. 23
TTS	Text-to-Speech, p. 1
WER	Word Error Rate, p. 2
WFSA	Weighted Finite-State Acceptor, p. 17
WFSTs	Weighted Finite State Transducers, p. 2

Chapter 1

Introduction

1.1 Motivation

Due to the exponential growth in the use of computers, personal digital assistants and smartphones, the development of Text-to-Speech (TTS) systems have become highly demanded during the last years. TTS systems are able to convert a written form of communication, text, in a spoken form of communication, speech. Some important applications of these systems include spoken dialogue systems, GPS navigation systems, reading aloud applications for people with vision impairments, audiobooks, communicative robots and speech-to-speech translation systems.

A TTS system is divided in two parts, a *front-end* part that consists in the Text Analysis block and a *back-end* part that is the Speech Waveform generation block, as can be seen in Figure 1.1. The Text Analysis block is the main focus of this dissertation and is responsible for converting the input text inserted by the user to the required linguistic specifications. These linguistic specifications are used as inputs to the Speech Waveform generation part. To do this conversion the Text Analysis block processes the input text through a series of Natural Language Processing (NLP) algorithms. These algorithms are responsible for important tasks such as the Grapheme-to-Phoneme conversion (G2P) , Syllabification and Stress Determination.

For Brazilian Portuguese (BP) many solutions for the algorithms presented in the Text Analysis part of TTS systems have been proposed [1–8]. Because BP is a language with shallow orthography the solutions have been focused in rule-based approaches. These algorithms perform well for BP but have many disadvantages. The rules must be designed by a linguistic specialist, which is really a time consuming and expensive task. Also, it is hard to incorporate these rules to the new words incorporated to the language as time goes by. So, for deep orthography languages, such as English, new data-driven approaches have been developed, especially for the G2P task, that is a really important part of Speech Recognition and Speech Synthesis

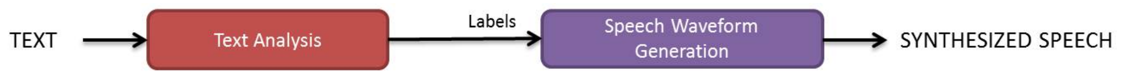

Figure 1.1: Basic architecture of a TTS system.

systems. Data-driven algorithms are much less time-consuming to develop and can easily adapt to the incorporation of new words to the desired language. Another advantage of these systems is that they are language-independent, in other words, the same algorithm can be used for different target languages. The performance of these solutions have been improving drastically, especially with the increasing of hardware power and the development of new Machine Learning and Deep Learning algorithms.

Data-driven approaches, such as Joint Sequence Models [9], Weighted Finite-State Transducers (WFSTs) [10] and Sequence-to-Sequence Models [11] reach the state-of-the-art performance for English G2P conversion. On the other hand, for BP there is still no research to evaluate and analyze the performance of these approaches in comparison with rule-based algorithms already presented in the literature. As the quality of synthesized speech is highly related to the efficiency of the Text Analysis block, an analysis of different solutions for the NLP algorithms is needed.

Also, we use Sequence-to-Sequence Models [11] to propose richer feature sets as explained in [12] to solve the G2P conversion task. The accuracy of the model is then evaluated. Moreover, we also propose the use of Sequence-to-Sequence Models to the Syllabification and the Stress Marking problems evaluating the performance in comparison to the related rule-based algorithms.

As a brief summary of the results obtained, we show that data-driven algorithms can achieve state-of-the-art performance for the NLP algorithms presented in the Text Analysis block of a BP TTS system. The Sequence-to-Sequence Models achieve the best accuracy for the G2P conversion with a mean Word Error Rate (WER) of 2.28%. We also show that a model that includes the stress information outperforms the previous model achieving a mean WER of 1.91%. For the Syllabification and the Stress Marking the proposed models achieved a mean WER of 1.70% and 0.45% respectively. The rule-based algorithms achieve mean WER equal to 3.26% for G2P Conversion, 4.15% for Syllabification and 0.03% for Stress Determination.

Given the motivation presented in this section, the objectives of this work are presented below.

1.2 Objectives

The proposal of this dissertation is to analyze and compare rule-based and data-driven approaches to the implementation of Natural Language Processing algorithms used in the development of TTS systems. So, the following specific goals are defined:

- Propose a new set of rules for Brazilian Portuguese based on the rule-based algorithm showed in [6], and evaluate the performance compared to the previous algorithm.
- The development of three lexicons composed of 33737 words extracted from the Ispell [13] dictionary used for the training step of data-driven approaches. The first lexicon has the G2P information, while the second one contains the syllable information and the third one contains the stressed syllable information.
- Propose a Sequence-to-Sequence Model approach [14] for the Syllabification and the Stress Marking problems.
- Compare the performance of the new set of rules proposed and three data-driven approaches for the G2P Conversion using k-fold cross validation [15].
- Compare the performance of the rule-based algorithms and the Sequence-to-Sequence Model proposed for the Syllabification and Stress Marking tasks using k-fold cross validation [15].
- Analyze the performance of the Sequence-to-Sequence Models for the G2P Conversion using the stress marking information in the feature set of the model compared with models that do not use this information.
- Analyze the influence of the training set size on the performance of Sequence-to-Sequence Models for the G2P Conversion, Syllabification and Stress Determination.

1.3 Description

The rest of this dissertation is organized as follows.

Chapter 2 reviews the theoretical background of TTS systems and presents the state-of-the-art techniques for the NLP algorithms used in this dissertation. The main advantages and disadvantages of each method are pointed out.

In Chapter 3, a new set of rules for the BP G2P Conversion is proposed and its accuracy is showed.

Chapter 4 presents the new Sequence-to-Sequence Models proposed in this dissertation. The first model adds the stress marking information to the G2P Conversion Model. The second and third model addresses the Syllabification and Stressed Syllable information respectively.

The Chapter 5 describes the experiments for the comparison of rule-based approaches and data-driven approaches for the BP NLP algorithms.

Finally, Chapter 6 brings the conclusions of this work, a summary of the achievements and the research directions for future works.

Chapter 2

Background

2.1 Introduction

This chapter aims to explain the theory behind TTS systems. First of all, it is interesting to recapitulate the brief explanation from the Chapter 1 about the basic idea of a TTS System. The objective of TTS systems is to generate speech given a text input. The desire is to generate intelligible and natural-sounding speech. It means, the user should clearly understand the sentence that was spoken by the synthesizer and it should sound as natural(human-like) as possible. As can be seen in [16] this conversion from words in written form into speech is nontrivial.

The TTS system is divided in two fundamental processes: Text Analysis (*front-end*) and Speech Generation (*back-end*), as already mentioned in Chapter 1. It can be explained using as an example the human task of reading a message aloud. This task can be understood as a decoding-encoding problem [17]. If we see both speech and writing/text as signals which encodes a message the task can be divided in two parts. In the first part, the written signal is decoded into a message and then, in the second part the message is then re-encoded into a speech signal.

During the last years, with the focus of the research projects towards the *back-end* part we were able to advance drastically especially regarding the naturalness of synthetic speech. Nevertheless, to keep the high intelligibility of these systems and to allow the development of new applications, such as Expressive Synthesizers, the development of reliably Text Analysis systems are crucial. Furthermore, we are going to pass through the whole development process of a TTS system in the next sections.

Thus, this chapter will present the background to understand a TTS system. It will first introduce the Text Analysis block explaining the steps and the state-of-the-art algorithms used in each part of the processing. The algorithms used to perform the comparison in this dissertation will be explained in detail. Then, the state-of-the-art solutions for the Speech Waveform generation block will be introduced.

2.2 Text Analysis

A TTS system should be able to accept any written input. After, the input inserted by the user should be processed to create the labels that serves as the input message for the speech signal generation. This processing is done by the Text Analysis block and consists in many different steps showed in Figure 2.1. As you must be thinking, the Text Analysis block is language-dependent, so, if you are developing a new synthesizer this block should be specific for your target language. Despite of this fact, many of the algorithms presented in the Text Processing part share the same objectives between different languages.

Therefore, looking to Figure 2.1 we are going to analyze each step of the Text Analysis block. The first step, before enter in the Normalization block, consists in separate the input text inserted by the user in sentences and words. After that, the new input is ready to enter a complex normalization step. This is necessary because the input contain information that can be troublesome for the Speech Synthesizer to read. For example, if the user inserted his birthday in date format as input how the system should read a date? Due to these problematic inputs, this step is responsible for normalizing numbers, dates, acronyms, initialisms and abbreviations. The numbers, dates and abbreviations should be expanded into full words. For example "Dr." should be expanded to "Doutor" (*doctor*) and "10" should be expanded to "Dez" (*ten*). The initialisms and acronyms need to be dealt with two different possibilities. The initialisms, called "Siglas" in BP, are read by spelling one letter at a time. One example is how you read the abbreviations for the Federal Universities in Brazil, such as "UFRJ", read one letter at a time. On the other hand, the acronyms, called "Acrônimos" in BP, are read as words. As an example we can cite the "NASA", National Aeronautics and Space Administration. After finish this step, the output is inserted in the Text Processing block.

The Text Processing block is the place where the NLP algorithms analyzed and compared in this dissertation are located. The first one consists of the Homograph Disambiguation algorithm. Homographs are pair of words that share the same text form. The homographs become a problem for speech synthesis when they differ in pronunciation, called "Homógrafos Heterófonos" in BP. For example the word "jogo" that can be pronounced with the open vowel "j[O]go" and means the verb *to play* when conjugated with the first person or can be pronounced as "j[o]go" with a close vowel and means *game* as a noun. These words should be treated by a special algorithm before the Grapheme-to-Phoneme Conversion step because they can insert errors in the phonetic transcription. Moreover, the synthesizer will pronounce the

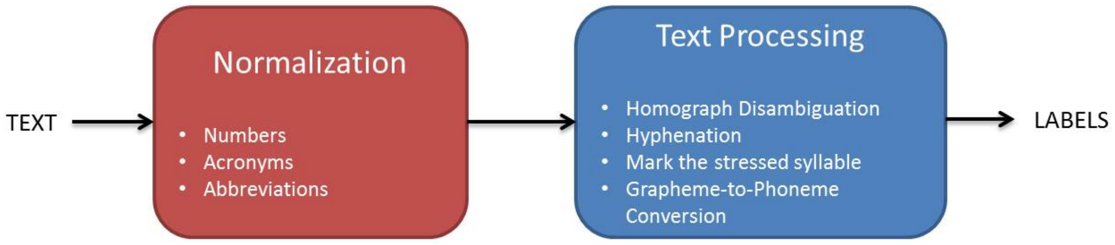

Figure 2.1: Details of the Text Analysis component.

word with the wrong phoneme during synthesis time and influences its intelligibility and naturalness. Another ambiguity that is common for specific languages is the syntactic ambiguity, that is when two sentences are written with the same text form but can have two different interpretations. This ambiguity is not treated in the development of TTS systems because of the need of complex semantic analysis and its little influence to the phonetic transcription, despite of being an important issue regarding prosodic aspects.

Another algorithm presented in the Text Processing block is the Stress Determination one. The algorithm will mark the syllable that contains the stress in a word. The stress will influence the pronunciation and especially the prosody related to the speech. The stress mark is highly correlated to the rise of the fundamental frequency, intensity and duration of the vowels [6]. So, the insertion of the stress information as one of the linguistic specifications that are used to generate speech is important to enhance the naturalness of the system. In addition to, this information will also be used on the rule-based G2P converters that will be presented in the next section and in Chapter 3. It also will be inserted in the Sequence-to-Sequence Models with a richer feature set proposed in Chapter 4 of this dissertation.

The Syllabification algorithm is another important step during Text Processing. In this algorithm, given a word, the algorithm divides the word in smaller pieces called syllables. These smaller pieces are considered the *building blocks* of a word [18]. The syllable information with the stress marking especially influences the rhythm and prosody of languages, and, consequently, of the speech synthesizers. This influence is even more important for BP because of its tendency to sound as a *syllable-timed* language. Then, for BP, the syllables directly affects the duration and pronunciation of a word.

The most important task during the Text Processing is the Grapheme-to-Phoneme Conversion. The G2P converter is responsible for guessing a word's pronunciation from its spelling [17]. It means, converting the word into their correspondent phoneme forms. Note that errors in this conversion will directly affect the intelligibility as the words will be pronounced wrongly by the synthesizer. It is important to mention that there can be more than one phoneme set for the same language, so the most suitable one for your application can be chosen. Usually, dictionary look-ups can be used to solve this problem. In this case, these dictionaries are a list of many words and their respective phoneme sequences. Words that are not presented in this dictionary need to be inferred using different solutions. These solutions depends on the target language and the project requirements. For deep orthography languages, such as English, where there is no correspondence between the graphemes and phonemes, even for experienced linguists, is hard to find patterns that can be transformed in linguistic rules. So, the development of data-driven approaches have become a need. On the other hand, for shallow orthography languages, such as Spanish and BP, the strong correspondence between graphemes and phonemes encouraged the use and the development of rule-based approaches by linguists.

With the description of these components, in the next sections it is time to explore the solutions used to solve the Syllabification, the Stress Determination and the G2P conversion algorithms. First, in Section 2.2.1, rule-based approaches that represents the state-of-the-art for BP will be presented. Then, Section 2.2.2 will present the state-of-the-art solutions for these problems using data-driven approaches.

2.2.1 Rule-based approaches

Despite of the big growth in the development of data-driven strategies to solve Natural Language Processing problems recently, rule-based approaches are still widely used in the production of NLP systems. Rule-based approaches have been the focus on the solutions of NLP problems for BP. Because of the phonological regularity of Brazilian Portuguese, rule-based approaches achieve good accuracy for problems like the G2P conversion, Syllabification and Stress Determination. In addition to, they can be more efficient computationally than data-driven approaches that require complex training and decoding algorithms. The running time of these algorithms can turn their use in real time for specific applications impossible. However, the development cost of the rules by a linguistic expert usually is high, and is hard to keep the rules updated to changes in the language or to create rules for foreign words and proper names, so that, they need to be treated as exceptions. Also, after you have hundreds of rules, they start to interact in complex ways turning even more difficult to keep the system maintenance. Another important application of the rule-based systems is accelerating the development of new corpora for the data-driven approaches training presented in Chapter 5.

This chapter will introduce the rule-based algorithms that can be found in the literature for BP. Moreover, the algorithms chosen to perform the experiments in

Chapter 5 will be briefly described.

Stress Determination Algorithm

Beyond the importance of stress information for the prosodical features of BP, this information is also used at the rule-based algorithms used for G2P conversion and Syllabification. Therefore, the accuracy of the stress determination algorithm also influences the accuracy of the other ones. Rule-based algorithms for BP are presented in [5, 6]. The algorithm used in this dissertation is briefly explained next and is detailed in [6]. It will be used for the analysis in later chapters.

The algorithm consists in 19 subsequential rules. A important information for the algorithm is the presence of accents in BP words. These accents are highly correlated with the stress determination in a word. Because of that, the first rule of the algorithm searches the presence of these accent markers. If the word has more than one marker, a precedence rule is involved. Then, the word is analyzed from the last to the first character. After a word is matched to a rule, the stress is determined and the next word is evaluated. Some words, like prepositions, pronouns and conjunctions do not have a stressed syllable. So, they need to be treated as exceptions to the algorithm rules.

Syllabification Algorithm

A speech synthesizer, to achieve a good performance, also need to include the information of the characteristics of the desired language. One of this important information for BP is the division of the word in syllables. The syllables influences the rhythm of speech in BP. The addition of this information to the contextual information of a speech synthesizer improves its naturalness. This problem have been mainly approached using rule-based strategies for BP. In [5, 6] a rule-based algorithm was proposed to the syllabification problem. A few modifications to this algorithm were proposed in [7, 8]. These modifications consisted especially to additional rules to treat diphtongs and hyatus cases. In this dissertation, the analysis showed in the next chapters will be based on a implementation of the algorithm presented in [6]. The algorithm is explained succinctly subsequently.

The syllables of the BP language are formed around a vowel. So, at first, the algorithm looks for the existing vowels in the word. Thereafter, each vowel is analyzed looking to its position inside the word and the characters located to its right and to its left. With this information, the rules try to huddle together the vowel with their neighbour characters to create a syllable based on eight possible cases. After the vowel is matched to one of the 25 rules presented in the algorithm the syllable is created and the next vowel can be evaluated. The rules first investigates the instances where the vowel is located at the beginning of the syllable, then the cases where the vowel is located in the middle or at the end of the syllable are analyzed. The detailed rules can be seen in [6].

Grapheme-to-Phoneme Conversion Algorithm

For the development of unrestricted TTS systems is important that there exists an algorithm capable of derive the pronunciation for any input word. For languages with shallow orthography, such as BP, a good option to solve this problem is using rule-based algorithms. Using this strategy the Grapheme-to-Phoneme conversion can be done by a series of rewrite rules. As mentioned earlier, these rules need to be designed by linguistic experts and are context dependent [19]. One way of representing the rules is given by the following form presented in [20]: $A\ [B]\ C \rightarrow D$, where A and C are surrounding contexts, B is the letter being converted and D is the target phoneme. Rules can use another linguistic characteristics or outputs from other Text Processing algorithms to increase the accuracy of the G2P conversion. Some of these characteristics are stress markers, syllable boundaries and part-of-speech tags.

An interesting point to mention is that even for the same language, we can have more than one rule-based system with different characteristics. The first characteristic to mention is the pronunciation. The pronunciations can vary from region to region inside a country, so, different set of rules are created for different systems. Another point is that different systems can also have different phonemic alphabets. For example, a rule-based system designed to language learning can have distinct sets compared to a system designed to a Computational Linguistics task, such as the NLP block of Speech Recognition and Speech Synthesis systems.

The basic idea of rule-based G2P algorithms works as follows. The input word is evaluated from left to right character by character. For each character, a set of rules is evaluated until a rule is matched. The rules are applied in a specific order, determined by a linguistic specialist. The first rules are the most specific ones, and the last one is usually a generalization case. Every time a rule is matched, a phoneme for that character is set and the search window is shifted the number of characters established by the rule. After scan all the characters the correspondent phoneme form for the word is found.

Rule-based algorithms for BP are presented in [2, 6, 21]. In [21] the rule-based algorithm is used to develop a language learning system. Due to this fact, the phonemic set is different from the one used in [2, 6]. The [2, 6] algorithms were developed to the development of TTS and Computational Linguistics applications, so, an implementation of [6] was chosen as the base algorithm for this dissertation. After a performance analysis of this algorithm, it was verified that the accuracy could be enhanced with the addition of new rules. Thus, a new set of rules based on [6] is proposed in Chapter 3. The proposed set of rules improves the performance of the algorithm from 97.44% to 99.18% when evaluated regarding Phoneme Error Rate and will be used to perform the comparison with data-driven solutions in Chapter 5.

2.2.2 Data-Driven approaches

Data-driven approaches had a big growth in the Natural Language Processing research recently. The interest in the development of data-driven solutions can be explained if we start to analyze the drawbacks of the rule-based systems that we have presented in the last section. The need of linguistic experts to develop and maintain the rules updated really increases the system cost. In addition to, for some specific languages, like English, because of its deep orthography nature, it is not easy to find rules and patterns, so rule-based systems are not a good choice. Another reason to think about is the language dependency of the rule-based systems. If you need to develop a rule-based system for a new language for the first time you need to develop all the rules from scratch. Considering that, there exists a big interest in the last years for language independent data-driven solutions. So, to analyze these solutions we will start understanding the idea of a data-driven approach.

The main idea behind the data-driven approach is to use a lexicon with many examples to perform the training of a model for a specific task. During the training, the system should be able to train a model purely by analyzing patterns on the data. With the model trained, then, it should be able to predict the output of this task for a previously unseen input. So, the system do not need to have any handcrafted rules. It just need to have a lexicon with examples for that task. It is so much easier for a native speaker to provide examples than to design linguistic rules [9]. Moreover, if just a lexicon is required, at first, we can deduce that independently of the language it should work at a reasonably level. To achieve the desired accuracy for a particular application, the lexicon provided should be representative of that application domain. For example, if we want to design a GPS system, we should provide as much examples as possible of sample phrases used to show directions, give information and that contains the name of streets. For NLP algorithms, not all of the data can be fit in exact patterns, and exceptions exists. So, a good algorithm should also be able to handle exceptions implicitly. This fact will be further analyzed during Chapter 5. As a summary of the problem, the objective of a data-driven algorithm is to find a model that minimizes the error on *seen* data. And using this model to achieve the desired accuracy for *unseen* data.

Before, we pointed out advantages of data-driven solutions compared to rule-based approaches, especially focusing on the fact that it just need a lexicon for training. In addition to, its language independency is an important advantage of these systems. But, it is also worthy noting that the data-driven solutions also have disadvantages, especially regarding to its latency implementing TTS systems. These approaches usually need a decoding algorithm that is computationally expensive and needs to be evaluated before implementing real time applications. Also, the training time depending on the strategy and the lexicon size used can vary from minutes to days. [22] shows its concern and develop new algorithms that address the latency problem when using Long Short Term Memory - Recurrent Neural Networks (LSTM-RNNs) as the acoustic model of a TTS system. If the TTS system also uses data-driven approaches in the Text Analysis block these concerns should be expanded and further explored during the development step.

For BP there is a lack of references for Natural Language Processing algorithms that uses data-driven approaches in the Text Analysis component showed in Chapter 1. If we extend our research to European Portuguese (EP) we can find a few works that explore the development of data-driven NLP algorithms. In [23] a Maximum Entropy probabilistic framework to the G2P conversion, Stress Marking and Boundary Prediction is presented. A neural network based G2P to convert EP common and proper names is showed in [24]. Another Machine Learning solution for EP G2P conversion is presented in [12]. This paper is interesting because it proposes a richer featured set for G2P systems, a concept that we will further explore in Chapter 5. At last, in [25] a G2P using Finite-State Transducers (FSTs) is implemented. This system is compared with a rule-based approach and, moreover, a hybrid approach linking both techniques is developed and evaluated. Therefore, an analysis of the use of data-driven solutions for BP NLP algorithms is important, especially data-driven solutions that are proven to achieve state-of-the-art results for a complex language such as English.

The next subsections will, first of all, introduce the G2P conversion, Syllabification and Stress Determination problems by a Machine Learning perspective. Then, the theory behind the strategies chosen for the comparison with the rule-based algorithms are explained and detailed.

Grapheme-to-Phoneme Problem

The G2P conversion is the most important problem between the ones we will present in this dissertation. Therefore, there exists a lot of research and different data-driven solutions proposed. The problem consists in predict the pronunciation of a word and it can be seen as a mapping from graphemes to phonemes for the desired language. In Figure 2.2 an example of the G2P conversion for the word “abacate” (avocado) can be seen. In a “perfect” language, this mapping would be one-to-one, it means, each grapheme would be the representation of one phone/sound of the language.

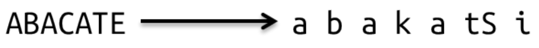

Figure 2.2: Grapheme-to-Phoneme Conversion Example.

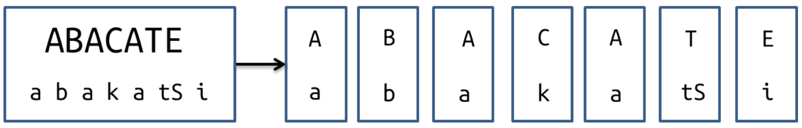

Figure 2.3: Alignment Example.

But, what happens in real is that this is not true, and this mapping is complex and highly dependent on the language style. This task even become more complex with the globalization and the insertion of foreign names and words in most of the languages spoken in the world.

Approaching this problem by a Machine Learning perspective, this mapping can be solved in different ways. For some strategies an alignment step is needed. In this alignment step each grapheme is associated to zero or more phonemes. An example of alignment for the word "Abacate" (avocado) is showed in Figure 2.3. The main strategies to solve the G2P conversion can be divided in three classes, as pointed out in [9], and briefly explained next:

Techniques based on local classification Usually this technique requires the alignment step. Some recent solutions can overcome the need of the alignment step, as the Sequence-to-Sequence Models [11]. For these techniques the decision is made one position at a time. For each word character, based on its context, a phoneme is chosen from the desired language phoneme set. Some examples of these type of solutions include neural networks [26–28] and decision trees [29, 30].

Pronunciation by Analogy This technique scan the lexicon searching for parts of words similar to the unknown word to be transcribed. The analogous parts are then linked to form the unknown word, creating a lattice of possible candidates. Then, the best path is determined to create the output pronunciation. In [31, 32] different solutions for this strategy are presented.

Probabilistic approaches These strategies approach the G2P conversion by a probabilistic perspective. In [33] a conditional maximum entropy model is used for predicting the pronunciation. Another solution presented in [34] find the conditional probability of a phoneme based on the current and previous letter and phonemes using a 7-gram model and the Bayes' Formula.

For this work, we chose three techniques that achieve a good performance to do our comparison. These techniques will be explained in the next subsections.

Joint-Sequence Models

The Joint-Sequence Model proposed by [9] is a probabilistic framework especially designed for the Grapheme-to-Phoneme conversion. In this subsection we will explain the theory of this technique focused on the understanding of the G2P task by a probabilistic perspective and the theory described in [9].

The G2P conversion task can be interpreted as a statistical problem. In this case, for a given word $\mathbf{g}$ we want to infer its most likely pronunciation φ (phonemic transcription). So, for a set of graphemes $\mathbf{G}$ and a set of phonemes $\mathbf{P}$ it can be formalized by the Bayes' decision rule as:

$$\varphi(\mathbf{g}) = \underset{\varphi' \in P^*}{\operatorname{argmax}}\, p(\mathbf{g}, \varphi') \tag{2.1}$$

where $\mathbf{g} \in G^*$ and $\varphi \in P^*$. To solve this problem, the Equation 2.1 should be optimized with respect to the Word Error, it means we are trying to increase the probability of get a correct pronunciation given a word as an input.

The Joint-Sequence Models idea is that the mapping from the grapheme (input) sequence to the phoneme sequence (output) can be done from a common sequence of joint units which carry both input and output symbols. These joint units are called "graphones". The graphones are a pair $q = (\mathbf{g}, \varphi) \in Q \subseteq G^* \times P^*$. In Figure 2.3 each block in the right side of the arrow can be considered a graphone with one grapheme and one phoneme. This kind of graphone with no more than one grapheme and one phoneme is called a *singular* graphone. So, it can be seen in Figure 2.3 that the word "Abacate" (avocado) can be generated using seven *singular* graphones. In the example, the size of the grapheme and phoneme sequences are the same, but they can differ depending on the word transcribed. Graphones with multiple inputs and multiple outputs are also used and are called *joint multigram.*

Thus, given a pair of input and output strings contained in the training lexicon, the system should be able to segment that pair into a graphone sequence $\mathbf{q}$. However, as can be seen in [9], this segmentation may not be unique. Consequently, this pair can be represented by more than one graphone sequences. Because of this fact, to calculate the joint probability for the pair (g, φ) we need to sum the probabilities of all the possible graphone sequences:

$$p(\mathbf{g}, \varphi) = \sum_{\mathbf{q} \in S((\mathbf{g}, \varphi))} p(\mathbf{q}) \tag{2.2}$$

where $\mathbf{q} \in Q^*$ and $S(\mathbf{g}, \varphi)$ is the set of all the possible segmentations for the pair $(\mathbf{g}, \varphi)$. Boundary symbols are inserted at the beginning and at the end of each graphone $\mathbf{q}$. For a specific $\mathbf{q}$, the set $S(\mathbf{g}, \varphi)$ is given by:

$$S(\mathbf{g},\varphi) := \left\{ \mathbf{q} \in Q^* \;\middle|\; \begin{array}{l} \mathbf{g}_{q_1} \smile \cdots \smile \mathbf{g}_{q_K} = \mathbf{g} \\ \varphi_{q_1} \smile \cdots \smile \varphi_{q_K} = \varphi \end{array} \right\} \tag{2.3}$$

where $K = |q|$ is the length of the graphone sequence and $\smile$ means concatenation. With the joint probability distribution $p(\mathbf{g},\varphi)$ and the set $S(\mathbf{g},\varphi)$ defined we can see that the calculation is reduced to find p($\mathbf{q}$) over all possible graphone sequences $\mathbf{q} = q_1, \cdots, q_K$. This probability distribution can be modeled using a standard M-gram approximation:

$$p(q^K) \cong \prod_{j=1}^{K+1} p(q_j \mid q_{j-1}, \cdots, q_{j-M+1}) \tag{2.4}$$

With the model specified, we should analyze how to infer this model from the training set. It is worthy noting that at this point the training set was not previously segmented. So, the segmentation problem should be addressed during the model estimation. The training set is a lexicon containing N words with their respective pronunciations: $\psi_1, \cdots, \psi_N = (\mathbf{g_1}, \varphi_\mathbf{1}), \cdots, (\mathbf{g_N}, \varphi_\mathbf{N})$. Then, we can compute the probability of any segmentation ς and define a joint sequence as:

$$p(\mathbf{g},\varphi,\varsigma) = p(\mathbf{q}) \tag{2.5}$$

To calculate the log-likelihood of the training set we sum over all segmentations:

$$log\ L(\psi_1, \cdots, \psi_N) = \sum_{i=1}^{N} log\ L(\psi_i) = \sum_{i=1}^{N} log\ \left(\sum_{\varsigma \in S(\psi_i)} p(\psi_i, \varsigma) \right) \tag{2.6}$$

Because the training data is not segmented a priori, ς is a hidden variable. The training of the joint-sequence model can be done using the Expectation-Maximization (EM) algorithm as demonstrated in [35]. The re-estimation equations for a M-gram model with parameters ϑ are the following ones:

$$p(\mathbf{q};\vartheta) = \prod_{j=1}^{|\mathbf{q}|} p(q_j|h_j;\vartheta) \tag{2.7}$$

$$\begin{aligned} e(q,h;\vartheta) &:= \sum_{i=1}^{N} \sum_{\mathbf{q} \in S(\mathbf{g_i},\varphi_\mathbf{i})} p(\mathbf{q}|\mathbf{g_i},\varphi_\mathbf{i};\vartheta)\eta_{q,h}(\mathbf{q}) \\ &= \sum_{i=1}^{N} \sum_{\mathbf{q} \in S(\mathbf{g_i},\varphi_\mathbf{i})} \frac{p(\mathbf{q};\vartheta)}{\sum\limits_{\mathbf{q}' \in S(\mathbf{g_i},\varphi_\mathbf{i})} p(\mathbf{q}';\vartheta)} \eta_{q,h}(\mathbf{q}) \end{aligned} \tag{2.8}$$

$$p(q|h;\vartheta') = \frac{e(q,h;\vartheta)}{\sum\limits_{q'} e(q',h;\vartheta)} \tag{2.9}$$

where h is the sequence of preceding graphones $h_j = (q_{j-M+1}, \cdots, qj-1)$, $\eta_{q,h}(\mathbf{q})$ define the number of occurrences of the M-gram $q_{j-M+1}, \cdots, q_j$ in q and $e(q,h;\vartheta)$ is the expected number of occurrences of the graphone q in the training sample under the current set of parameters ϑ. The Equation 2.8 is calculated using a forward-backward procedure as described in [36].

The estimation done with the Equation 2.9 tend to have an overfitting problem. It means that the model will have a low error after finish the model training but when needed to predict *unseen* data the accuracy will be not good. To solve these problems, [9] also proposes a smoothing and a trimming algorithm. It also shows how the parameters must be initialized.

Finished the training, Equation 2.1 can be used to decode new words. To do the G2P Conversion we approximate the sum of Equation 2.2 by the maximum probability:

$$p(\mathbf{g},\varphi) \approx \max_{\mathbf{q}\in S((\mathbf{g},\varphi))} p(\mathbf{q}) \tag{2.10}$$

Therefore, given the input word, the most likely graphone sequence should be calculated and the output should be the respective phoneme sequence:

$$\varphi(\mathbf{g}) = \varphi\left(\underset{\mathbf{q}\in Q^*|\mathbf{g}(\mathbf{q})=\mathbf{g}}{\operatorname{argmax}} p(\mathbf{q})\right) \tag{2.11}$$

The joint-sequence modeling is considered the standard probabilistic technique for the G2P conversion task. An important feature of this solution is the ability to handle a training set without the need of previous alignment. [9] also shows that the finite length characteristics of the joint sequence models allow their representation by Finite-State Transducers. In the next subsection we will present a technique that uses some of the Joint-Sequence Models ideas to present a modified approach exploiting the alignment information and Weighted Finite-State Transducers.

WFST-based

The Finite-State Transducers are automata with a finite number of states. The transitions between states are labeled with an input/output pair. Therefore, the transducer path performs a mapping from an input symbol sequence to an output symbol sequence [37]. The insertion of *weights* in the state transitions of the FSTs is an important addition to these systems. The weights works as probabilities or penalties between the possible paths of the transducer. So, you can calculate the

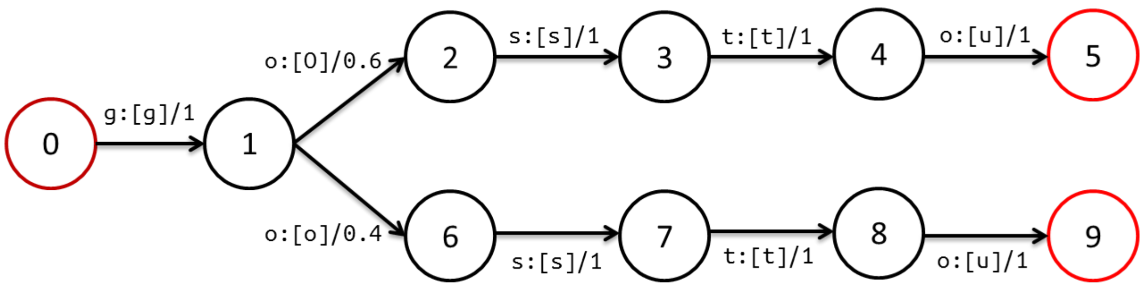

Figure 2.4: A possible WFST to the G2P conversion of the word "gosto" in BP.

overall weight and the best possible path to the mapping. Hidden Markov Models (HMMs), tree lexicons and N-gram language models can be represented as WFSTs [37]. So, WFSTs are a flexible and computationally efficient way to solve problems that can be translated into probabilistic finite-state models. A simple example of a WFST model for the G2P conversion task can be seen in Figure 2.4.

The applicability of WFSTs to Grapheme-to-Phoneme conversion was first proposed in [25]. It was justified by the flexibility of WFSTs and by the possibility of integration between the information provided by other "Text Analysis" blocks and the model [25]. WFSTs also make easier the development of hybrid approaches combining data-driven and knowledge-based solutions. The first step of the WFST-based G2P consists on the alignment of the training lexicon. After that, we have aligned sequence pairs for each word in the training lexicon $\psi_i = (g_1, \cdots, g_n), (p_1, \cdots, p_n)$. These aligned sequence pairs are then converted to sequences of aligned pairs $\psi_i = (g_1 : p_1, \cdots, g_n : p_n)$. This alignment can be multiple-to-multiple, it means that multiple graphemes can be aligned with multiple phonemes. Techniques to perform the alignment step are presented in [10, 38, 39].

Next to the alignment is the model training. In this dissertation, we explain the approach presented in [10] where a joint M-gram formulation is used. From the aligned pairs $(g_1 : p_1, \cdots, g_n : p_n)$ M-gram models will be trained. This model is interpreted as the probability of a grapheme matching a particular phoneme based on the previous (M-1) aligned pairs [25]. Then, the model is first converted to a Weighted Finite-State Acceptor (WFSA) and then to a Weighted Finite-State Transducer where each aligned pair is converted into an input/output label. The inputs are the grapheme labels and the outputs are the phoneme labels. Thus, terminating the model training.

To decode a new word we should compute the shortest path from a weighted list of pronunciation hypothesis. First of all, the word inserted is compiled into a Finite-State Acceptor (FSA) that provides a list of all the possible grapheme subsequences. Then, a WFST is constructed from the joint G2P M-gram model. The pronunciation hypothesis can then be calculated by operations using the FSA and the WFST constructed before. The shortest path is chosen as the best phonemic transcription.

Sequence-to-Sequence Models

This subsection represents the first place where Sequence-to-Sequence Models are presented in this dissertation. So, before explore the applicability of the models to the Grapheme-to-Phoneme conversion, we will explore the theory behind this approach. This will be useful when we propose this technique for the solution of the Syllabification and Stress Determination tasks in Chapter 4.

The Sequence-to-Sequence Models came to fill a gap in the Deep Neural Networks (DNNs) research. The DNNs worked well for numerous kind of problems where large training sets were available but could not map sequences to sequences. One of the problems DNNs had to face to solve sequence mapping problems was the inability of the architecture to handle variable size inputs and outputs. Therefore, the Sequence-to-Sequence Models were proposed in [14] as a domain-independent method to solve these types of problems. In [14] the method developed is applied to the Machine Translation task, and outperforms mature strategies used to traditionally solve this problem. Consequently, the use of this method for other challenging sequence-to-sequence problems was suggested to also achieve good performance. This method can be used in a variety of tasks that include Image Captioning Systems, Question and Answering Systems and Grapheme-to-Phoneme conversion, that is our main focus here.

To understand how the method works, first of all, a brief introduction of Recurrent Neural Networks is necessary. RNNs are a class of neural networks where each unit is directed connected to every other unit, allowing information to persist. So, they can use this connected structure to act like an internal memory to process arbitrary sequences of inputs. This nature of RNNs is intimately related to the concept of problems that consists of sequences and lists. The problem with the basic architecture of RNNs is that they have a long-term dependency problem. For tasks that requires memories of events that happened long time steps before it doesn't work properly. To solve this problem, a special kind of RNN called Long Short-Term Memory [40] (LSTMs) were developed. The classic architecture of a LSTM is presented in the Figure 2.5. The main idea of the LSTM architecture is the existence of a cell state (a vector that can be modified or not in each time step) controlled by gates that are able to carry information along [41]. If the actual state have important information to be inserted in the cell state, the gates are active and let information get through. The gates can also remove information from the cell state that lost its relevance during the chain. The LSTM updating equations are presented in Equation (2.12).

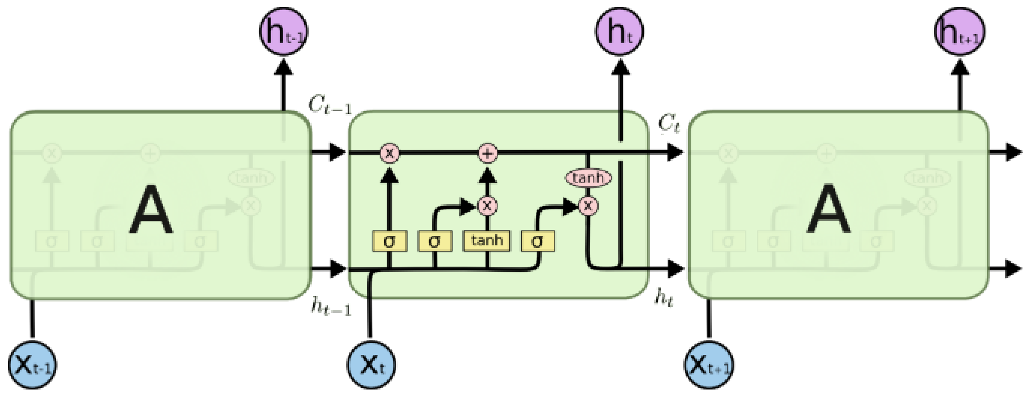

Figure 2.5: LSTM chain [41].

$$C_t = \sigma(W_f.[h_{t-1}, x_t] + b_f) * C_{t-1} + \sigma(W_i.[h_{t-1}, x_t] + b_i) * tanh(W_C.[h_{t-1}, x_t] + b_c)$$
$$h_t = \sigma(W_o.[h_{t-1}, x_t] + b_o) * tanh(C_t) \tag{2.12}$$

where the W_x are weight vectors, b_x are bias vectors and σ is the sigmoid function. Another variations on the LSTM architecture that tackle the long-term dependencies can also be used. Some of these variations include Gated Recurrent Units (GRUs) [42] and Depth Gated LSTM [43], that achieve similar performances as the architecture showed in Figure 2.5 [44].

The Sequence-to-Sequence Models acts like an encoder-decoder RNN. First, it does a RNN chain with the input sequence, reading one time step at a time. The input, for each time step, for example, is a word for a Machine Translation system and a character to a Grapheme-to-Phoneme converter. To each possible input, an index (positive integer) is assigned. Each possible index is converted to a dense vector of a fixed size during the embedding step. This vector is used as the input of the RNN architecture chosen. At the end of the encoder chain, a large fixed-dimensional vector representation of the input data is obtained. Independently of the size of the input sequence, this vector will have the same size. The second RNN chain, with the output sequence, works like a language model, but its conditioned on the vector that was the output of the inputs chain. A block diagram of this structure can be seen in the Figure 2.6, where each block represents a RNN architecture block, such as LSTMs or GRUs. A great power of this model is the possibility to insert another feature information in the input chain without modify the architecture of the network. This flexibility will be explored when we propose the insertion of stress markers for the G2P conversion in Chapter 4.

The Sequence-to-Sequence Models were used successfully to the task of Machine

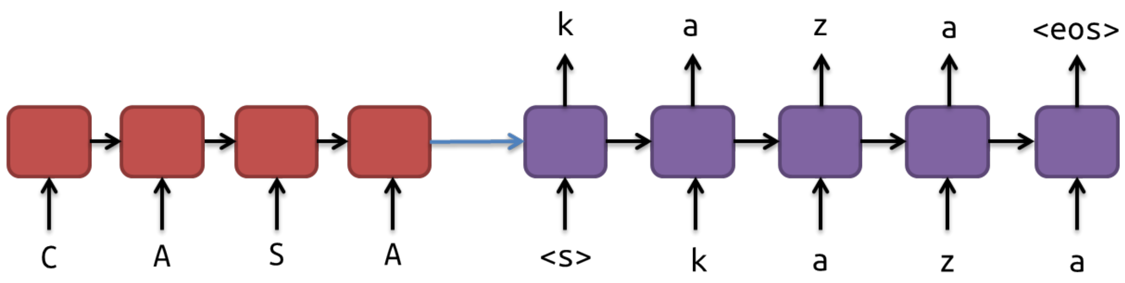

Figure 2.6: Sequence-to-Sequence Model for the G2P conversion.

Translation as showed in [14]. However, we are interested in the G2P conversion. So, in [11] it explores the applicability of Sequence-to-Sequence Models to the G2P task. The input sequence in this case is the word to be converted. The input is read one character at a time until its end to form the fixed-dimensional vector. The output RNN is used to decode the phoneme sequence from this vector. This model is showed in Figure 2.6 for the conversion of the word "casa" (house). In Figure 2.6 just one layer is presented, but the architecture also works with 2 or more layers as a deep LSTM. During the model implementation, the input word is reversed to introduce more short term dependencies, as showed in [14] this simple trick improves the performance of the trained model. Thus, the word "casa" would be inserted in the model as "asac". The target sequence keeps the same order. The model can then be trained using Backpropagation Through Time (BPTT) . [11] shows that this model achieve state-of-the-art results when compared to other top performing methods in the G2P task.

Another point that is worthy noting is that for languages like BP, where the effect of co-articulation between words exists, data-driven approaches trained over a lexicon with independent words can not model that effect. Because of the Sequence-to-Sequence Models ability to successfully learn on data with long range temporal dependencies, as showed in [14], possibly using a lexicon composed of sentences with their respective phoneme translations can also model this effect. This can be a future work on this theme.

Syllabification Problem

For a long time, rule-based approaches were assumed to be the best technique for the Syllabification of words. With the development of data-driven approaches that are able to outperform the rule-based algorithms to solve this task for the English language changed this belief. Then, [45] shows that data-driven methods also surpass the performance of rule-based approaches for low complexity languages, such as Italian. These findings opened the door for a deeper analysis of these data-driven solutions in a wider range of languages and for the development of new models to solve it.

The Syllabification problem presents additional challenges when compared to the G2P conversion. The first thing to point out is that the definition of a syllable is not clear. Here, we will define a syllable as the phonological "building block" of words. They are usually constructed over a syllable nucleus (usually a vowel) with additional elements in the nucleus margins (usually consonants). Another important thing we need to mention before analyzing the problem we need to solve is to know that there are two different domains for this problem. The spelling and the pronunciation domains. As our main use case for these NLP algorithms is in the development of TTS systems, we are more interested in the pronunciation/phonetic syllabification of words, that influences directly the rhythm, prosody and stress marking of the BP.

To analyze the Syllabification as a Machine Learning problem we are first going to present the traditional view of this problem. The Syllabification problem consists in determine the location of the syllable boundaries in a word. Between two characters only two decisions are available. To insert a boundary in that position or to not insert a boundary there. So, for every position, between two characters, we need to infer if exists a syllable boundary or not. The collection of the decisions taken for all the positions between characters solve the problem. This can be understood as a structured classification problem [46]. To solve this problem, [46] evaluates and compare the performance of five data-driven different solutions and concludes that Syllabification by Analogy [31] and Hidden Markov Support Vector Machines [47] provides the best performances. [45] compares a rule-based algorithm versus data-driven methods for the Italian language. A Maximum-Entropy solution for the European Portuguese Syllabification is showed in [23].

In this dissertation, we will propose Sequence-to-Sequence Models to solve the Syllabification for BP. The technique is language-independent and can be easily applied for other languages. The model will be presented in Chapter 4 and will approach this task as a sequence mapping problem. The technique proposed will be later evaluated against the rule-based algorithm presented in Section 2.2.1.

Stress Determination Problem

Rule-based approaches are the main technique for Stress Determination. For languages with shallow orthography they achieve a high accuracy level. Despite of being an important information for the pronunciation modeling [48], there are not many papers that addresses the stress determination problem directly using data-driven algorithms.

By the Machine Learning perspective the problem consists in predicting the position of the stress in a word. For different languages the same word can have more than one stress marker, determining a primary, secondary and sometimes a tertiary stress. The problem also changes its complexity level depending on the language type, as *stress-timed* ones tends to be harder than *syllable-timed* ones regarding of accuracy. This problem can be defined as a structured classification problem, the same way as the syllabification one. So, by this interpretation, the same strategies showed to solve the Syllabification problem could be used to solve the Stress Determination problem. In [23] the same Maximum-Entropy strategy is used to solve both Syllabification and Stress Determination. A specific solution for Stress Determination is proposed in [49]. It uses a ranking approach for stress marking approaching it as a sequence mapping problem.

The proposal of this dissertation is to use Sequence-to-Sequence Models to model the Stress Determination problem. As the Syllabification task, the model will try to solve it as a sequence mapping problem. The proposed model is showed in Chapter 4 and will be further evaluated with the rule-based algorithm presented in Section 2.2.1.

2.3 Speech Waveform Generation

In this section we will give a brief explanation about the *back-end* part of the speech synthesizers. The *back-end* is responsible to transform the linguistic specifications, many of them explained in the sections before, in intelligible and natural-sounding speech waveforms. For this step, unlike the Text Analysis part, the block is language-independent. From the beginning of the research in Speech Synthesis until nowadays the approaches to solve this problem changed from knowledge-based and rule-based ones to data-driven ones. This is mainly because of the increase of the speech resources, computer power and Machine Learning algorithms. The principal solutions used nowadays are concatenative speech synthesis [50] and statistical parametric speech synthesis, such as HMM-based speech synthesis [51] and DNN-based speech synthesis [52].

The concatenative speech synthesis, also known as "unit selection" uses a large speech database, and given the linguistic specification, choose the best match speech units to generate synthetic speech. Other factors such as the lexical stress, pitch accent and part-of-speech tags are also used as linguistic context in order to choose the desired speech unit. This technique generate a high-quality natural sounding synthetic speech and is usually used in the development of commercial systems. Because the speech database is previously recorded by a professional speaker in a studio, the speech generated by this technique will have the tendency to have the same style as recorded. Resulting in a synthesizer with the same characteristics as the speaker and a reading-style prosody [50]. If we want to create speech with various styles, emotions and different speaker characteristics is necessary to record large speech databases containing these characteristics a priori, what is time-consuming and cost inefficient [53].

To overcome these disadvantages, Statistical Parametric Speech Synthesis (SPSS) techniques were developed. The main advantage of SPSS is the flexibility to build new voices and new speaking styles [51]. In this approach, during training, acoustic parameters are extracted and used to model speech as a time-series generative model. The most popular SPSS solutions use hidden Markov models [51] or deep neural networks [52] as its generative model. These models represents the acoustic parameters for the linguistic specifications and are used to drive a vocoder in order to generate a speech waveform. HMM-based and DNN-based have been an important topic in research and responsible for most of the papers published in the Speech Processing conferences during the last years. According to the Blizzard Challenge 2008, an annual competition between Speech Synthesis techniques, Statistical Parametric synthesizers are as intelligible as human speech [54]. Despite of this fact, they still need to improve its naturalness, as because of the actual vocoding techniques they sound “buzzy/robotic”. HMMs and DNNs are also used for Automatic Speech Recognition and were responsible for a big improvement in the area in the last few years [55, 56].

Chapter 3

A new set of rules for Brazilian Portuguese G2P Conversion

3.1 Introduction

For the Brazilian Portuguese language, some rule-based G2P converters have been previously reported in [1–4]. For the comparison with other data-driven approaches the algorithm proposed in [6] was chosen. However, after a specific analysis in the algorithm performance it was verified that the accuracy could be enhanced with the addition of new rules. These rules aims both to correct nasalization errors and to improve the performance regarding the transcription for the graphemes <e> and <o>. Other minor errors are also revised. The proposed new set of rules improve the performance of the algorithm from 97.44% to 99.18%.

Therefore, this chapter will introduce a Grapheme-to-Phoneme algorithm with a new set of rules developed specifically for this dissertation. The proposed rules are described in detail and tests were performed to measure the accuracy of the algorithm.

3.2 The rule-based algorithm

The symbols used in the G2P converter are presented in Table 3.1 and the Masculine and Feminine pronouns for BP were previously defined.

3.2.1 The Phoneme Set

The phoneme set used by the converter is presented in Table 3.2 and is the same as the one published in [2], consisting in 38 phonemes. It is represented by SAMPA (Speech Assessment Methods Phonetic Alphabet) [57].

Table 3.1: Symbols used for the rules.

Symbol	Meaning
...	Any Character
V	Vowel
V_ton	Stressed Vowel
V_aton	Vowel not stressed
C	Consonant
C_v	Voiced Consonant
C_uv	Unvoiced Consonant
$< x_i >$	Grapheme x_i
Pont	Punctuation
$[y_i]$	Phone y_i
[*]	No Phone Transcribe
Ltr	Letter
... $< x_i >$ ...	*Default* case for grapheme x_i
<**W_bgn** x_i >	Grapheme x_i comes in the beginning of the word
$< x_1, (x_2, x_3) >$	$< x_1 x_2 >$ or $< x_1 x_3 >$
x_atn	x is not stressed
$< \theta - x_i >$	The set θ excluding character x_i
sp	Blank Space
Prn_M	Masculine pronouns
Prn_F	Feminine pronouns

3.2.2 The set of rules

The proposed rules come to improve the performance of the algorithms presented in [1, 2, 6]. As can be seen in [6], the nasalization errors and the errors of the conversion for the graphemes <e> and <o> were responsible for the largest number of occurrences. To solve the nasalization errors, the precedence of the rules for the grapheme <a> is changed and one additional rule for the grapheme <u> is proposed (Rule 2). For the graphemes <e> and <o>, the errors were due to a lack of rules for words with the open vowel sound, like in the words “ferro” (*iron*) and “forte” (*strong*). To decrease the number of errors for this case, 26 and 15 additional rules were proposed for the graphemes <e> and <o>, respectively, when comparing to the rules of [6]. The proposed rules were mostly conceived using termination patterns for Brazilian Portuguese words. It was observed that the termination patterns can be one strategy to approach the lack of rules for words that contain the sound of the phones [E] and [O]. Furthermore, one rule for the grapheme <r> is also proposed (Rule 4). This rule corrects the transcription of the word “honra” (*honor*), where the grapheme <r> should be transcribed to the phone [R].

Table 3.2: Phoneme set.

Phone	Examples
Oral Vowels	
[a]	jatobá, capacete, cabeça, lua
[E]	é, pajé, pele, ferro, velho
[e]	capacete, resolver, respeito
[i]	justiça, país, lápis, idiota, ele
[O]	ópio, jogos, sozinho, forte
[o]	jogo, golfinho, corpo
[u]	raul, culpa, baú, cururu, logo
Voiced fricatives	
[z]	casa, coisa, quase, exato
[v]	vovó, vamos, avião
[Z]	geladeira, trovejar
Affricates	
[tS]	tia, pacote, constituinte
[dZ]	dia, cidade, disco
Plosives	
[b]	barba, absinto
[d]	dados, administrar
[t]	pato, constituinte
[k]	casca, quero, quanto
[g]	guerra, gato, agüentar, agnóstico
[p]	papai, psicólogo, apto
Nasal Vowels	
[a~]	andar, tampar, canção
[e~]	então, tempo, bem, menos
[i~]	ninho, tinta, latina, importa
[o~]	onda, campeões, somos, homem
[u~]	um, muito, umbigo
Semi-Vowels	
[w]	fácil, voltar, eu, quase
[j]	pai, foi, caracóis, micróbio
[w~]	não, cão
[j~]	muito, bem, parabéns, compõe
Liquids	
[l]	laranja, leitão
[L]	calhar, colheita, melhor
[R]	carro, rua, rato, carga, germe
[X]	casar, certo, arpa, arco
[r]	garoto, frango, por exemplo
Unvoiced fricatives	
[f]	festa, fanfarrão, afta, afluente
[s]	sapo, caçar, crescer, sessão, lápis, excesso
[S]	chá, xaveco, cachorro
Nasal consonants	
[m]	mamãe, emancipar
[n]	nome, atenuar, encanação
[J]	casinha, galinha

The new set of rules can be seen in the Tables 3.3 to 3.12. For each grapheme, the left column contains the algorithm for the rules and the right column contains the correspondent phoneme to be transcribed and word examples for that rule. The precedence of the rule is important during the conversion and must be followed. When a word is inserted in the converter, the rules for each letter are verified until one rule is matched. The last rule for each grapheme usually is a generalization case. In some cases, two letters might be converted into a single phoneme. One example of this case is in the word "sinto" (*feel*) where the graphemes <i> and <n> are converted to just one phoneme according to the Rule 3 (<i>) of the Table 3.6, the nasal vowel [i˜]. Another important case is for the grapheme <x>. The transcription for the <x> follows basic rules, but there are exception cases when it appears in the middle of words where this grapheme is transcribed as two phones: [k s] and [k z]. So, these words need to be treated as exceptions and are shown in Table 3.13.

Table 3.3: Rules for the grapheme <a, b, c, d>.

Rule	**Algorithm for the Grapheme <a>**	**Phone**	**Example**
1	...<an><**Pont**>...	[a˜]	Ivan, Itapoan
2	...<am><**Pont**>...	[a˜w˜]	andam, cresçam
3	...<a (m,n)><**C**-h>...	[a˜]	antena, ampola
4	...<a(**V_ton**)><m,n>...	[a˜]	cama, banho
5	...<â (m,n)><**C**-h>...	[a˜]	lâmpada, cântico
6	...<ão>...	[a˜w˜]	avião
7	...<ã,â>...	[a˜]	amanhã, câmara
8	...<á,à>...	[a]	Antártica, àquela
9	...<a>...	[a]	aracnofobia

Rule	**Algorithm for the Grapheme <b>**	**Phone**	**Example**
1	...<b>...	[b]	abacate

Rule	**Algorithm for the Grapheme <c>**	**Phone**	**Example**
1	...<c><e,i>...	[s]	aceitar, jacinto
2	...<ç>...	[s]	almoço
3	...<c h>...	[S]	acho
4	...<c>...	[k]	claro

Rule	**Algorithm for the Grapheme <d>**	**Phone**	**Example**
1	...<d><i,[i]>...	[dZ]	dia, tarde
2	...<d><**C**-r,l>...	[dZ]	advogado
3	...<d><**Pont**>...	[dZ]	Raid
4	...<d>...	[d]	dote

Table 3.4: Rules for the grapheme <e> - Part 1.

Rule	Algorithm for the Grapheme <e>	Phone	Example
1	...<en><**Pont**>...	[ẽ]	hífen, abdômen
2	...<e(**V_ton**)><l><**C**-h,**Pont**>...	[E]	papel, selva
3	...<e(**V_aton**)><l><**C**-h,**Pont**>...	[e]	sensível, selvagem
4	...<õ,ã><e>...	[j̃]	mães, corações
5	...<a><e>...	[j]	Caetano
6	...<(**Prn_M**) e (**V_ton**)>..	[e]	ele, este
7	...<(**Prn_F**) e (**V_ton**)>..	[e]	ela, esta
8	...<e><rra><**Pont**>...	[E]	terra, guerra
9	...<(**W_bgn**)e><x><**V**>...	[e]	exato, existe
10	...<(**W_bgn**)e,ê><x,s><**C**>...	[e]	exceto, êxtase, estrada
11	...<(**W_bgn**)e><**C**-(m,n,x)>...	[e]	errado, elefante
12	...<e><ne,me><**Pont**>...	[e]	creme, higiene
13	...<e><la,lo><**Pont**>...	[E]	vela, aquarela **Exceção:** pela, pelo, modelo ([e])
14	...<e><ss (a,o)><**Pont**>...	[E]	promessa, acesso
15	...<e><sse><**Pont**>...	[e]	nascesse
16	...<e><sa,so,za,sos><**Pont**>...	[e]	despesa, nobreza, preso, pesos
17	...<e(**V_ton**)><r><**Pont**>...	[e]	beber, ler
18	...<e><st (a,e,o)><**Pont**>...	[E]	festa, modesto, soubeste
19	...<e><ce,sce><**Pont**>...	[E]	conhece, cresce
20	...<e><j(a,o,as,os,am)><**Pont**>...	[e]	igreja, sertanejo, igrejas, sertanejo, desejam
21	...<e><da,do><**Pont**>...	[e]	alameda, medo
22	...<e><sm(a,o,as,os)><**Pont**>...	[e]	mesmo, mesma, mesmos, mesmas
23	...<e><lh(a,o,as,os)><**Pont**>...	[e]	ajoelha, espelho, abelhas, coelhos **Exceção:** velho(s), velha(s) ([E])
24	...<e><vo><**Pont**>...	[e]	escrevo
25	...<e><ça,ço,ços><**Pont**>...	[e]	mereça, conheço, preços
26	...<n><e><gr(a,o,as,os)><**Pont**>...	[e]	negra, alvinegro, negras, alvinegros
27	...<e><s (as,es)><**Pont**>...	[e]	francesas, japoneses
28	...<e><xt (o,a)><**Pont**>...	[e]	sexta, contexto
29	...<e(**V_ton**)><de><**Pont**>...	[E]	procede
30	...<e><to,tos><**Pont**>...	[E]	secreto, objetos
31	...<e><ram><**Pont**>...	[e]	sofreram, morreram
32	...<e(**V_ton**)><ta,tas><**Pont**>...	[E]	atleta, poetas
33	...<e(**V_ton**)><te><**Pont**>...	[E]	basquete, promete
34	...<ê (m,n)><**C**-h>...	[ẽ]	ciência
35	...<e (m,n)><**C**-h>...	[ẽ]	embora, entoação

Table 3.5: Rules for the grapheme <e> - Part 2.

Rule	Algorithm for the Grapheme <e>	Phone	Example
36	...<e(**V_ton**)><m,n>...	[e˜]	tema, cena
37	...<e(**V_ton**)><r><**C**>...	[E]	aberto, conversa
38	...<(e,é,ê)m><**Pont**>...	[e˜j˜]	contém, têm, ontem
39	...<ê>...	[e]	bebê
40	...<é>...	[E]	época
41	...<e><s><**Pont**>...	[i]	frases
42	...<e(**V_ton**)i>...	[e j]	areia, fiquei
43	...<e(**V_ton**)><u><**Pont**,**Ltr**>...	[e]	museu
44	...<e(**V_ton**)><z><**Pont**>...	[e]	xadrez
45	...<u><e(**V_ton**)><ta><**Pont**>...	[e]	plaqueta
46	...<t><e(**V_ton**)><ve><**Pont**>...	[e]	manteve
47	...<e><**Ltr**><**V_ton**>...	[e]	semestre
48	...<**Pont**><e><**Pont**>...	[e]	o João e a Ana
49	...<e><**Pont**>...	[i]	índice
50	...<e(**V_ton**)>...	[E]	nega
51	...<e(**V_aton**)>...	[e]	secreto

Table 3.6: Rules for the graphemes <f, g, h, i>.

Rule	Algorithm for the Grapheme <f>	Phone	Example
1	...<f>...	[f]	faca

Rule	Algorithm for the Grapheme <g>	Phone	Example
1	...<g><e,i>...	[Z]	gelo
2	...<g u><e,i>...	[g]	guindaste
3	...<g>...	[g]	garoto

Rule	Algorithm for the Grapheme <h>	Phone	Example
1	...<h>...	[*]	hoje

Rule	Algorithm for the Grapheme <i>	Phone	Example
1	...<u **V_ton**><i><t>...	[j˜]	muito
2	...<i e><**Pont**>...	[i]	superfície, planície, cárie
3	...<i,í (m,n)><**C**-h,**Pont**>...	[i˜]	sinto, símbolo, fim
4	...<i><m,n><**V**,h>...	[i˜]	inoperante, imaginar, ninho
5	...<V-i><i>...	[j]	coisa, sai, queixa
6	...<i,í>...	[i]	líquido, saí

Table 3.7: Rules for the graphemes <j, k, l, m, n>.

Rule	Algorithm for the Grapheme <j>	Phone	Example
1	...<j>...	[Z]	jambo

Rule	Algorithm for the Grapheme <k>	Phone	Example
1	...<k>...	[k]	kelvin

Rule	Algorithm for the Grapheme <l>	Phone	Example
1	...<l><**V**>...	[l]	ala
2	...<l h>...	[L]	alho
3	...<l>...	[w]	vogal

Rule	Algorithm for the Grapheme <m>	Phone	Example
1	...<e,é,ê,i><m><**sp**><**V**>...	[J]	Alguém usou, Quem está
2	...<m>...	[m]	mameluco

Rule	Algorithm for the Grapheme <n>	Phone	Example
1	...<n h>...	[J]	ganho
2	...<n>...	[n]	nata

Table 3.8: Rules for the grapheme <o> - Part 1.

Rule	Algorithm for the Grapheme <o>	Phone	Example
1	...<o(**V_ton**)><l><**C**-h,**Pont**>...	[O]	sol, girassol, futebol **Exceção:** gol ([o])
2	...<o(**V_aton**)><l><**C**-h,**Pont**>...	[o]	soldadura, soltar
3	...<ou>...	[ow]	ouvir, couve, estou
4	...<o(**V_ton**)><a><**Pont**,**Ltr**>...	[o]	pessoa, canoa
5	...<o><so><**Pont**>...	[o]	saudoso, virtuoso
6	...<o><s (a,as,os)><**Pont**>...	[O]	saudosa, virtuosas, caprichosos
7	...<o,ô (m,n)><**C**-h,**Pont**>...	[o~]	compositor, gôndola
8	...<o(**V_ton**)><m,n>...	[o~]	soma, risonho
9	...<o><r><**Pont**>...	[o]	compor, amor, dor **Exceção:** maior(es), menor(es), pior(es), suor(es) ([O])
10	...<o><z><**Pont**>...	[O]	voz, algoz **Exceção:** arroz ([o])
11	...<o><rre><**Pont**>...	[O]	corre, morre
12	...<o><res><**Pont**>...	[o]	cores, leitores
13	...<o><b (re,ra)><**Pont**>...	[O]	nobre, redobra
14	...<o><s (sa,se,so)><**Pont**>...	[O]	nossa, posse, vosso
15	...<o(**V_ton**)><la><**Pont**>...	[O]	bola, cartola
16	...<o(**V_ton**)><te><**Pont**>...	[O]	mascote, filhote
17	...<o><rno><**Pont**>...	[o]	forno, transtorno
18	...<o><rro><**Pont**>...	[o]	recorro, cachorro
19	...<o><ço><**Pont**>...	[o]	almoço, pescoço
20	...<d,r,ss><o><r(a,as)><**Pont**>...	[o]	professora, metralhadora, pintora
21	...<o(**V_ton**)><r><**C**-h>...	[O]	sorte, reforma

Table 3.9: Rules for the grapheme <o> - Part 2.

Rule	**Algorithm for the Grapheme <o>**	**Phone**	**Example**
22	...<ô>...	[o]	vovô
23	...<ó>...	[O]	vovó
24	...<õ>...	[o~]	corações
25	...<o(**V_ton**)><i>...	[o]	oito
26	...<(**W_bgn**)c><o><**Pont**>...	[o]	co-produção
27	...<a><o><**C**,**Pont**>...	[w]	ao, caos
28	...<o><**Ltr**><**V_ton**>...	[o]	opor
29	...<o><**Pont**>...	[u]	músico
30	...<o><s><**Pont**>...	[u]	carros
31	...<o><lh (a,o)><**Pont**>...	[o]	repolho, folha
32	...<o><sto><**Pont**>...	[o]	imposto, rosto
33	...<o(**V_ton**)>...	[O]	mole, logo
34	...<o(**V_aton**)>...	[o]	começa

Table 3.10: Rules for the graphemes <p, q, r>.

Rule	**Algorithm for the Grapheme <p>**	**Phone**	**Example**
1	...<ph>...	[f]	Philipe
2	...<p>...	[p]	pato
Rule	**Algorithm for the Grapheme <q>**	**Phone**	**Example**
1	...<qu><i,e,o>...	[k]	quito, quente, quota
2	...<q><ü,ua>...	[k]	cinqüenta, quase
Rule	**Algorithm for the Grapheme <r>**	**Phone**	**Example**
1	...<rr>...	[R]	carro
2	...<r **sp** r>...	[R]	Um pomar rodeado de flores.
3	...<(**W_bgn**)r>...	[R]	A rua foi interditada.
4	...<n><r><**V**>...	[R]	honra, Henrique
5	...<r><**V**>...	[r]	ratoeira
6	...<r><**sp**><**V**,h>...	[r]	Falta acertar apenas uma.
7	...<r><**sp**><**C_uv**>...	[X]	Pecar pelo meio do caminho.
8	...<r><**sp**><**C_v**>...	[R]	Injetar grãos de arroz.
9	...<r><**C_uv**>...	[X]	perco
10	...<r><**C_v**>...	[R]	carga
11	...<r>...	[X]	Ela irá se lascar.

Table 3.11: Rules for the grapheme <s, t, u, v, w, x>.

Rule	Algorithm for the Grapheme <s>	Phone	Example
1	...<tr(a,â)n><s><**V**>...	[z]	transação, trânsito
2	...<ob><s><équio>...	[z]	obséquio
3	...<ã><s><**Pont**>...	[j˜s]	fãs, maçãs
4	...<**V_ton**><s><**Pont**>...	[j s]	mas, gás, atrás
5	...<sh>...	[S]	shiatsu
6	...<(**W_bgn**)s>...	[s]	O sapato está lustrado.
7	...<**V**><s><**V**>...	[z]	asa
8	...<s><**C_v**>...	[z]	transgredir
9	...<ss>...	[s]	assar
10	...<sc><e,i>...	[s]	crescer
11	...<sç>...	[s]	cresçam
12	...<s **sp** j>...	[Z]	Eles jogaram bola.
13	...<s><**sp**><**V**,**C_v**,h>...	[z]	Os aros são cromados.
14	...<s>...	[s]	Eles receberam o prêmio.

Rule	Algorithm for the Grapheme <t>	Phone	Example
1	...<th><**Pont**>...	[tS]	Ruth
2	...<th>...	[t]	Arthur
3	...<t><**C**-r,l>...	[tS]	algoritmo
4	...<t><i,[i]>...	[tS]	tia, mete
5	...<t><**Pont**>...	[tS]	Aquele set foi difícil.
6	...<t>...	[t]	tato

Rule	Algorithm for the Grapheme <u>	Phone	Example
1	...<ü>...	[w]	lingüística
2	...<u,ú (m,n)><**C**-h,**Pont**>...	[u˜]	abundante, atum
3	...<u><m,n>...	[u˜]	uma, unha
4	...<u,ú>...	[u]	acústica

Rule	Algorithm for the Grapheme <v>	Phone	Example
1	...<v>...	[v]	voando

Rule	Algorithm for the Grapheme <w>	Phone	Example
1	...<w>...	[w]	watt

Rule	Algorithm for the Grapheme <x>	Phone	Example
1	...<(**W_bgn**)e><x><**V**,**C_v**>...	[z]	execrar
2	...<(**W_bgn**)e,ê><x><**C_uv**>...	[s]	excêntrico
3	...<(**W_bgn**)e,ê><x><**Pont**><**V**,**C_v**>...	[z]	ex-amigo
4	...<(**W_bgn**)e,ê><x><**Pont**><**C_uv**>...	[s]	ex-presidente
5	...<x><**Pont**>...	[k s]	Marx
6	...<**V**-e><x><**V**>...	[exception]	Check Table 3.13
7	...<x>...	[S]	faxina

Table 3.12: Rules for the graphemes <y, z>.

Rule	**Algorithm for the Grapheme <y>**	**Phone**	**Example**
1	...<y><**C**>...	[i]	Yguaçu
2	...<y>...	[j]	Yanomami
Rule	**Algorithm for the Grapheme <z>**	**Phone**	**Example**
1	...<z><**sp**><**C_uv**>...	[s]	Ferraz furou o ferro.
2	...<z><**sp**><**C_v,V,h**>...	[z]	Faz anos que não o vemos.
4	...<**V_ton**><z><**Pont**>...	[j s]	faz
5	...<z><**Pont**>...	[s]	O que José fez?
6	...<z>...	[z]	zumbido

Table 3.13: Exceptions for the grapheme <x>.

Grapheme-to-Phone mapping		% Partial List of Words
<x>	[k s]	oxítono, oxítona, oxítonos, oxítonas, oxidas oxidação, complexo, anexar, oxigênio, oxiúro oxalado, úxer, uxoricida, axila, axiologia, íxia táxi, sintaxe, axioma, complexo, conexo, convexão convexo, crucifixo, fixo, fixar, fixação fluxo, galáxia, indexação, léxico maximizar, marxismo, nexo, paradoxo, paroxítono perplexo, prefixo, saxofone, prolixo
<x>	[k z]	ixofagia, ixomielite, ixolite, ixômetro, ixora, ixoscopia, ox-acético

3.2.3 Experimental Results

The proposed rules were implemented and tested using a list of words from the Ispell dictionary [13] containing a total of 33737 words divided in 11 test sets to perform K-Fold Cross Validation [15]. Each set contained 3067 words. The phonemes generated by the algorithm were automatically checked using Levenshtein Distance [58] and 99.18% ± 0.07% were correctly converted (Phoneme Error Rate).

The highest number of errors comes from the conversion of the graphemes <e> and <o>, which are complex in terms of their conversion variability, and, moreover, their occurrences are high in the BP language. Foreign words, acronyms and proper names are another source of errors during the transcription. This problem should be considered, but because of the difficult approach to solve them in a rule-based way, the frequent words incorporated to the spoken Brazilian Portuguese could be inserted in a dictionary with their respective phoneme transcriptions. The foreign words present additional problems as they acquire many different pronunciations after incorporated to another language.

For the development of text-to-speech systems another important analysis is the percentage of words that present conversion error (Word Error Rate). The words with a G2P error will be pronounced wrongly by the synthesizer and will affect directly its intelligibility and its naturalness. Moreover, these errors can also affect the training of the acoustic models. The experimental results show that $3.26\% \pm 0.14\%$ of the words are converted with some associated error. This proves the importance of the G2P step during the development of a TTS system and a need for high accuracy levels for the G2P algorithms.

Comparisons with the results of the algorithm presented in [6] show that the additional rules proposed in this chapter solved the high number of nasalization errors and improved the general performance of the G2P converter from 97.44% to 99.18%. Informal subjective tests realized with an HMM-based text-to-speech system that was trained and tested using the set of proposed rules also showed an improvement on the naturalness of the generated speech when comparing to the G2P of [6].

Chapter 4

Proposal of Sequence-to-Sequence Models for G2P with a richer feature set, Syllabification and Stress Determination

4.1 Introduction

Sequence-to-Sequence Models, proposed in [14], presents a generalized solution to solve sequence to sequence problems. The basic idea consists in encode an input sequence into a fixed-dimensional vector using a RNN chain. Then, use a second RNN chain to decode the output sequence conditioned to the vector generated. The second RNN chain works essentially as a language model conditioned on the input sequence [14]. Recently, this framework have been applied to several tasks with promising results. Some of these applications include Machine Translation [14], Image Captioning [59] and Grapheme-to-Phoneme Conversion [11].

In Chapter 2 we already explored the applicability of the Sequence-to-Sequence Models to the BP G2P conversion. Now, in this chapter we will propose a model to the G2P Conversion using a richer feature set. In addition to, we will model the Syllabification and Stress Determination tasks as sequence mapping problems. These tasks also share some of the characteristics of the G2P problem, where the lexicon is relatively small and the use of plain n-gram models are supposed to perform well. So, Sequence-to-Sequence Models for these tasks are proposed. The proposed models will be evaluated and the results showed in Chapter 5.

4.2 G2P Model using a richer feature set

In Chapter 2 it was already described a Sequence-to-Sequence Model to solve the G2P conversion proposed in [11]. In the model, the word is the input and its respective phonemic transcription is the output. The model is able to infer the phoneme sequence using just the grapheme information, working in the same way as a basic language model. However, if we analyze a rule-based G2P algorithm, besides of the grapheme information, additional information is used to design the rules. For example, in the rule-based algorithm presented in Chapter 3, the stress information is required for many rules, especially regarding the transcription of the BP vowels. This information directly influences the algorithm performance and shows that some of the language patterns cannot be deduced just by the grapheme sequence information. The information from the other Text Analysis parts might embody valuable information for deducing the word pronunciation. So, it raises a question: can the use of a richer feature set in the model improve the G2P conversion? Other researchers tried to answer this question before. In [12] the syllable information was inserted to the model of a G2P converter for the Portuguese spoken in Portugal. It showed that the results improved in general and the difference was significant statistically. [60] also tried to insert the syllable information to a G2P system for English. As he used the output of a Syllabification by Analogy system that included errors, he could not achieve a big difference in performance with the addition of the syllable information. Although, he shows that including good quality syllable information can enhance the performance of the pronunciation systems used in text-to-speech. Then, [61] presents a G2P system using morphology information for German. This additional information improved significantly the performance of the converter. Thereupon, we propose the insertion of the stress information in the encoding block of the Sequence-to-Sequence Models for the G2P task. The proposed model will be further compared with the model that does not contain the stress information and the results are showed in Chapter 5.

The power of the Sequence-to-Sequence Models are clear when we think in how to add the stress information to it. The important information is the location of the stress marker between the graphemes of the input sequence. It is not necessary for the network to understand the meaning of that symbol before training. Thus, we should choose a character that will represent the stress information, here we chose the character “ ’”, and insert it in their input word position. This can be seen in Figure 4.1. So, now the input word has one extra character that represents the stress marker and will be included in the encoding RNN chain. Figure 4.2 represents the model used for training. It is worthy noting that this extra character is encoded exactly like the other ones not affecting the training procedure.

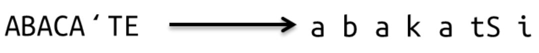

Figure 4.1: G2P with Stress information Example.

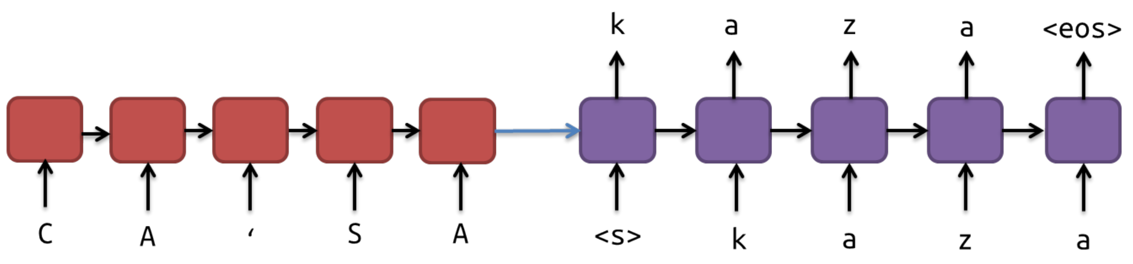

Figure 4.2: Sequence-to-Sequence Model for G2P with Stress information.

4.3 Syllabification Model

In Chapter 2 we presented different data-driven techniques to solve the Syllabification problem. The idea was to identify the existence of syllable boundaries between graphemes. Only two options were available, the existence or the non-existence of the boundary between two graphemes. Here, however, the interpretation will be different. Our objective is to design the Syllabification task using Sequence-to-Sequence Models. So, model this task as a mapping sequence problem. To do this we will keep using the word as the input, inserted one character at a time. Remind that the word is inserted in reverse other as [14]. The modifications are related to the mapping on the output side of the model. The output chosen is the word with the syllable boundaries in the right position. The syllable boundaries are represented by an extra character. In our case, we chose the character "-", but any character outside the alphabet symbols for the target language can be used. An example for the new line of the lexicon is showed in Figure 4.3. In Figure 4.3, we show that the graphemes in the right part are exactly the same as the ones in the left part for the input word. Nevertheless, this does not mean that they will be encoded the same way by the model. This relation between the input and output graphemes forming the same word should be learned by the network during training. The problem, in the end, can be seen as a generative model that inserts an extra character representing the boundaries in their expected position. If we think about that in Figure 4.4 when we input the grapheme "A" at the decoding part of the model, it can be interpreted as the following question: Should I insert the boundary character "-" as the next symbol having the information that the input word is "casa" and I already generated "CA"? In this case, the network should have a higher probability for the character "-" being generated than for any other possible character. The Figure 4.4

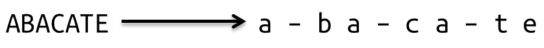

Figure 4.3: Syllabification Example.

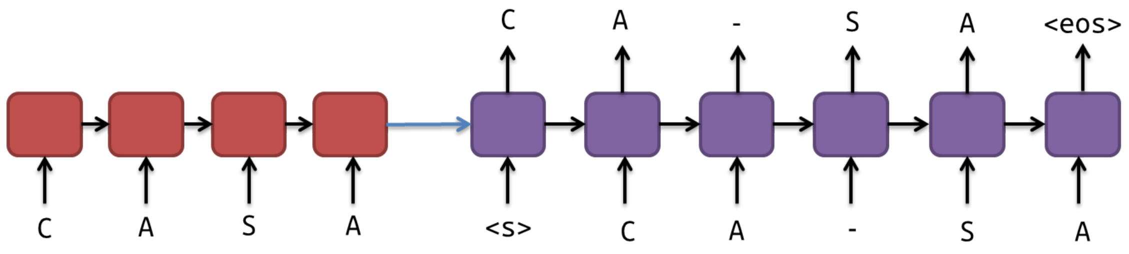

Figure 4.4: Sequence-to-Sequence Model for Syllabification.

represents the model used for training.

4.4 Stress Determination Model

The same idea of turning the structured classification problem into a mapping sequence problem used for Syllabification is also used for Stress Determination. The idea now is to identify the existence of stress markers between graphemes. To do this we will keep using the unmodified word as input. The input word is time-reversed [14]. One more time, the modifications are made in the decoding side of the model. The output chosen is the word with the stress markers in their right position. The stress marker is represented by the character “ ’”. The new lexicon line is presented in Figure 4.5. As in the Syllabification task presented in the last subsection, the problem can be seen as predicting the next character generated, where the prediction of the character “ ’” indicates the presence of stress. The stress corresponds to the grapheme generated previously by the decoding LSTM chain. For example, if the stress marker is generated after the vowel “A”, it means that this vowel is stressed. For example, in Figure 4.6 the question that arises for the network after generate the character “A” in the decoding RNN chain is: Should I insert the stress marker “ ’” as the next symbol knowing that the input word is “casa” and I already generated “CA”? This probability should be maximized in comparison with the probability of generating the other possible characters. The Figure 4.6 represents the model used for training.

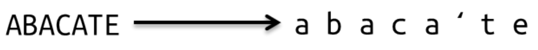

Figure 4.5: Stress Determination Example.

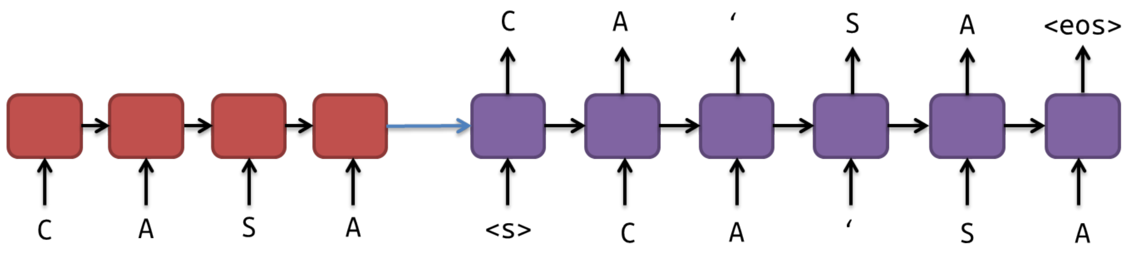

Figure 4.6: Sequence-to-Sequence Model for Stress Marking.

Chapter 5

Experiments and Results

5.1 Datasets and Performance Metrics

The lack of open source standardized datasets for BP is a problem. So, in order to perform the experiments of this chapter three datasets for Brazilian Portuguese were created. One for G2P conversion, one for Syllabification and one for Stress Determination.

First of all, to develop the datasets, a list of all the words contained in the Ispell Dictionary [13] was created. In this list, some foreign language material and words that contained simple orthographic errors were discarded. The final version of the list then contained 33737 words. This is also the length of the datasets created. A histogram that shows the length of the lexicon words is presented in Figure 5.1. These words were then put into the rule-based algorithms described in Chapter 2 to create a first version of the datasets. These versions that contained errors inserted by the rule-based approaches were then manually corrected to generate the final versions used for training.

The G2P dataset contains orthographic words with their phonetic transcription. The SAMPA phonetic alphabet for BP [57] was used in the transcription. The homographs were excluded, it means that just one possible pronunciation of these words are presented in the dataset. Co-articulation effects were also excluded. An alternative version of this dataset with the stress information included in the orthographic form of the word was also created. To the evaluation of the G2P Conversion Word Error Rate (WER) and Phoneme Error Rate (PER) calculated using Levenshtein Distance [58] were used. The WER calculates the percentage of words transcribed correctly. The Levenshtein distance calculates the number of insertions, deletions and substitutions necessary to transform the predicted transcription in the correct transcription and then this number is divided by the total number of phonemes in the set to acquire the PER. For the Syllabification dataset the orthographic word

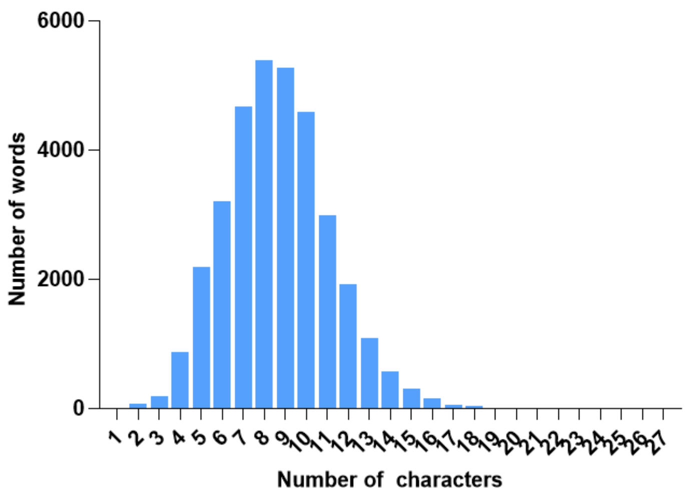

Figure 5.1: Histogram to show the number of words for each word length in the datasets created.

with their transcription separated by a syllable boundary character were used. This is the same model used in Chapter 4. The evaluation was made regarding the WER. In the Stress Determination dataset the orthographic word was followed by the transcription including a character representing the primary stress location, like the model proposed in Chapter 4. WER was also used for evaluation.

As the Syllabification and Stress Determination datasets were created to the training of the Sequence-to-Sequence Models proposed in Chapter 4, the use of these datasets for other data-driven approaches might need adaptation scripts.

The evaluation is performed using k-fold cross validation [15]. The datasets were randomly organized and splitted in k folds. The k models were trained independently on *k-1/k-th* of the dataset and evaluated on the remaining *k-th*. Error rates, in this case, are computed over the union of the k test sets. The standard deviation and standard error [62] are calculated as follows:

$$SD = \sqrt{\frac{1}{k-1}\sum_{i=1}^{k}(x_i - \bar{x})^2} \tag{5.1}$$

$$SE = \frac{SD}{\sqrt{k}} \tag{5.2}$$

where $\bar{x}$ is the mean value of the observations. Considering a normal distribution for the mean error, the three sigma rule [63] shows that 95% of the values lies within 1.96 standard errors away from the mean. So, for 95% confidence intervals, our results will be:

$$\bar{x} \pm (1.96 \times SE) \tag{5.3}$$

5.2 Experiments

The experiments described in the next sections were simulated in an environment consisting of a *Linux* platform using the distribution *Ubuntu 14.04*, a *Intel Core i3 2.4GHz* x *4* processor with *3Gb* of RAM memory.

5.2.1 Comparison between Rule-based and Data-Driven approaches for G2P

The first experiment consisted in the comparison between rule-based and data-driven solutions for the Grapheme-to-Phoneme Conversion. The rule-based algorithm presented in Chapter 3 was compared with three data-driven approaches explained in Chapter 2. To perform the evaluation 11-fold cross validation was used. The order of the words in the G2P dataset were randomly permuted and used to create the 11 sets for simulations.

Because it is a knowledge-based solution, the rule-based algorithm did not require a training step. So, just the evaluation of the algorithm performance for the test sets was needed. The exceptions lists were turned off during evaluation. For the WFST-based approach, the implementation of [64] was used. The training model consisted of a 7-gram trained using the *mitlm* toolkit [65]. The decoding of the test set used the default decoder that calculates the shortest path through the phoneme lattice created by the model. In the Joint-Sequence Model approach, [66] was used to train the model and decode the test set. The model was trained using a 7-gram language model. For the decoding step, first-best search was used [9]. The Sequence-to-Sequence Models implementation used is the one presented in [67]. The architecture chosen was a deep network with 2 layers of 64 GRU units. Ten percent of the training set was used as the validation set. The learning rate started in 0.5 with a decay factor of 0.99. The gradients were clipped to a maximum value of 5. Batches of 64 words were used for training. Then, the test set decoding was performed using a greedy search decoder.

Before analyzing the results it is interesting to also compare the data-driven approaches used in the comparison regarding their latency. Because the comparison is performed using 11-fold cross validation the training step need to be repeated many times so it turns that this in an important factor. For example, for the comparison presented in this section, for each data-driven approach is necessary the training of 11 different models and the decoding of 11 test sets. We are going to present the training time for each data-driven approach in orders of magnitude that are divided in the following categories: minutes, hours and days. This division is made mainly because the training time depends on the size of each dataset, number of parameters, and for the neural network approaches the training design regarding convergence and number of epochs. So, for the data-driven approaches chosen in this dissertation the fastest algorithm consists in the WFST-based strategy. The training time for this algorithm is in the order of 'minutes' and [10] shows that this algorithm is designed to achieve faster training times when compared with other data-driven approaches. For our dataset, the training time was completed in approximately thirty minutes. The Joint-Sequence Models takes the order of 'hours' to complete the training step. In our case, the training of our dataset took approximately four hours. The slowest approach regarding of training time was the Sequence-to-Sequence Models. Because it uses a deep neural network architecture, this technique takes an order of magnitude of 'days' to achieve the convergence of the model performing the training using BPTT. Using the G2P dataset created, the training time took approximately two days. Regarding the decoding time, the WFST-based approach achieves the best latency, and the Sequence-to-Sequence Model strategy is the slowest one, especially because it needs to load the network parameters before start decoding. An important observation to the training times presented before is that the simulation environment consisted in a CPU with limited processing power. With the utilization of powerful GPUs or Super Computers this training time can be reduced by a relevant factor. Primary tests using a *GeForce GTX 1070* GPU reduced the training time by more than ninety percent. Therefore, the available hardware needs to be considered during the development of projects that uses data-driven approaches.

Table 5.1: Comparison between rule-based and data-driven approaches for G2P

Method	PER (%)	WER(%)
Rule-Based	0.82 ± 0.07	3.26 ± 0.14
WFST-based [64]	0.73 ± 0.05	4.05 ± 0.15
Joint Sequence Models [66]	0.67 ± 0.06	4.02 ± 0.20
Sequence-to-Sequence Models [67]	0.52 ± 0.05	2.28 ± 0.13

Thus, we report results for the rule-based and data-driven approaches for G2P conversion in Table 5.1. The first conclusion that can be taken from the table is the ability of the data-driven approaches to yield similar or outperform the G2P rule-based algorithm. The three data-driven algorithms chosen were able to present better Phoneme Error Rate than the rule-based algorithm, with the Sequence-to-Sequence Models achieving the best rate overall with PER equal to $0.52 \pm 0.05\%$. These results can also be seen in Figure 5.2. For the Word Error Rate, the rule-based algorithm showed better results than two data-driven algorithms, but was outperformed by Sequence-to-Sequence Models that presented WER equal to $2.28 \pm 0.13\%$. The Figure 5.3 shows these results graphically. So, this shows that the data-driven algorithms can be used successfully to the G2P task for a shallow ortography lan-

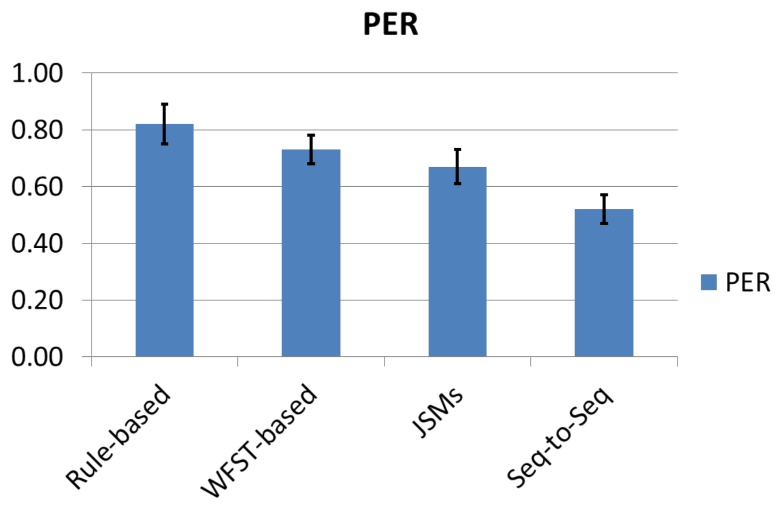

Figure 5.2: PER Comparison.

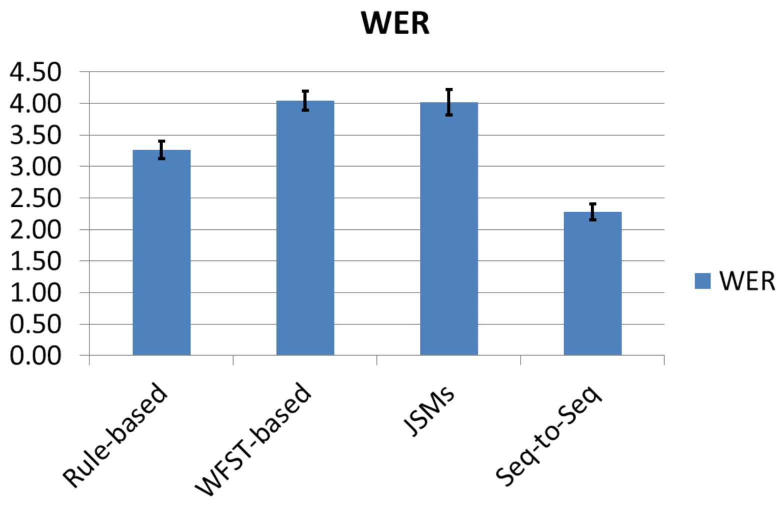

Figure 5.3: WER Comparison.

guage, such as BP, and the technique based in a Deep Neural Network architecture using Gated Recurrent Units showed the best performance both at PER and WER.

In Table 5.1 we showed that a Sequence-to-Sequence Model architecture with 2 layers of 64 units achieve the best performance for the BP conversion when compared with a rule-based algorithm and other two data-driven algorithms. Now, we are going to analyze how this architecture compare to the similar architecture presented in [11] for the G2P task used to evaluate the performance for English. For a deep orthography language, such as English, the G2P is more complex than for a shallow orthography language, such as BP. To prove this fact we will compare the architectures and WER of both languages using a encoder-decoder Sequence-to-Sequence model. For BP, using 2 layers of 64 units was enough to achieve state-of-the-art performance with a WER equal to $2.28 \pm 0.13\%$ and outperform a rule-based algorithm

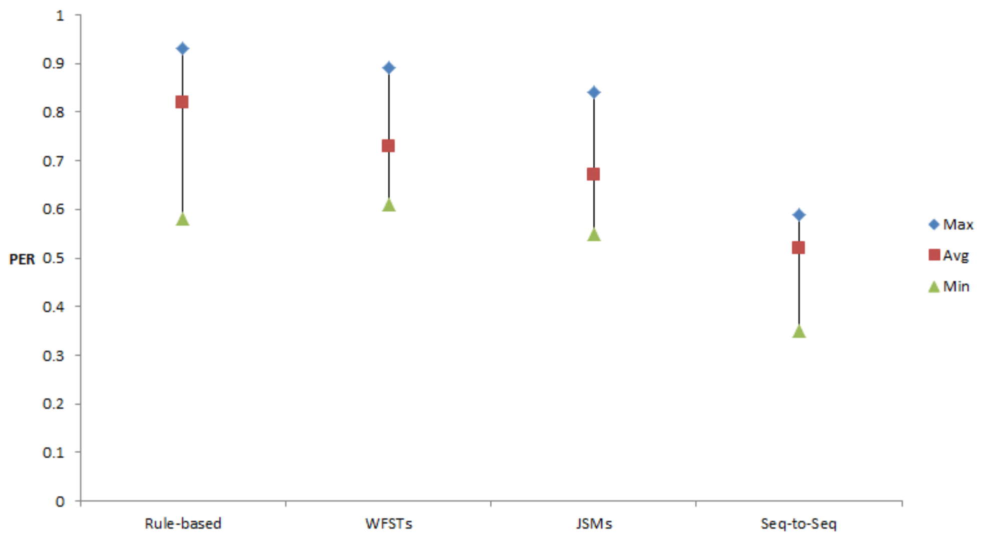

Figure 5.4: PER -Maximum, Minimum and Average values per each approach.

and other two data-driven approaches. However, for English, the same architecture achieves a WER of 32.03% trained in the CMUDict dataset [68], where the state-of-the-art baseline is 24.40%, as showed in [67]. To achieve a similar performance with the baseline, the network architecture need to change the number of units from 64 to 512 units, achieving, then, a WER equal to 24.23% [67]. The result of this change is a significant increase on the number of parameters on the network architecture.

We started this chapter talking about the lack of standardized datasets for the tasks presented in this dissertation for BP. However, why a standard training and test set is important when comparing these algorithms? What is the importance of performing cross validation for the comparison of the values obtained? To answer this question we need to analyze how the performance metrics can vary depending on the training and test set used for development. Because we used a 11-fold cross validation in our results for this comparison we have, for each technique, eleven different values for WER and PER. So, we could list the minimum, maximum and average values to analyze the variance of the performance metrics. Figure 5.4 shows these values for the Phoneme Error Rate. It can be seen that the worst value for the Sequence-to-Sequence Models, the technique with the best performance in our comparison, is really close to the best accuracies obtained for the other three approaches. For example, if the best value of the rule-based approach coincides with the worst value of the Sequence-to-Sequence Model for a specific dataset, it could be misunderstood that both techniques presents similar accuracy. In Figure 5.5 the variation for the WER is showed. The variance of the values can again be verified. So, to perform the comparison of two techniques is important the employment of the same standardized set, and, in addition to, the statistical validation of the results.

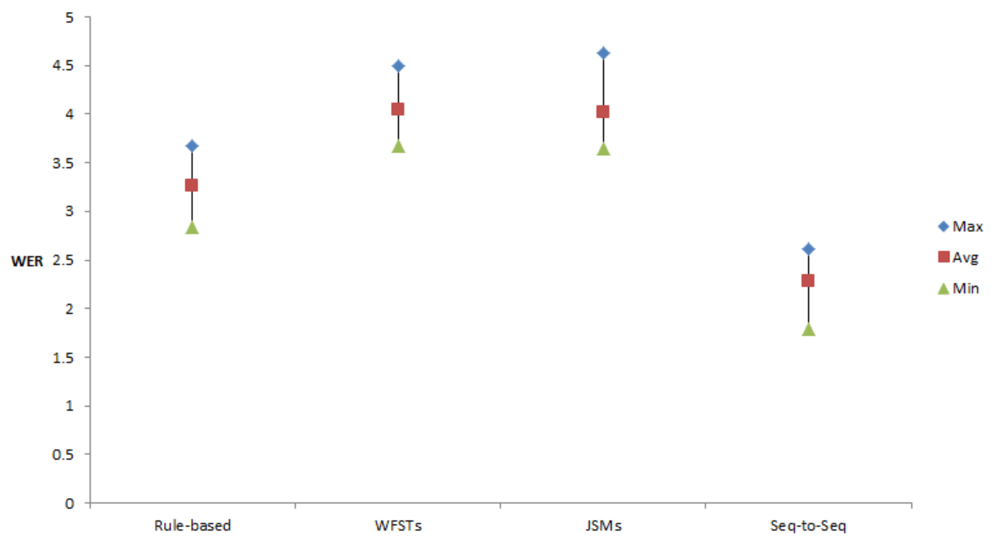

Figure 5.5: WER -Maximum, Minimum and Average values per each approach.

5.2.2 Sequence-to-Sequence Models for G2P with and without the stress information

The use of a richer feature set have already been proven useful for the G2P task [12]. So, in this experiment we insert the stress information in a Sequence-to-Sequence Model, as proposed in Chapter 4, and compare this model with the one that presented the best performance in the last section. Considering that the stress information is used in the design of the rule-based algorithm from Chapter 3 is expected that the insertion of this information decreases the error rate on the performance metrics.

This experiment also uses a 11-fold cross validation in the created dataset and the same variables from the last section are used for the training of the proposed Sequence-to-Sequence Model. With respect to training time of the network, the insertion of stress information turns the convergence more difficult. Thus, the training time increased by a factor of 1.5 to 2 times in comparison with the simulation of a model without stress information.

The Table 5.2 shows that the expected improvement in performance after the addition of stress information is confirmed. The value for the average PER is improved by 0.08% and the average WER is improved by 0.37%. Therefore, the best performance for G2P conversion is achieved with the proposed model that contains the stress information. This model achieve a PER of $0.44 \pm 0.05\%$ and a WER of $1.91 \pm 0.14\%$.

These results shows that a model that includes information from the Text Analysis blocks that are relevant for the G2P conversion can increase the performance of these systems and consequently the accuracy of TTS systems. By virtue of the flexibility and encoder-decoder architecture of Sequence-to-Sequence Models the modeling of different NLP algorithms in just one chain can be seen as a future work that can increase even more the performance of these tasks. Another possibility is the modeling of multiple chains, that contains the information from multiple blocks, in parallel and the use of this multi-parallel-chain to generate the output sequence.

Table 5.2: Comparison between rule-based and data-driven approaches for G2P

Method	PER (%)	WER(%)
Sequence-to-Sequence Models without stress information	0.52 ± 0.05	2.28 ± 0.13
Sequence-to-Sequence Models with stress information	0.44 ± 0.05	1.91 ± 0.14

5.2.3 Syllabification using Sequence-to-Sequence Models

In this experiment, a data-driven model for Syllabification proposed in Chapter 4 is compared with a rule-based algorithm presented in [6]. For this experiment, a 5-fold cross validation was used. To turn the dataset size divisible for 5, two one-letter words from the dataset were excluded. Thus, the dataset used for training had 33735 words. The environment was similar to the one used to do the training and testing of the other experiments that used Sequence-to-Sequence Models.

The results obtained after the simulations can be seen in Table 5.3. First of all, we should explain that for this experiment just WER is used as a performance metric because the PER using Levenshtein distance has no meaning for the proposed model. For this model, a new performance metric regarding the right positioning of the syllable boundaries should be proposed and implemented. So, analyzing the results we can see that the proposed model clearly outperforms the rule-based algorithm. The proposed Sequence-to-Sequence Model achieve a WER of $1.70 \pm 0.32\%$ against $4.15 \pm 0.18\%$ from the rule-based algorithm.

Table 5.3: Comparison between the proposed data-driven approach and a rule-based approach for Syllabification

Method	WER(%)
Sequence-to-Sequence Models	1.70 ± 0.32
Rule-based	4.15 ± 0.18

5.2.4 Stress Determination using Sequence-to-Sequence Models

This experiment also compare the proposed data-driven model for Stress Determination proposed in the last chapter with the respective rule-based algorithm from [6]. 5-fold cross validation was used to analyze the results with two one-letter words being excluded from the created dataset to turn its size divisible for 5. The training set, then, contained 33735 words. The environment was the same as the other Sequence-to-Sequence Models experiments.

It is important to mention that the interpretation will be different from the last experiments. This is mainly because of the difficulty in the development of the Stress Determination dataset and the good performance of the rule-based algorithm. The dataset was created by the rule-based algorithm presented in [6] and then manually corrected. In [6] is showed that the rule-based performance achieve a WER of approximately 0.1% when discarded proper names, foreign words, acronyms and initialisms. The main reason for this good performance is the presence of accent markers in BP, as they are highly correlated with stress patterns. As can be seen in Table 5.4, the number of mistakes corrected is really low, so, the author considered the dataset being representative of the rule-based algorithm. Therefore, the results in this experiment will be interpreted as the competence of the proposed model to fit the patterns for the Stress Determination task described in the rule-based algorithm.

The results in Table 5.4 shows that only 0.03% of the words contained in the dataset were corrected in comparison with the rule-based algorithm performance. The proposed model, then, achieved a WER of $0.45 \pm 0.13\%$. This result indicate that the model has the ability to detect the stress patterns of the BP language. Nevertheless, is necessary a collaborative work between BP linguistic experts on the creation of a standardized dataset for this task to perform reliably comparisons between different approaches.

Table 5.4: Comparison between the proposed data-driven approach and a rule-based approach for Stress Determination

Method	WER(%)
Sequence-to-Sequence Models	0.45 ± 0.13
Rule-based	0.03 ± 0.02

5.2.5 Sequence-to-Sequence Models trained with datasets of different sizes

At last, this experiment will analyze the influence of the training set size in the performance of Sequence-to-Sequence Models for the applications presented in earlier

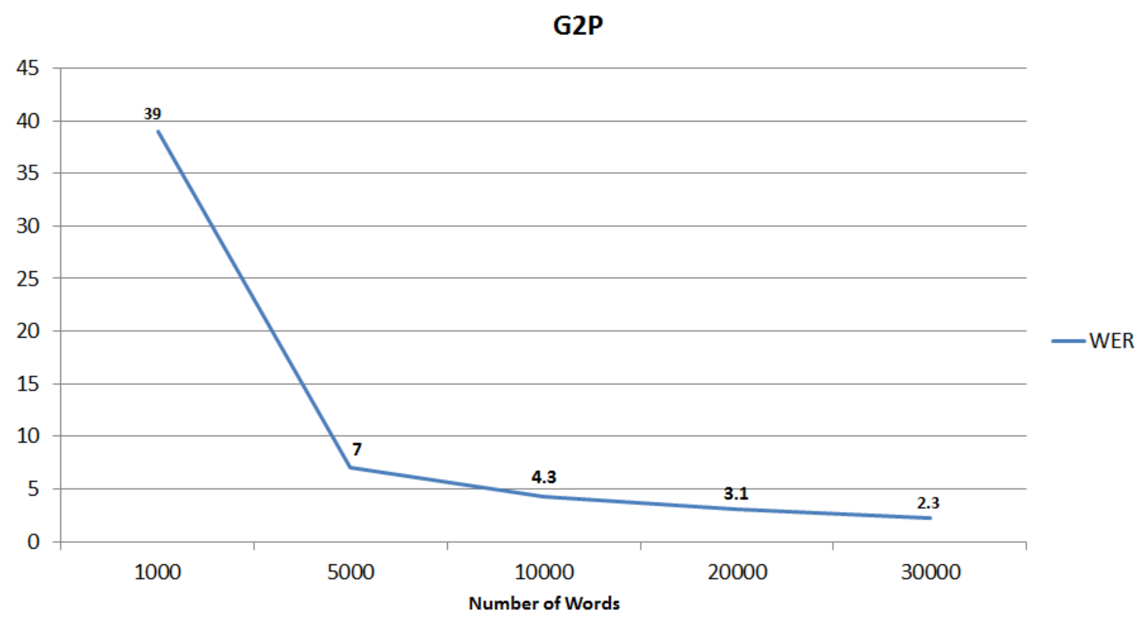

Figure 5.6: G2P model trained with datasets of different sizes.

sections. For each task, the datasets were divided in subsets containing 1000, 5000, 10000, 20000 and 30000 words. Each subset was further separated in a training set (85%), validation set (10%) and test set (5%). The other parameters were the same as the simulations detailed before.

The results for G2P conversion are showed in Figure 5.6. It can be seen that for datasets containing less than 5000 words, the model is not able to have a good performance. For a set containing more than 10000 words, it starts to have an accuracy similar to other data-driven and rule-based approaches described in Section 5.2.1. For 30000 words it achieves a WER of 2.3%. In Figures 5.7 and 5.8 results for the Syllabification and Stress Determination tasks are presented. Similarly to the G2P Conversion, the curve steepness starts to be reduced after 10000 words. The WER for Syllabification at 30000 words is equal to 1.7%. Then, for Stress Determination, the WER achieves the lowest value of 0.4% for the set with 30000 words.

As a conclusion, we can see clearly that the training set size have a strong influence on the final performance of a Sequence-to-Sequence Model architecture. This pattern is repeated for other data-driven approaches. It can be seen that if a training set with 10000 words were used, the model would not be able to outperform the rule-based algorithm for the G2P conversion. This reinforces the need of broad and quality datasets for the use of data-driven approaches to solve Natural Language Processing problems.

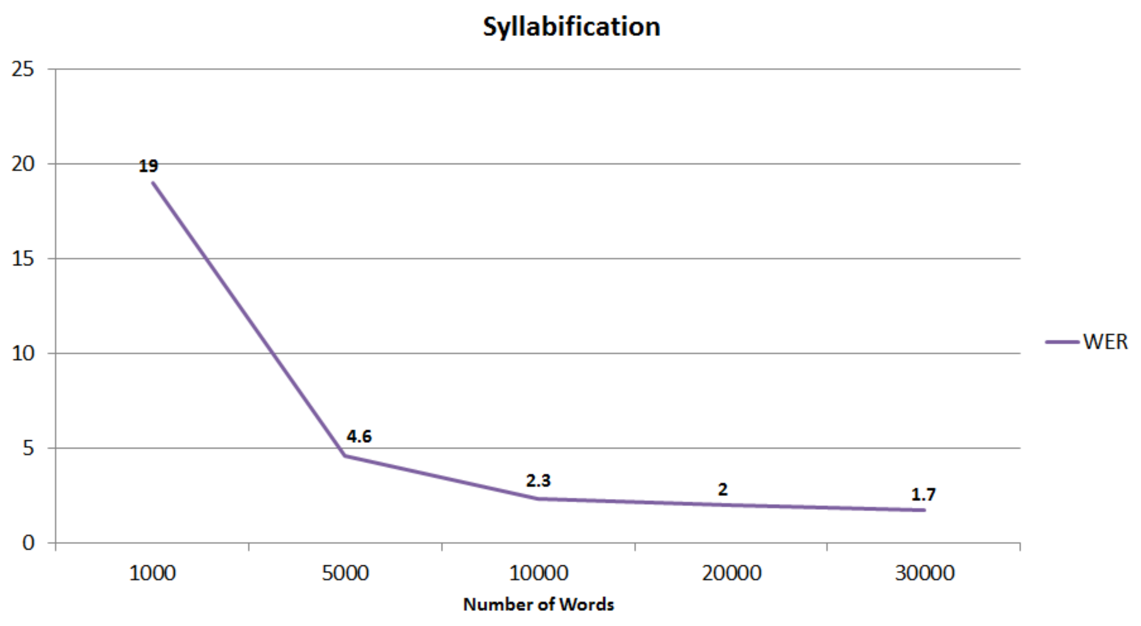

Figure 5.7: Syllabification model trained with datasets of different sizes.

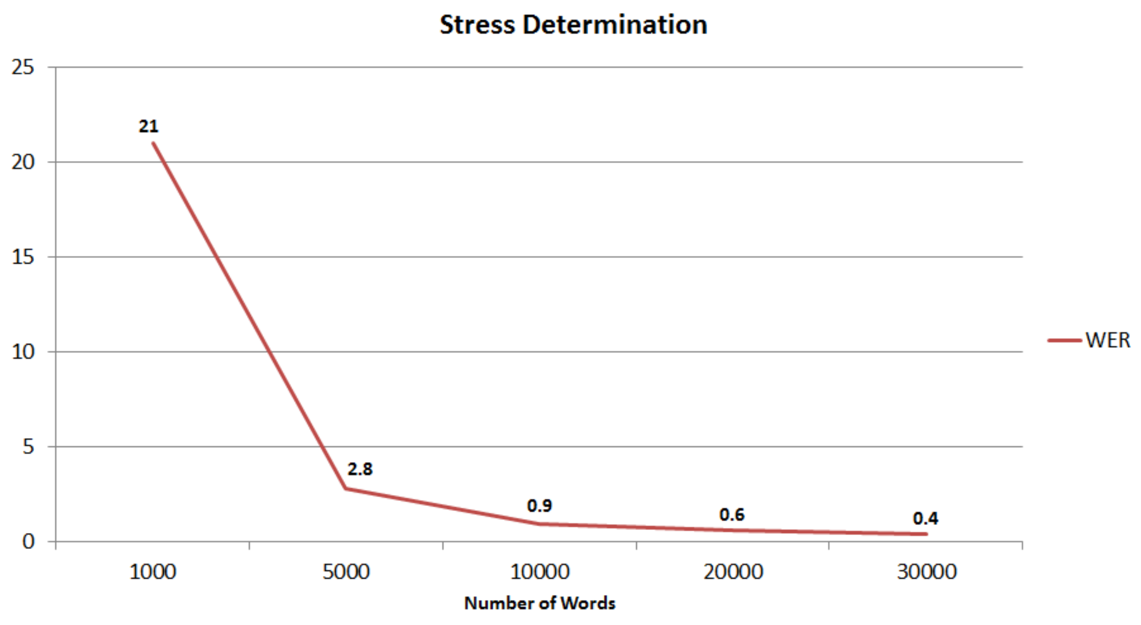

Figure 5.8: Stress Determination model trained with datasets of different sizes.

Chapter 6

Conclusions and Future Works

6.1 Conclusions

This dissertation was dedicated to exploring the use of data-driven solutions and compare their performances with rule-based approaches in the Natural Language Processing algorithms of Brazilian Portuguese Text-to-Speech systems.

First of all, Chapter 2 discussed the theoretical background of TTS systems. The Text Analysis block was detailed and rule-based algorithms for BP presented. The data-driven approaches for the NLP algorithms were also described, and three solutions were chosen to perform the comparison for the G2P conversion task: Weight Finite-State Tranducers [10], Joint-Sequence Models [9] and Sequence-to-Sequence Models [11]. The theory behind these solutions were explained in detail. Finally, the traditional approaches to solve the Speech Waveform generation block were described.

Following, in Chapter 3 a new rule-based algorithm for the BP G2P conversion is proposed. The proposed rules are based on the algorithm described in [6] and aim mainly to solve the nasalization and the errors in the conversion of the graphemes <e> and <o>. Comparisons with the results of the previous algorithm show that the additional rules proposed improved the general performance of the G2P converter from 97.44% to 99.18%. Informal subjective tests realized with an HMM-based TTS system that was trained and tested using the set of proposed rules also showed an improvement on the naturalness of the generated speech.

In Chapter 4 we explore the flexibility of the Sequence-to-Sequence architecture to propose three models. The first model insert the stress information to the Sequence-to-Sequence Model used to the G2P conversion. It is expected that the performance of the data-driven converter improves as this information is also used in the rule design of knowledge-based converters. The other two proposed models explores the applicability of Sequence-to-Sequence Models to the Syllabification and the Stress Determination problems. For the Syllabification, syllable boundaries characters are embedded in the decoding chain and the task is modeled as a sequence-mapping problem. In the Stress Determination model, stress marker characters are included in the decoding chain and the problem is also modeled as a sequence-mapping problem.

Lastly, in Chapter 5 the datasets created and the experiments performed were presented. For this work, three datasets containing 33737 words extracted from the Ispell dictionary [13] were created. One for each of the following tasks: G2P conversion, Syllabification and Stress Determination. With these datasets, four experiments were implemented. The first experiment was the comparison between three data-driven techniques and a rule-based algorithm proposed in Chapter 3 for the G2P problem. This experiment showed that data-driven algorithms are able to outperform the rule-based algorithm for the G2P conversion. The best rates were observed for the Sequence-to-Sequence Models. They achieved a PER of 0.52% and a WER of 2.28%. In comparison to these numbers, the rule-based algorithm achieved PER and WER equal to 0.82% and 3.26% respectively. The other two data-driven algorithms yield results similar to the rule-based algorithm. The results were calculated using 11-fold cross validation. The second experiment compared the Sequence-to-Sequence Model, that achieved the best performance in the first experiment, with a similar model proposed that included the stress information. The results showed that the model with the stress information outperformed the one without this information for the G2P conversion. The proposed model achieved a PER of 0.44% and a WER of 1.91% using 11-fold cross validation. The third experiment explored the comparison between the proposed Sequence-to-Sequence Model and the rule-based algorithm [6] for Syllabification. It is showed that the proposed model achieved a WER of 1.70% against 4.15% from the rule-based approach. The comparison was performed using 5-fold cross validation. Finally, the fourth experiment compared the proposed data-driven model and the rule-based algorithm [6] for Stress Determination. The results showed that the proposed model is not able to outperform the rule-based algorithm for this task. The proposed model achieved WER equal to 0.45% againt 0.03% of the rule-based solution using 5-fold cross validation.

As a general conclusion, this dissertation showed that data-driven solutions are able to achieve state-of-the-art results on the Natural Language Processing algorithms presented in a Brazilian Portuguese Text-to-Speech system. Also, the proposed data-driven models outperformed rule-based algorithms for the G2P conversion and Syllabification problems.

6.2 Future Works

It is interesting to observe that the results obtained with the Sequence-to-Sequence Models in this dissertation can still be improved. [11] achieved the best performance for English with a different architecture that used the alignment information. In our case, results were obtained in a simple encoder-decoder architecture with 2 layers of 64 units. So, as a future work a detailed analysis on the disparity in performance for different architectures and distinct parameters is important.

Another interesting work could be the implementation of a Text Analysis block that uses the data-driven approaches presented in this dissertation. Also, multiple information can be used to create models with richer feature sets. For example, the stress and syllable information could be used together in the G2P model with a multitask architecture, as proposed in [69].

To spread the research using data-driven approaches for these tasks the release of the datasets created with an open source license, such as the one used by the CMUdict [68], is crucial. The release can disseminate the collaboration between linguistic specialists in BP to increase the size and the quality of the datasets. The release will also stimulate other researchers will to try and develop new algorithms in this area.

At last, the creation of a new dataset with sentence-level structures that comprise the co-articulation effects and the homograph disambiguation can be used to train new Sequence-to-Sequence Models. These models can be trained with small modifications to the prevailing model and can provide an important feature that will be important during the development of TTS systems.

Bibliography

[1] BARBOSA, F., PINTO, G., JR, F. G. V. R., et al. "Grapheme-Phone Transcription Algorithm for a Brazilian Portuguese TTS". In: *Propor*, pp. 23–30, 2003.

[2] SILVA, D. C., DE LIMA, A. A., MAIA, R., et al. "A rule-based grapheme-phone converter and stress determination for Brazilian Portuguese natural language processing". In: *ITS*, pp. 992–996, 2006.

[3] MARQUIAFAVEL, V., BOKAN, A., ZAVAGLIA, C. "PETRUS: A rule-based grapheme-to-phone converter for Brazilian Portuguese". In: *Propor-Demo*, 2014.

[4] VASILEVSKI, V. "Phonologic Patterns of Brazilian Portuguese: a grapheme to phoneme converter based study". In: *Proc. of the EACL 2012 Workshop on Computational Models of Language Acquisition and Loss*, pp. 51–60, 2012.

[5] SILVA, D. C., BRAGA, D., JR, F. G. V. R. "Separação das sílabas e Determinação da Tonicidade no Português Brasileiro". In: *XXVI Simpósio Brasileiro de Telecomunicações*, 2008.

[6] SILVA, D. C. *Algoritmos de processamento da linguagem e síntese de voz com emocões aplicados a um conversor texto-fala baseado em HMM*. Tese de D.Sc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil, 2011.

[7] MONTE, A., RIBEIRO, D., NETO, N., et al. "A rule-based syllabification algorithm with stress determination for Brazilian Portuguese Natural Language Processing". In: *Proc. of International Congress of Phonetic Sciences*, pp. 1418–1421, 2011.

[8] NETO, N., ROCHA, W., SOUSA, G. "An open-source rule-based syllabification tool for Brazilian Portuguese", *Journal of the Brazilian Computer Society*, v. 21, 2015.

[9] BISANI, M., NEY, H. "Joint-Sequence Models for Grapheme-to-Phoneme Conversion", *Speech Communication*, v. 50, n. 5, pp. 434–451, maio 2008.

[10] NOVAK, J., DIXON, P., MINEMATSU, N., et al. "Improving WFST-based G2P Conversion with Alignment Constraints and RNNLM N-best Rescoring". In: *Proc. Interspeech*, 2012.

[11] YAO, K., ZWEIG, G. "Sequence-to-Sequence Neural Net Models for Grapheme-to-Phoneme Conversion". In: *arXiv:1506.00196 [cs.CL]*, 2015.

[12] TEIXEIRA, A., OLIVEIRA, C., MOUTINHO, L. C. "Machine Learning of European Portuguese Grapheme-to-Phone Conversion using a Richer Feature Set", *Revista do DETUA*, v. 4, n. 6, mar. 2006.

[13] "Dictionary Ispell". Disponível em: <`https://www.ime.usp.br/~ueda/br.ispell/summary.html`>.

[14] SUTSKEVER, I., VINYALS, O., LE, Q. V. "Sequence to Sequence Learning with Neural Networks". In: *arXiv:1409.3215 [cs.CL]*, 2014.

[15] KOHAVI, R. "A study of cross-validation and bootstrap for accuracy estimation and model selection". In: *International joint Conference on artificial intelligence*, pp. 1137–1145, 1995.

[16] HUANG, X., ACERO, A., HSIAO-WUEN, H. *Spoken Language Processing, a guide to theory, algorithm and system development.* Prentice Hall Inc, 2001.

[17] TAYLOR, P. *Text-to-Speech Synthesis.* Cambridge University Press, 2009.

[18] "Definition of Syllabe - Wikipedia". Disponível em: <`https://en.wikipedia.org/wiki/Syllable`>.

[19] POLYAKOVA, T. *Grapheme-to-Phoneme Conversion in the Era of Globalization.* Ph.D. dissertation, Universitat Politecnica de Catalunya, Barcelona, Spain, 2014.

[20] CHOMSKY, N., HALLE, M. *Sound Pattern of English.* Harper and Row, 1968.

[21] VASILEVSKI, V. *Construção de um programa computacional para suporte a pesquisa em fonologia do portugiês do Brasil.* Tese de D.Sc., UFSC, Florianópolis, SC, Brasil, 2008.

[22] ZEN, H., SAK, H. "Unidirectional Long Short-Term Memory Recurrent Neural Network with Recurrent Output Layer for Low-Latency Speech Synthesis". In: *Proc. of ICASSP*, pp. 4470–4474, 2015.

[23] BARROS, M. J., WEISS, C. "Maximum Entropy motivated Grapheme-to-Phoneme, stress and syllable boundary prediction for portuguese text-to-speech". In: *IV Jornadas en Tecnologia del Habla*, 2006.

[24] TRANCOSO, I., VIANA, M., SILVA, F., et al. "Rule-based vs. Neural Network Based Approaches to Letter-to-Phone Conversion for Portuguese Common and Proper Names". In: *Proc. ICSLP*, 1994.

[25] CASEIRO, D., TRANCOSO, I., OLIVEIRA, L., et al. "Grapheme-to-Phoneme using finite-state transducers". In: *IEEE Workshop on Speech Synthesis*, 2006.

[26] JENSEN, K. J., RIIS, S. "Self-organizing letter code-book for text-to-phoneme neural network model". In: *Proc. Int. Conf. on Spoken Language Processing*, pp. 318–321, 2000.

[27] HAKKINEN, J., SUONTAUSTA, J., RIIS, J., et al. "Assessing text-to-phoneme mapping strategies in speaker independent isolated word recognition", *Speech Communication*, v. 41, n. 2, pp. 455–467, out. 2003.

[28] RAO, K., PENG, F., SAK, H., et al. "Grapheme-to-Phoneme Conversion using Long Short-Term Memory Recurrent Neural Networks". In: *Proc. ICASSP*, 2015.

[29] ANDERSEN, O., KUHN, R., LAZARIDES, A., et al. "Comparison of two tree-structured approaches for grapheme-to-phoneme conversion". In: *Proc. Int. Conf. on Spoken Language Processing*, pp. 1700–1703, 1996.

[30] PAGEL, V., LENZO, K., BLACK, A. W. "Letter-to-sound rules for accented lexicon compression". In: *Proc. Int. Conf. on Spoken Language Processing*, pp. 2015–2018, 1998.

[31] MARCHAND, Y., DAMPER, R. I. "A multistrategy approach to improving pronunciation by analogy", *Computational Linguistics*, v. 26, n. 2, pp. 195–219, 2000.

[32] BELLEGARDA, J. R. "Unsupervised, language-independente grapheme-to-phoneme conversion by latent analogy", *Speech Communication*, v. 46, n. 2, pp. 140–152, 2005.

[33] CHEN, S. F., GOODMAN, J. "An empirical study of smoothing techniques for language modeling", *Computer Speech and Language*, v. 13, n. 4, pp. 359–394, out. 1999.

[34] BESLING, S. "Heuristical and statistical methods for grapheme-to-phoneme conversion". In: *Konferenz zur Verarbeitung naturlicher Sprache (KONVENS)*, pp. 24–31, 1994.

[35] DELIGNE, S., BIMBOT, F. "Language Modeling by variable length sequences: Theoretical formulation and evaluation of multigrams". In: *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 1995.

[36] DELIGNE, S., BIMBOT, F. "Inference of variable-length acoustic units for continuous speech recognition", *Speech Communication*, v. 1, pp. 223–241, 1997.

[37] MOHRI, M., PEREIRA, F. C. N., RILEY, M. "Weighted Finite-State Transducers in Speech Recognition", *Computer Speech and Language*, v. 16, n. 1, pp. 69–88, 2002.

[38] JIAMPOJAMARN, S., KONDRAK, G., SHERIF, T. "Applying Many-to-Many Alignments and Hidden Markov Models to Letter-to-Phoneme Conversion". In: *Proceedings of NAACL HLT*, pp. 372–379, 2007.

[39] JIAMPOJAMARN, S., KONDRAK, G. "Letter-to-Phoneme Alignment: An Exploration". In: *Proceedings of the ACL*, pp. 780–788, 2010.

[40] HOCHREITER, S., SCHMIDHUBER, J. "Long Short-Term Memory", *Neural Computation*, v. 9, n. 8, pp. 1735–1780, 1997.

[41] "Understanding LSTM Networks". Disponível em: <`http://colah.github.io/posts/2015-08-Understanding-LSTMs/`>.

[42] CHO, K., V MERRIENBOER, B., GULCEHRE, C., et al. "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation". In: *Proc. EMNLP*, 2014.

[43] YAO, K., COHN, T., VYLOMOVA, K., et al. "Depth-Gated LSTM". In: *arXiv:1508.03790 [cs.NE]*, 2015.

[44] GREFF, K., SRIVASTAVA, R. K., KOUTNIK, J., et al. "LSTM: A Search Space Odissey". In: *arXiv:1503.04069 [cs.NE]*, 2015.

[45] ADSETT, C. R., MARCHAND, Y., KESELJ, V. "Syllabification rules versus Data-driven methods in a Language with Low Syllabic Complexity: The Case of Italian", *Computer Speech and Language*, 2009.

[46] ADSETT, C. R., MARCHAND, Y., KESELJ, V. "A comparison of Data-Driven Automatic Syllabification Methods", *SPIRE 2009, LNCS 5721*, pp. 174–181, 2009.

[47] ALTUN, Y., TSOCHANTARIDIS, I., HOFMANN, T. "Hidden Markov Support Vector Machines". In: *Proceedings of ICML*, pp. 3–10, 2003.

[48] DAMPER, R. I., MARCHAND, Y., ADAMSON, M. J., et al. "Evaluating the Pronunciation Component of Text-to-Speech Systems for English: a performance comparison of different approaches", *Computer Speech and Language*, v. 13, n. 2, pp. 155–176, 1999.

[49] DOU, Q., BERGSMA, S., JIAMPOJAMARN, S., et al. "A ranking approach to Stress Prediction for Letter-to-Phoneme Conversion". In: *Proceesing of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP*, pp. 118–126, 2009.

[50] HUNT, A., BLACK, A. W. "Unit selection in a concatenative speech synthesis system using a large speech database". In: *Proc. ICASSP*, pp. 373–376, 1996.

[51] TOKUDA, K., NANKAKU, Y., TODA, T., et al. "Speech Synthesis Based on Hidden Markov Models", *Proceedings of the IEEE*, v. 101, n. 5, pp. 1234–1252, 2013.

[52] ZEN, H., SENIOR, A., SCHUSTER, M. "Statistical Parametric Speech Synthesis using deep neural networks". In: *Proc. ICASSP*, pp. 7962–7966, 2013.

[53] BLACK, A. W. "Unit Selection and Emotional Speech". In: *Proceedings of Eurospeech*, pp. 1649–1652, 2003.

[54] KARAISKOS, V., KING, S., CLARK, R. A. J., et al. "The Blizzard Challenge 2008". In: *Proc. Blizzard Challenge Workshop*, 2008.

[55] GALES, M., YANG, S. "The Application of Hidden Markov Models in Speech Recognition", *Foundations and Trends in Signal Processing*, v. 1, n. 3, pp. 195–308, 2007.

[56] HINTON, G., DENG, L., YU, D., et al. "Deep Neural Networks for Acoustic Modeling in Speech Recognition", *IEEE Signal Processing Magazine*, v. 29, n. 6, pp. 82–97, 2012.

[57] "Speech Assessment Methods Phonetic Alphabet (SAMPA)". Disponível em: <`https://www.phon.ucl.ac.uk/home/sampa/`>.

[58] LEVENSHTEIN, V. "Binary codes capable of correcting deletions, insertions, and reversals", *Soviet Physics Doklady*, v. 10, pp. 707–710, 1966.

[59] XU, K., BA, J. L., KIROS, R., et al. "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention". In: *arXiv:1502.03044 [cs.LG]*, 2016.

[60] MARCHAND, Y., DAMPER, R. "Can syllabification improve pronunciation by analogy of english?" *Natural Language Engineering*, v. 1, n. 1, pp. 1–25, 2005.

[61] REICHEL, U. D., SCHIEL, F. "Using morphology and phoneme history to improve grapheme-to-phoneme conversion". In: *Proceedings of InterSpeech*, 2005.

[62] STREINER, D. L. "Mantaining Standards: Differences between the Standard Deviation and Standard Error, and when to use each", *The Canadian Journal of Psychiatry*, v. 41, pp. 498–502, 1996.

[63] PUKELSHEIM, F. "The three sigma rule", *American Statistician*, v. 48, pp. 88–91, 1994.

[64] "WFST-based (Phonetisaurus) G2P Open Source code". Disponível em: <`https://code.google.com/p/phonetisaurus/`>.

[65] "MIT Language Model Open Source code". Disponível em: <`https://github.com/mitlm/mitlm`>.

[66] "Joint Sequence Models (Sequitur) G2P Open Source code". Disponível em: <`https://www-i6.informatik.rwth-aachen.de/web/Software/g2p.html`>.

[67] "Sequence-to-Sequence Models Open Source code". Disponível em: <`https://github.com/cmusphinx/g2p-seq2seq`>.

[68] "CMUDict". Disponível em: <`https://github.com/cmusphinx/cmudict`>.

[69] LUONG, M., LE, Q. V., SUTSKEVER, I., et al. "Multi-task Sequence to Sequence Learning". In: *Proc. ICLR*, 2016.