Inference of pronunciation difficulty from non-native data

João Pedro de Sousa Correia

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisors: Prof. Isabel Maria Martins Trancoso
Xavier Anguera

January 2021

Declaration

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Acknowledgments

I would like to thank my family for all the support that they gave me, during these years.

Abstract

ELSA Corp. developed a CAPT (Computer Aided Pronunciation Training) system that assists its users to improve their American English accent. In order to develop exercises appropriate for the level of its users, it is important to have a metric capable of assessing the difficulty of their exercises, according to the user's proficiency level. Therefore, the objective of this thesis is to develop a system capable of determining the pronunciation difficulty associated with a certain utterance and its phonemes, for a Vietnamese student of English. Our model uses a Neural Network in order to forecast the probabilities associated to how competently the user pronounce each of the utterance's phonemes. Then, using these probabilities, the system computes the difficulty score associated to the phoneme and the difficulty score associated to the utterance. In the end, we have a system able to receive as input an utterance and the proficiency level of the user. Then, the system outputs difficulty scores for the utterance and its phonemes.

Keywords

Pronunciation Difficulty; Phonemes; Neural Networks; Computer Aided Pronunciation Training

Resumo

A ELSA Corp. desenvolveu um sistema CAPT que auxilia os seus utilizadores a melhorar o seu sotaque Inglês Americano. De forma a criar exercícios apropriados ao nível de proficiência dos seus utilizadores, torna-se importante desenvolver uma métrica capaz de avaliar automaticamente a dificuldade dos seus exercícios, de acordo com o nível de proficiência do utilizador. Assim, o objetivo da nossa tese é desenvolver um sistema capaz de determinar automaticamente a dificuldade que um estudante Vietnamita de Inglês, com um determinado nível de proficiência, terá em pronunciar uma determinada frase. O nosso modelo utiliza redes neuronais de forma a auxiliar o cálculo das probabilidades associadas a quão corretamente um utilizador pronunciará um certo fonema da frase. Essas probabilidades serão depois utilizadas de forma a calcular a pontuação de dificuldade para cada fonema e para a frase. No final, teremos um sistema que recebe como entrada uma frase e o nível de proficiência do utilizador. A partir destes dados, o sistema produz pontuações associadas à dificuldade de pronúncia para a frase e para cada um dos seus fonemas.

Palavras Chave

Dificuldade de pronúncia; Fonemas; Redes Neuronais; Treino de Pronúncia Auxiliado por Computador

Contents

List of Figures

List of Tables

List of Algorithms

Acronyms

NN	Neural Network
SVM	Support Vector Machine
DS	Difficulty Score
SSS	Speech Server Score
pdf	probability distribution function
SAE	Standard American English
IPA	International Phonetic Alphabet
MoA	Manner of Articulation
PoA	Place of Articulation
ML	Machine Learning
RNN	Recurrent Neural Network
LSTM	Long-short Term Memory

1 Introduction

Contents

In this chapter we will discuss the motivation and objectives of this thesis, we will present the company ELSA Corp that helped us developed this project and we will also present the structure of this thesis.

1.1 Motivation

Nowadays, English is the language with most non-native speakers in the world and has become the default language for people with different idioms to communicate. Therefore, it has become valuable to learn this language whether to easily enter the job market, to study abroad or to simply travel around the world.

In order to learn a new language people have to study and familiarise themselves with new linguistic rules and memorise thousands of unfamiliar words, which makes learning a new language a challenging and time consuming task.

An important but often overlooked topic when studying a language is pronunciation. Perfecting a native accent can be one of the most difficult aspects of studying a language.

This problem is particularly evident when the student has a native language (L1) with very different phonological characteristics from the target language (L2). In these situations, it is common to observe students with an high theoretical knowledge of a language that still retain their native accent. In some particular cases, the mannerisms of their foreign accent might be an actual hindrance for a comprehensible speech. In these cases it is particularly important to study the target accent. However, even if a student does not have difficulties to communicate with a native speaker, improving his accent might still be advantageous. Possessing a native level accent helps removing the language barrier and might facilitate the integration of a person in a new community or workplace. Also, people with a native accent tend to be perceived as more knowledgeable, which might also be advantageous in a workplace.

ELSA Corp. is a company that strives to improve the Standard American English (SAE) accent of their clients. In order to accomplish this goal they developed a software, that offers their users several exercises for them to practice. As a student finishes an exercise his attempt is automatically evaluated and assigned with a score, that reflects how approximate he was from the SAE accent. In addition to the score, the software also provides a simple report with the errors committed and advice on how to correct them.

In order to create the exercises for their students, ELSA has a team of linguists that assigns their exercises according to the users proficiency level. In order to help them with this task, it would be useful to have a tool capable of measuring the exercises difficulty.

1.2 Objectives

The objective of this thesis is to develop a software that automatically assigns a score to a certain input text, that is reflective of the difficulty that a Vietnamese person, with a particular proficiency level, would have to simulate the SAE accent.

In order to develop this software, we have access to a corpus that contains ELSA's exercises and the evaluation that multiple Vietnamese users with different proficiency levels obtained.

Each exercise of the corpus has associated a word or sentence and its transcription. The corpus also contains the grades that multiple users obtained. For each user attempt it is associated his proficiency level and the grades obtained for the sentence and for each of its phonemes.

With this corpus, we will basically train an algorithm that is able to generalise the information present in the data set and then apply it for any input text. In the end, the algorithm should be able to obtain a score for the sentence and for each of its phonemes.

In summary, the algorithm should be able to take as input a certain sentence and proficiency level of the user and in return output a difficulty score for the utterance and its phonemes. This difficulty scores should also be representative of its pronunciation difficulty for a Vietnamese user with a certain proficiency level.

1.3 Thesis Structure

This report is divided in six parts: Introduction, Related Work, Data Analysis, Methodology, Experiment Setup and Conclusions.

In the Related Work Chapter, we will discuss published papers from other authors, which we used some elements for our thesis project.

It is important to note that in the Related Work Chapter we will approach topics that are very extensive and would take too much of our thesis to describe. So we will summarise some topics and only describe some of the information that we find useful for our particular problem.

In the Data Analysis Chapter, we will describe the corpus in detail and present some statistics in order to understand the data, the results obtained and some of the decisions we made.

In the Methodology Chapter, we will describe in detail the methods that we used to develop our thesis project.

In the Experiment Setup Chapter, we will present the results obtained for the approach described in the previous chapter and we will also discuss some experiments performed in order to better tune the model.

Finally, in the Conclusions Chapter we will present and analyse the results we obtained. We will also discuss the experiments we conducted, the limitations of our system and some improvements that could be done to our system in the future.

2 Related Work

Contents

The literature on assessing pronunciation difficulty from non-native data is not very extensive. Most of the bibliography about pronunciation difficulty, focus on identifying phonological characteristics of English that cause difficulties for Vietnamese students.

Although, the literature on this theme is not very extensive, there are still research work that relates to our problem. An example of a research problem that has a lot of similarities to our problem is readability assessment, which is a very studied area.

This Chapter is divided in three sections. In the first section, we will compare the English and Vietnamese phonology, and discuss some of the difficulties Vietnamese students usually have. In the second section, we will present a research work that has similar objectives to ours, but for Korean students. In the third section, we will discuss research work on readability assessment.

2.1 Comparison between English and Vietnamese Phonology

Speech results from the combination of multiple sounds, following certain language rules. The area of linguistics that studies these sounds and rules is phonology. As a consequence, a lot of aspects that determine accent are studied in this area, such as: phonemes, rhythm, syllable structure and intonation. As a result, it becomes essential to understand certain phonology concepts in order to evaluate pronunciation difficulty. In the section 2.1, we will present some basic concepts that will be useful for this thesis.

2.1.1 English Phonemes

A phoneme is "the smallest phonetic unit in a language that is capable of conveying a distinction in meaning" [3]. A phoneme can be represented using multiple notations, but the international standard is to use International Phonetic Alphabet (IPA). Another commonly used notation is ARPABET that is equivalent to IPA but uses letters from the English alphabet instead of the greek alphabet, which makes it more appealing to use computationally.

We use these notations instead of letters (graphemes), because graphemes are phonetically ambiguous. In other words a grapheme can represent different sounds and is context dependent, whereas a phoneme corresponds to a specific sound regardless of the context.

Each language has its own phoneme inventory, so phonemes that exist in a language might not be present in other languages and even the same language can have different phoneme sets according to the local accent. For example, the English from England has forty four distinct phonemes, but in the SAE the phonemes: ʊə/, /ɪə/ and /ɪə/ do not exist, so this number is reduced to forty one [4].

Phonemes can be divided in two different groups: vowels and consonants. With consonants, usually, there is obstruction of the air flow in the vocal track, contrary to what happens with vowels [5]. However, in some cases this distinction is not so clear, for example, with the consonants: /j/ and /w/ we have to analyse the pattern of the sounds in the word and syllable in order to make the distinction. In both of the previous examples we expect that the sound that follows to be a vowel, so we can consider them consonants, these type of consonants are glides or semi-vowels [6].

In order to describe the phonemes in a language we can continue to divide them according to their characteristics. In the rest of this section, we will explain in more detail how vowels and consonants are characterised and differ from each other.

Vowels

According to [6], we can differentiate vowels using two parameters related to the position of the tongue and one parameter related to the position of the lips. Thereby, a vowel can be characterised by:

- Which part of the tongue is more elevated. Therefore, following this criteria a vowel can be defined as: Front, Central or Back vowel.
- How close the tongue is to the palate. If they are: very close they are closed vowels, in the opposite situation they are open vowels. There are also vowels classified as close-mid and open-mid, in the situations the tongue is not as close or far from the palate, respectively.
- The position of the lips, if we round the lips we classify the vowel as rounded, otherwise we classify it as unrounded.

In the table 2.1, we present the classifications for English and Vietnamese vowels.

Table 2.1: Simple Vowels in the phoneme inventory of SAE. In red there are the phonemes that exist in English, but not in Vietnamese.

	Front		Central		Back	
	Unrounded	Rounded	Unrounded	Rounded	Unrounded	Rounded
Close	i, ɪ					u, ʊ
Close-mid						
Open-mid	ɛ		ɜ		ʌ	ɔ
Open	æ				ɑ	

The vowels presented in 2.1 are simple vowels, because they only have one phase of pronunciation and the classifications in the table above are therefore useful to classify them. However, there are certain vowels that have two different phases, where the first part starts with a sound of a particular pure vowel and ends with the sound of other. [6]

The diphthong inventories are completely different, SAE has five diphthongs: /eɪ/, /əʊ/, /ɔɪ/, /aʊ/ and /aɪ/, on the other hand Vietnamese has three diphthongs: /ie/, /uo/ and /ɯɤ/. [7]

Finally there are also the rhotic vowels such as: ɝ(ex: hurt) and ɚ(ex: another), which place an r in the end of the vowel. Both of these two phonemes also do not exist in the Vietnamese phonetic inventory. [6] [2]

Consonants

According to [6], consonants can be classified by:

- the Place of Articulation (PoA): Represents the position, in the vocal track, where the obstruction of air flow occurs.
- Manner of Articulation (MoA): As the name alludes, it classifies a phoneme according to how the airflow flows through the vocal track.
- Voicing: This classification is related to the vibration in the vocal track. If there is vibration the phoneme is voiced, otherwise is unvoiced.

In the table 2.2, we present the classifications for English and Vietnamese consonants.

Table 2.2: Consonants in the phoneme inventory of English and Vietnamese. In red there are the phonemes that exist in English, but not in Vietnamese.

PoA / MoA	Bilabial	Labiodental	Dental	Alveolar	Postalveolar	Retroflex	Palatal	Velar	Uvular	Glottal
Plosive	p b			t d				k g		
Nasal	m			n			ŋ			
Trill										
Flap				ɾ						
Fricative		f v	θ, ð	s z	ʃ, ʒ					h
Affricates				tʒ, dð						h
Approximant							j			
Lateral Aproximant				l						

It is important to note that although we are presenting a very strict classification for consonants and vowels, in reality phonemes with the same classification can present slight variations. These variations are called allophones and they might occur due to: the position of the phoneme in a word, the position in the syllable, the syllable type ... .

Difficult phonemes to pronounce

From the previous analysis, we can observe that the phoneme inventory of Vietnamese and English are different. As a consequence, some phonemes cause more difficulties to pronounce than others. In the table 2.3, we present the phonemes that usually cause problems for Vietnamese students.

Table 2.3: Phonemes difficulties and common mistakes of Vietnamese students. Adapted from: [2]

Phoneme	Confusion
ɵ	t,s
ð	d,z
p	b
g	k
dʒ	z
ʒ	z, dʒ
s	ʃ
v	j
ɪ	i
ɛ	æ
æ	ɛ, a
ʊ	u, ʌ

2.1.2 Syllables

There are at most 3 components in a syllable: Onset, Nucleous and Coda. The Nucleous is the only component every syllable must have, it is also its most sonorant component and in the majority of cases corresponds to a vowel. The Onset is the part of the syllable that preceeds the Nucleous, it is usually composed by one or more consonants.The Coda proceeds the Nucleous and also is usually composed by one or more consonants. Both the Onset or/and Coda might not be a part of a syllable.

The syllable structure in SAE and Vietnamese are different. SAE can have at most three consonant clusters in the coda or onset, whereas in Vietnamese there are not consonant clusters, instead there are vowel clusters. The most common syllable structures are: CV, CVV, V, VV, VVC, CVC and CVVC, where C and V represent a consonant and a vowel, respectively.

The non familiarity of Vietnamese students with some of the SAE syllable structure tends to cause complications in their accent. The nonexistence of clusters in Vietnamese causes that consonants inside clusters to be harder to pronounce than isolated ones. Another difficulty experienced by Vietnamese students are consonants in the end of a syllable, because although the Vietnamese language allows such a syllable structure, it only permits nasals and stops in this position. [2]

In the table 2.4, we present the common difficulties of Vietnamese students, when consonants appear in the final position of the syllable.

Table 2.4: Common mistakes with consonants in the final position of the syllable. Adapted from: [2]

Phoneme	Confusion
b	p
d	t
f	p
v	p, b
s	Omitted , ʃ
ʃ	Omitted
z	s , ʃ
tʃ	ʃ
l	n

2.1.3 Stress and Rhythm

A language can be classified according to its rhythm as: syllable-timed and stress-timed. Syllable-timed languages spend the same time in each syllable, whereas stressed-timed languages the length of the syllable is different according to its stress. In this type of language, stressed syllables usually are lengthier than the non-stressed ones. SAE is a stressed language and Vietnamese is a syllable timed language. [8]

Students from different languages are used to certain rhythm rules, therefore they sometimes experience difficulties assimilating a new set of rules. Vietnamese students usually have these problems, due to the different rhythm of English.

2.1.4 Intonation

In general, people may not know how to describe pitch, but they usually have a basic intuition of its meaning. For example, people understand the difference between a low and a high pitch voice and they expect a man to have a lower pitch than a woman. However, pitch is more complex than this simple notion, in reality there are more variations to pitch and they alternate throughout a sentence. These different variations correspond to different tones. [9]

In physical terms, pitch relates to the frequency of a sound. In pure tones (sounds with just one frequency component) the pitch can be easily characterised by the frequency of that component. However, the majority of sounds have multiple frequency components which also change over time. In these situations, we characterise the pitch by its fundamental frequency at each time point. [10]

Intonation is directly correlated to pitch and corresponds to the variation of pitch in speech. Therefore, different tones are characterised by different pitch variations.

Vietnamese is a tonal language, which means that a change in the tone of a syllable may change the meaning of the word. On the other hand, English is not a tonal language, therefore Vietnamese students learning English do not have to worry as much about tones compared to English students learning Vietnamese. [8]

2.2 Determining a sentence pronunciation difficulty for Korean students

In the problem presented in [11], the authors developed a system capable of automatically assessing the inherent pronunciation difficulty of a certain input text. The algorithm selected to solve this problem was Support Vector Machine (SVM).

In order to train this model they used a corpus with multiple sentences classified in three difficulty classes (Easy, Medium and Hard).

To represent the input sentences they tested multiple features according to the *a priori* knowledge of the Vietnamese students difficulties in simulating the English accent. Therefore, they trained the model multiple times with different sets of features, in order to obtain the set of features that obtains better results.

After testing multiple sets of features, they obtained the best model for the SVM using sixteen features as input:

- **Length Features :**
  1. Number of phonemes.
  2. Number of graphemes.
  3. Number of words.
  4. Average number of phonemes per word.
  5. Maximum number of phonemes in a word.
  6. Average number of graphemes per word.
  7. Maximum number of graphemes in a word.
- **Phoneme Features :**
  1. Number of phonemes that are more prone to errors, according to the bibliography in Korean students pronunciation difficulties in English.
  2. Number of codas in a sentence, excluding from this count words that have a number of codas equal or inferior to two.
  3. Weighted sum with: number of commonly mispronounce phonemes (weigh=1), if the sentence has two codas with two consonants (weight=1), if the sentence has three codas with more than two consonants (weight=3) and if the sentence has three codas with more than two consonants (weight=5).
  4. Maximum cov [1] in a sentence.

5. Average cov in a sentence.
6. Maximum number of syllables in a word.
7. Average number of syllables in a word.

- **Word Frequency Features -** The words were ranked from zero to five according to the frequency in which they appear in a corpus (British National Corpus):
  1. Average frequency rank in the text.
  2. Maximum frequency rank in the text.

Although, the premises of this problem and the problem of our thesis are very similar, there are some significant differences. For example, they try to evaluate the difficulty of a English sentence for Korean students, whereas we evaluate the difficulty for Vietnamese students, other important difference is that their corpus already has the evaluation for each sentence according to their level of difficulty, whereas we have for each sentence multiple scores that evaluate the proficiency of different users when pronouncing that particular sentence.

Furthermore, even though, the problems are different and we applied a different methodology to our problem, there are still some useful aspects that were useful to model our project, such as: machine learning is a valid approach to model our problem and the phonemes present in the sentence are the most important element of determining the pronunciation difficulty, but we can add other information in order to obtain a better model.

2.3 Assess the readability of text using Neural Networks

In order to study a language, it is important to practice reading in that language. However, if the texts available are excessively easy for the students, they will not be able to learn. On the other hand, if the texts are excessively hard the students will not be able to understand most of the text and might feel unmotivated. Therefore, it is important to have an estimation of a certain text difficulty.

A readability measure, evaluates the inherent difficulty of a certain text . [12]

Initially, a lot of readability formulas used to mainly focus on characteristics of the sentence like the word and sentence length. As the problem was further studied, other features of the sentence were included and new formulas emerged.[2].

Machine Learning (ML) algorithms have also been used to assess the readability of text. One of the approaches that uses ML, focus on extracting specific features from the input text and then uses them as

[1]According to [11] represents ”consonant over value” which represents the number of consonants divided by the number of vowels.

[2]There is a vast research on Readability. In this thesis our focus is not on this topic, rather we will focus on the methods used. As a consequence, we will not extend this topic.For more information: [12], [13] and [14].

input for training the algorithm. An example of this approach is present in [15] and [16]. In this research work, the authors used SVM [15] and Naive Bayes [16] algorithms combined with feature engineering techniques in order to obtain a model capable of assessing a readability score for the text and to select the best features for their respective task.

More recently, Neural Networks have also been used to assess the readability of text. One of the advantages of using Neural Network (NN) is that they do not require very specific input features. Using NN we can just input the text (encoded) instead of selecting features for the algorithm. This allows the NN to find patterns that were not explicitly encoded in the feature array and in some cases might find particular patterns that were not yet studied.

In the problem presented in [1], the authors developed Vec2read. Vec2read is an algorithm based on a Multiattentive Neural Network that assesses a readability score for text. In Figure 2.1, we present the Neural Network architecture used in Vec2read.

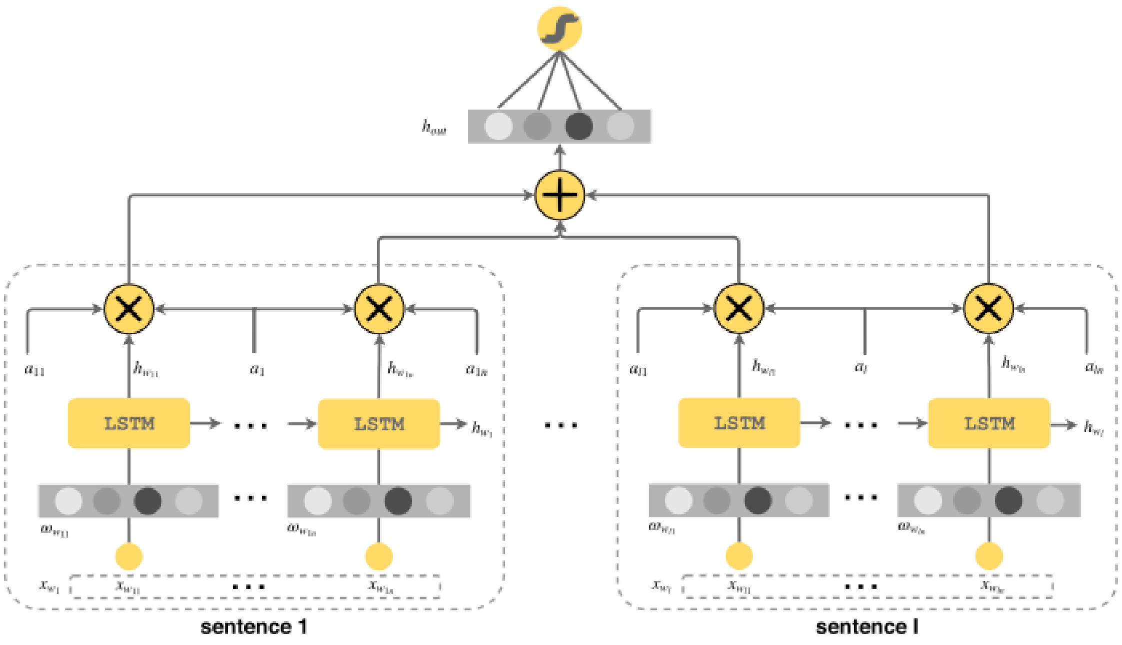

Figure 2.1: Vec2read Neural Network architecture. Extracted from: [1]

As we can observe in Figure 2.1, the text is divided in sentences and each sentence is divided in words. The words' code (w_{ij}) is the concatenation of three different code components: the embedding of the word, the code of the part of speech tag and also a code associated to the morphological characteristics of the word. Each word corresponds to a time-step in a Long-short Term Memory (LSTM) layer.

The attention mechanism in Vec2Read is hierarchical, which means that the system has multiple types of parameters to determine which part of the text it should focus on. It has: a parameter that determines which sentence the system should focus (a_i) and a parameter that determines which word of a sentence the system should focus (a_ij). In Figure 2.2, we present the system that produces the attention for words.

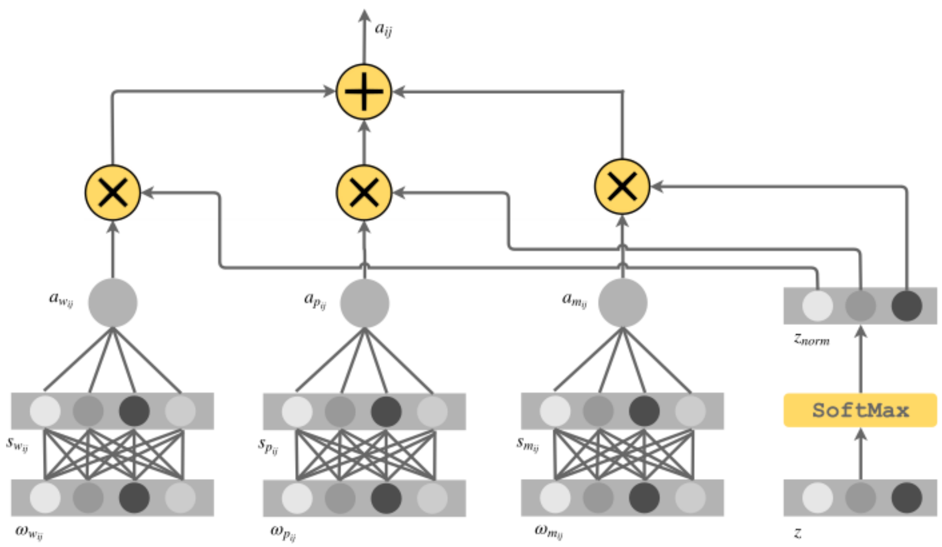

Figure 2.2: Attention Mechanism used for the words. Extracted from: [1]

As we can observe in Figure 2.2, the attention parameters of each word are created also using an attention parameter (${z_{norm}}^{3}$), that allows the NN to pay more attention to one of the components of the word's code: part of speech, embedding or morphological code.

In order to train Vec2Read and evaluate the model, the authors trained the NN using multiple data-sets. These data sets contained multiple input text and their respective output tag that indicates its readability score. Furthermore, each of these data sets had a specific language and for some data sets the readability score was discrete, whereas for others was continuous. The NN had a specific output according to the the tags of the data-set.

Although the premises of this problem and of our thesis are different, there are some similarities. For example, the authors also tried to predict difficulty from text, the difference being the meaning of these difficulties. Other important difference is that in all the cited papers above, the corpus to train the models had a tag or a score, representative of the difficulty of the sentence, to train the model. In our problem, we have multiple scores that indicate the performance of different users when pronouncing an exercise, which also presents a different problem.

[3]The sum of z_{norm} is equal to one and corresponds to the weight of each word component. Furthermore, z_{norm} was created after the parameter z, which was automatically computed during the training and then used as input for the corresponding word.

3

Data Analysis

Contents

In this chapter we will describe the data and analyse it in a statistical point of view. This analysis will then be useful to create the model for our project.

3.1 Corpus Description

As mentioned in previous chapters, ELSA creates multiple exercises for its users to practice. These exercises consist of a sentence (or word) that the user says and which the software records. This recorded audio is then automatically evaluated by the system. Once the system finishes the evaluation, it delivers a simple report with: the score obtained by the user, the errors committed and some advice in order for the user to improve his accent.

Our corpus contains 3334 exercises and the evaluation obtained by multiple Vietnamese users. For each exercise, we have the information regarding the attempts of multiple Vietnamese users. More specifically, for each attempt, the corpus contains the following information:

- **A sentence or word:** Each exercise has its own sentence (or word).
- **Sentence phonetic transcription:** The transcription of the sentence follows the two letter Arpa-Bet transcription code.
- **Stress of the vowels:** For each phoneme it is indicated if the phoneme is a vowel or not. Furthermore, for vowels, there is also information about the type of stress: no stress, primary stress and secondary stress.
- **Exercise Score:** There is a score for the exercise representative of how close the user was to the SAE accent, this score ranges from 0 (unsatisfactory accent) to 100 (perfect accent).
- **Phoneme Score:** There is also a score for each phoneme. This score ranges from 0 (perfect accent) to 1 (unsatisfactory accent).
- **The proficiency of the user:** Value reflective of the user skill. In our project, we divided the proficiency of the user in four levels: low, mid-low, mid-high, high. In this corpus, there are approximately the same number of attempts in all these levels. In Figure 3.2 we can observe the frequency of these intervals.

In total, the data set contains: 123 090 attempts and there is a median of 29 attempts per exercise by users with multiple proficiency levels. Although, not all the exercises have the same number of attempts, in the majority of the exercises the number of exercises is close to the median. In Figure 3.1 we present the number of attempts per exercise.

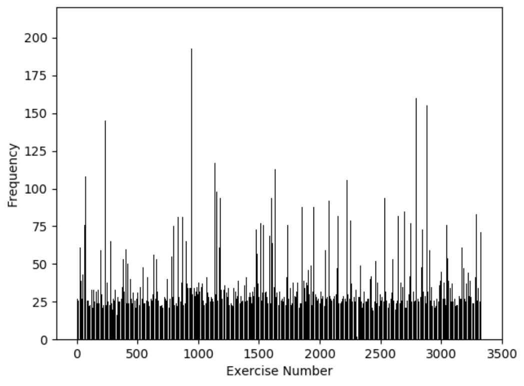

Figure 3.1: Number of attempts in each of the 3334 exercises

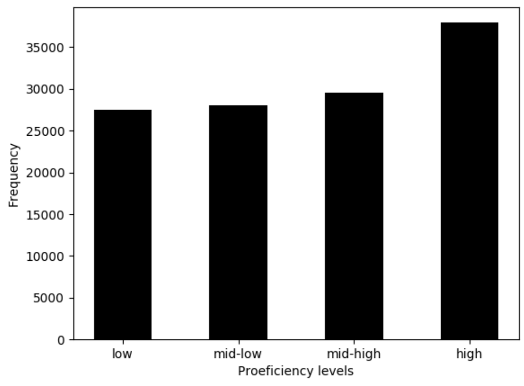

Figure 3.2: Frequency of the proficiency levels

3.2 Relationship between phonemes and their score histogram

Analysing the data, we can observe that the phonemes scores are very polarised. These scores have a bigger tendency of obtaining values close to 0 (perfect) and close to 1 (very bad), than obtaining an intermediate score, as we can observe in the histogram of Figure 3.3.

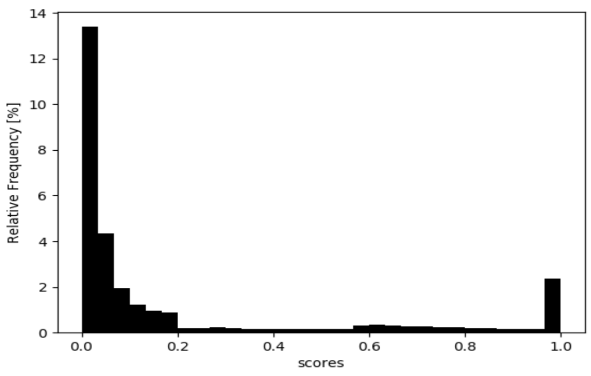

Figure 3.3: Frequency of all phonemes scores

Although, every phoneme has its own score distribution there are similarities that we should take into account. Observing the data, we can infer that phonemes that are easy to pronounce for Vietnamese users tend to have a specific histogram for its scores, whereas phonemes that are harder to pronounce tend to have other type of histogram.

Easy phonemes are likely to have a similar histogram to the general phoneme distribution (Figure 3.3), with the difference that they have less relative frequency of scores near the value 1, as we can observe in Figure 3.6 and 3.7. On the other hand, harder phonemes follow an opposite trend, where the scores close to the value 1 tend to have an increase in relative frequency and the scores close to 0 tend to have a decrease. In the figures: 3.4 and 3.5 we present examples of histograms for easy phonemes and in figures: 3.6 and 3.7 we present examples of difficult phonemes.

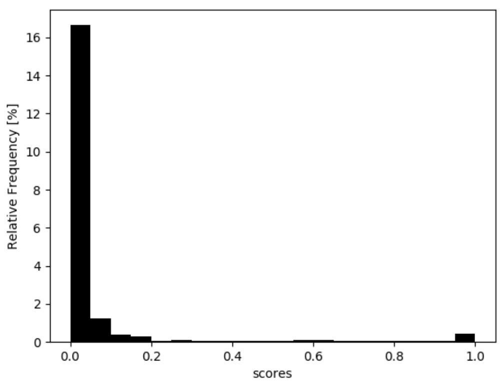

Figure 3.4: Scores' histogram of phoneme m

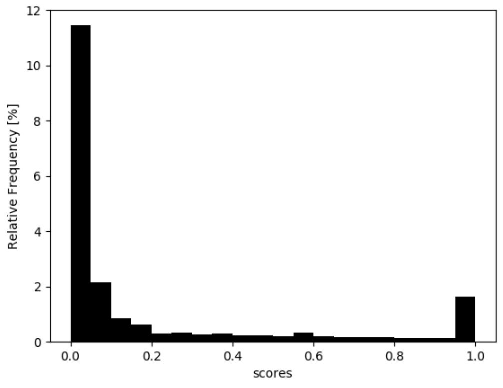

Figure 3.5: Scores' histogram of phoneme eɪ

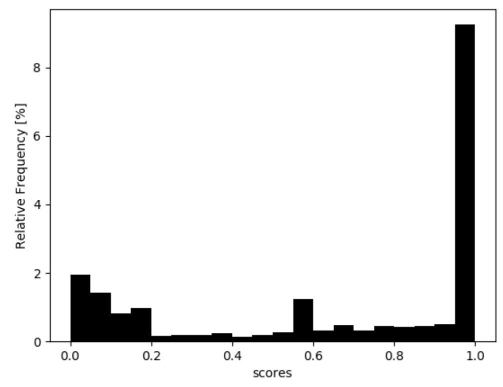

Figure 3.6: Scores' histogram of phoneme ʒ

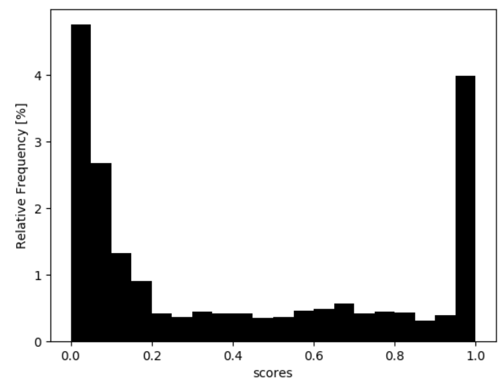

Figure 3.7: Scores' histogram of phoneme ʊ

3.3 Exercises score histogram

The ELSA system computes the score of a certain utterance using a complex formula that depends on the utterance itself and on the scores of each one of its phonemes [1]. As a consequence, the score of a sentence ranges from 0 (very bad) to 100 (perfectly pronounced).

[1]We will not describe this formula due to its confidentiality and also because for our system we just need to know which are the factors that the formula depends on.

Therefore, the histogram of the scores of a sentence is very different from the histogram of the phonemes. In the figure 3.8 we present the histogram with all the scores of the 3334 utterances.

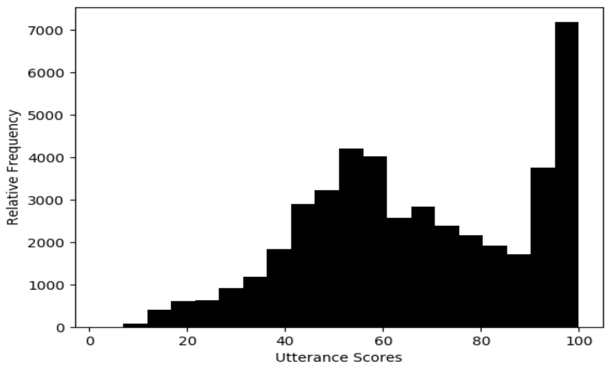

Figure 3.8: Frequency of all utterance scores

Similarly to what happens with the phonemes scores histogram, in this case difficult and easier sentences have different histograms. In the Figures 3.9, 3.10, 3.11, 3.12 we present the histograms of the exercises by order of difficulty. In other words, from Figure 3.9 (easiest) to Figure 3.12 (hardest).

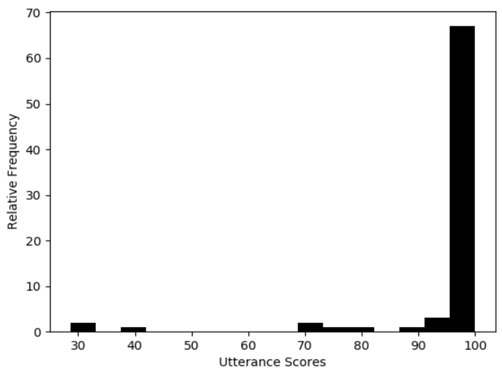

Figure 3.9: Scores' histogram of exercise with word sea

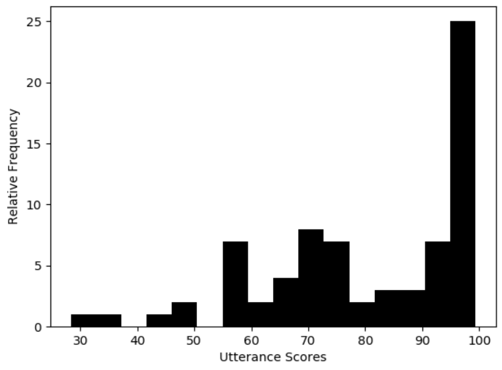

Figure 3.10: Scores' histogram of exercise with word agreement

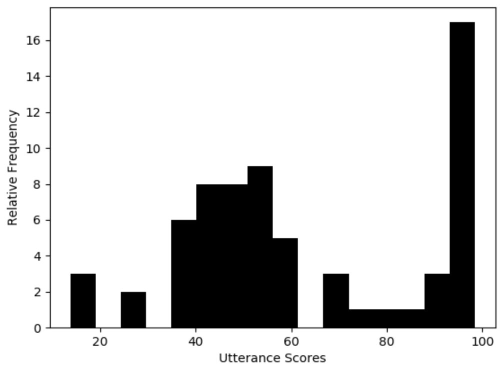

Figure 3.11: Scores' histogram of exercise with word shampoo

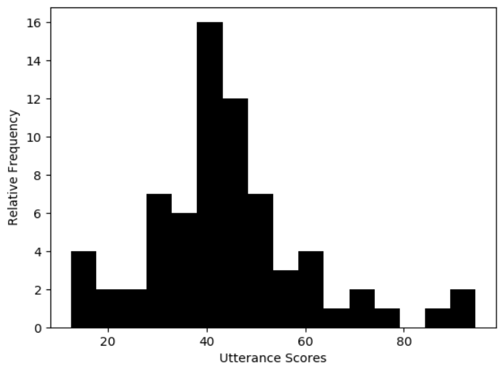

Figure 3.12: Scores' histogram of exercise with word luxury

As we can observe from the figures: 3.9, 3.10, 3.11 and 3.12, the histograms for the exercises are not as polarised as the histograms for the phonemes, but they also tend to change according to its difficulty. For example, the histogram of figure 3.12 has more sample scores close to 0 than the histogram of Figure 3.9, which means that the word luxury is more difficult to pronounce than the word sea [2].

3.4 Data set division

For reasons that will be explained in the next chapter, it was convenient to divide the corpus two times. First we selected 350 utterances and their respective scores from the corpus. This set will be used as a Test Set for the Algorithm.

The remaining data from the corpus will be divided in three sets: Train, Validation and Test Set in order to train one of the modules of the proposed system (the Neural Network). The division of the data can be observed in figure 3.13.

[2] In Chapter 4 we will explore this notion, which will help us define a Difficulty Score metric.

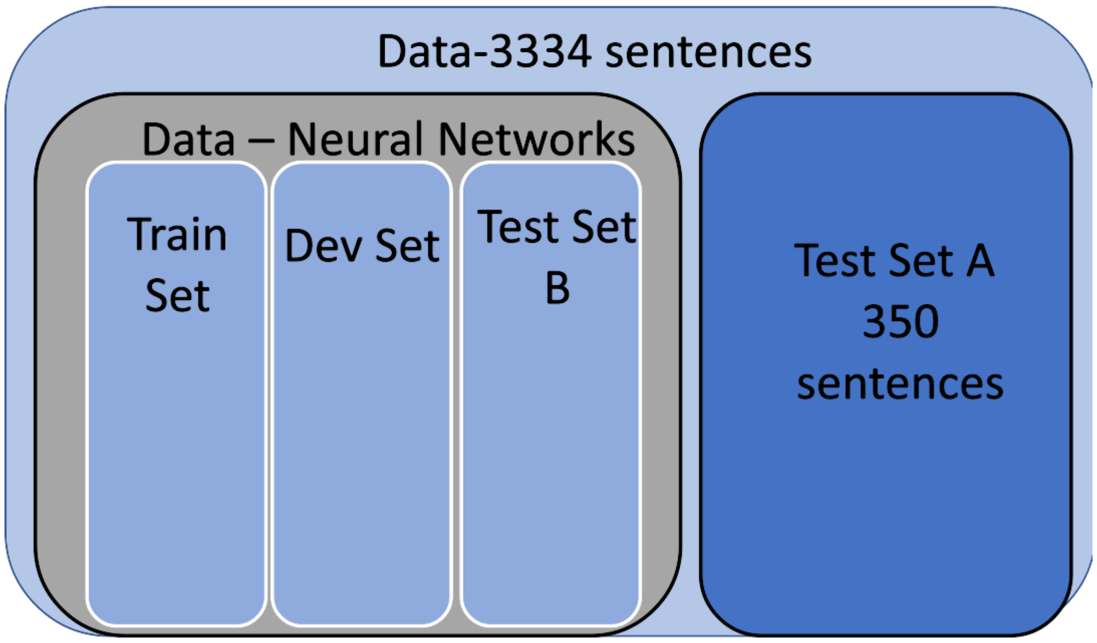

Figure 3.13: Visual representation of the dataset splitting.

4 Problem Definition and Methodology

Contents

In this chapter, we will define the problem of this thesis. We will also present the methodologies used to solve it and discuss the choices that we made.

4.1 Problem Definition

The objective is to create an algorithm that, given an input sentence, it provides a score representing the pronunciation difficulty of a Vietnamese user with a certain proficiency level. In addition, it attributes a difficulty score to each one of the sentence phonemes.

Therefore, the algorithm inputs are: a sentence (string format) and their proficiency level. The outputs are: Difficulty Score (DS) for the sentence and a DS for its phonemes.

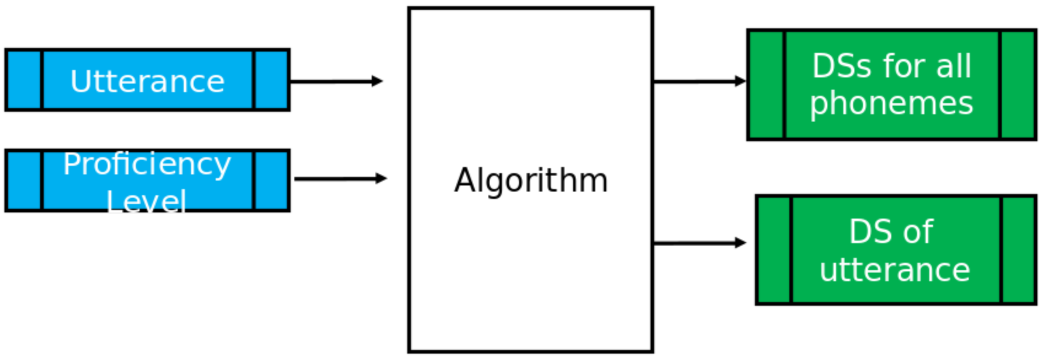

Figure 4.1: Visual representation of the proposed problem

In order to determine the DS (output scores), we use a corpus with the assessment results obtained by multiple users for each of the app's exercises. These results include multiple scores that represent the quality of their pronunciation. These scores are given for the overall utterance and its phonemes.

The main difficulty is that we do not have all the possible phonemes combinations and all the utterances in the corpus. To solve this problem, we will predict the probability distribution function (pdf) of the n-grams using an NN. Then, using the output of the NN we generate scores sample sets for the sentence and its phonemes, which will then be used to compute the outputs of the system.

4.2 Relation between Speech Server and Difficulty Scores

Before presenting the methods used in order to solve the problem, first we have to explain the difference between Speech Server Score (SSS) and DS because it is an important topic to understand the methodology presented in the next sections.

The SSSs are given to a sentence (and its phonemes) in order to evaluate the American English accent of the user. On the other hand, the DS is a score inherent to a sentence (or phoneme) that evaluates its pronunciation difficulty.

This means that a user at a given moment may obtain a certain SSS and in another moment a different one. This happens because the SSS evaluates performance, that can change over time due to factors that we cannot control such as: the mood of the user, his fatigue, problems in his voice ... The same logic can be applied for different users, which means different users will have different SSS values for the same sentence, not only because of the factors previously mentioned but also because of factors characteristic to each user, such as: his L1 language, his proficiency level ... .

It becomes clear that in order to compute the DS of a sentence (or phoneme), it is not enough to look at a single SSS, instead we have to look at multiple instances in which this phoneme/sentence was uttered and then evaluating according to those values.

Now the only unanswered question is how to distinguish between a difficult sentence and a simpler sentence. To answer this question, we can compare this problem with assessing the difficulty of a school exam. If an exam is difficult we can expect the students to obtain lower grades, but if it is easier we can expect them to obtain higher grades. So to evaluate the difficulty of an exam we could look at the students' scores pdf and then assess the exam difficulty. And the same logic can be applied on this thesis problem.

In conclusion, we can evaluate the difficulty of a phoneme/sentence (obtain DS), evaluating its SSS pdf. In the Section 4.3 we will detail the methods used to solve this problem.

4.3 Methods

As mentioned in previous chapters, when the Elsa app evaluates the pronunciation of a certain utterance, the system first computes the SSS of the phonemes, and then using those scores it computes the SSS for the sentence.

To compute the DS we will follow a similar approach: first we compute the SSS pdf of each of the utterance's phonemes, then we compute the SSS pdf of the sentence. Once we obtain the pdf for the phonemes and the sentence we compute their DS.

Figure 4.2 presents the workflow of the algorithm.

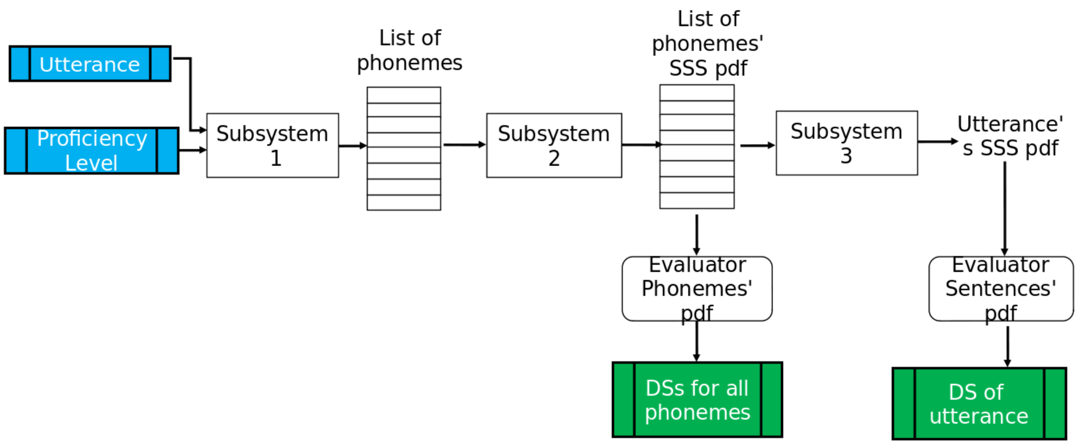

Figure 4.2: Visual representation of the algorithm and its subsystems

In the following subsections we will present how we divided the corpus and it is going to be presented in more detail how each of subsystems in Figure 4.2 work.

4.3.1 Subsystem 1 - Obtain phonemes from utterance.

The first thing the algorithm does is pre-processing the utterance: removing any punctuation and converting upper-case letters into lower-case letters.

The second step is obtaining the phonetic transcription for the sentence, using a preprepared hash table with all the transcriptions for the utterances. The resulting phonemes are presented in the 2-letter ARPABET code. In addition, the vowels code include their different levels of stress - primary, secondary or no stress and markers that indicate the position of the phoneme in the word - B -beginning, I- Intermediate, E-end, S-isolated .

Finally, we divide the utterance phonemes in groups. These groups are selected using the sliding window algorithm that divides the sentence phonemes in groups of a fixed size. In the end, there are as many groups as there are phonemes in the utterance.

Figure 4.3 presents all the steps necessary to convert the sentence into a list of phonemes groups.

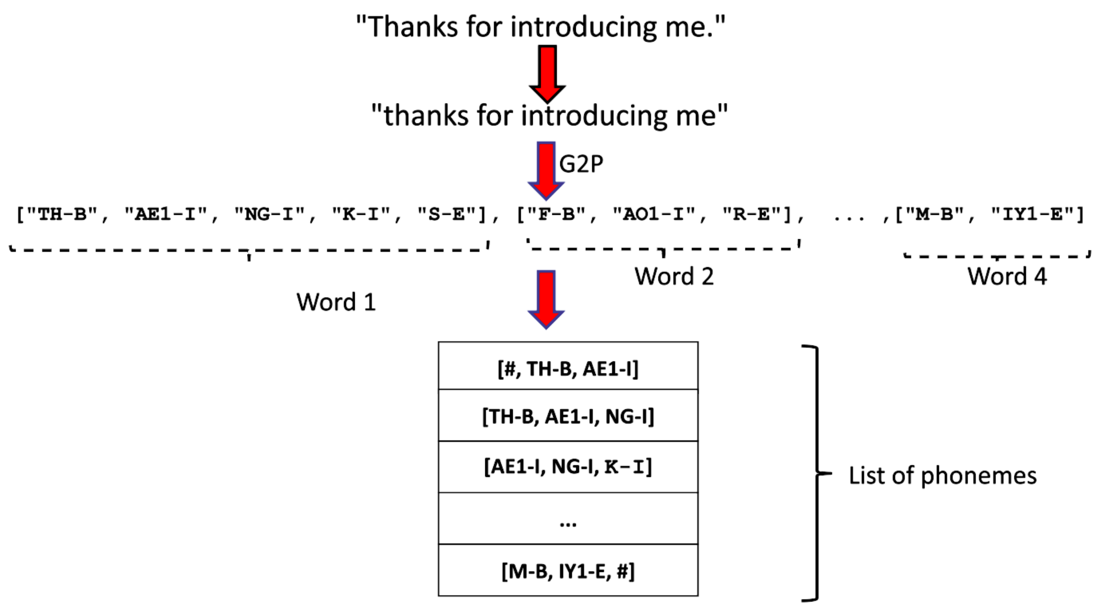
"Thanks for introducing me."
"thanks for introducing me"
G2P
["TH-B", "AE1-I", "NG-I", "K-I", "S-E"], ["F-B", "AO1-I", "R-E"], ... ,["M-B", "IY1-E"]
Word 1
Word 2
Word 4
[#, TH-B, AE1-I]
[TH-B, AE1-I, NG-I]
[AE1-I, NG-I, K-I]
...
[M-B, IY1-E, #]
List of phonemes

Figure 4.3: Pre-processing performed in Subsystem 1

As mentioned in Chapter 2, the context in which the phoneme appears is very important to evaluate its difficulty. Factors such as: the position of the phoneme in the word, the other phonemes that surround it and also if the phoneme is a vowel (and its type of stress) are important to evaluate its difficulty. Due to these factors, instead of dividing the phonemes individually, we added the other phonemes that surround the central phoneme (which is the one that we intend to evaluate) and also added other information such as: stress and the position of each phoneme in the word, as shown in the Figure 4.3.

4.3.2 Subsystem 2 - N-Grams sampling

The inputs of this subsystem are: the user proficiency level and the list of phonemes obtained in the previous subsystem. The outputs are: a list of SSS sample sets for each of the n-grams (there are as many sample sets in the output list as there are group of phonemes in the input list) .

Beforehand, it was extracted all the groups of windows and the respective SSSs of their central phonemes and they were stored in an associative array so it can be used in this subsystem. These groups of windows were separated according to the proficiency level of the user.

So, in order to obtain the SSS sample set for the phonemes, we just have to input: the window of phonemes and the proficiency level of the user in the associative array. And we will obtain a sample set that has all the SSSs in the corpus with these characteristics.

The problem with this method is that we might not have a long enough sample set, with those features. So in those cases we use a NN to solve the problem. The process of this subsystem can be observed in Figure 4.4.

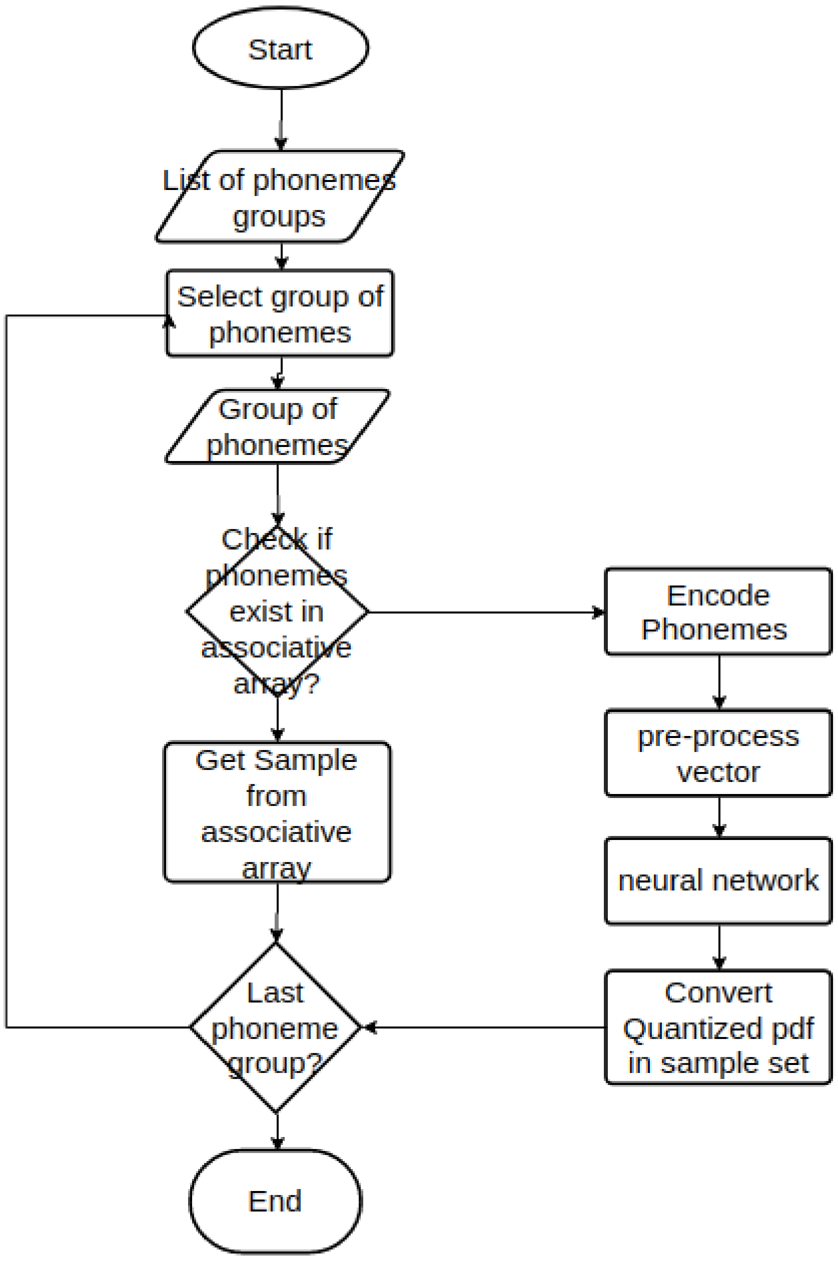

Figure 4.4: Steps performed in Subsystem 2

4.3.2.A Obtaining SSS sample set using Neural Networks

The machine learning algorithm used in order to predict the SSS pdf of the n-gram's central phoneme is the Feed Forward Neural Network [1]. The problem is a regression problem and the output is a vector of 5 elements.

[1] In the Chapter 5 we present some experiments that we did, in order to justify this choice.

The training uses as inputs (encoded): the group of phonemes and the proficiency level of the user. As output it uses: a quantized version of the SSS pdf of the input.

To encode the input of the NN, first we encode each phoneme using one-hot-encoding. Then for each phoneme we add three other features: the stress feature (0-consonant, 1- no stress, 2-primary stress, 3 -secondary stress), the word position (0-Beginning, 1- Intermediate, 2- End position, 3-Unique phoneme that constitutes word) and finally probability of error of the phoneme without the context (taken beforehand). If replacing the phoneme is the symbol "#" (that means the start or end of the utterance - check Figure 4.3) we just encode it with an array full of zeros. Then, we join all the the vectors of each phoneme and added one feature characteristic of the user: the proficiency level of the user (each group of proficiency levels has a number). This encoding process can be observed in Figure 4.5.

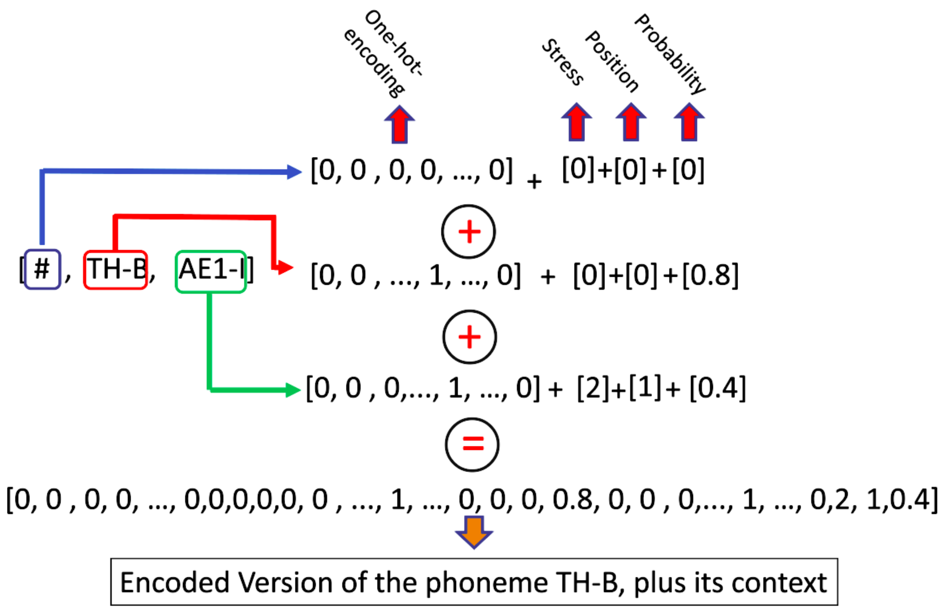

Figure 4.5: Example of the encoding of a group of phonemes

To encode the output of the NN we have to obtain the SSS sample set for the group of phonemes of a user and his proficiency level. Then, we need to obtain a quantized version of the pdf, by computing the probability that the input sample has of obtaining an SSS between certain intervals. Each of these intervals correspond to the following output neurons :

- **Output Neuron 1:** Probability of phoneme to score between : [0, 0.02).
- **Output Neuron 2:** Probability of phoneme to score between : [0.02, 0.07).
- **Output Neuron 3:** Probability of phoneme to score between : [0.07, 0.2).
- **Output Neuron 4:** Probability of phoneme to score between : [0.2, 0.995).

- **Output Neuron 5:** Probability of phoneme to score between : [0.995, 1].

We chose these intervals by observing the histograms from the phonemes SSSs sample set. After observation we concluded that, in the majority of cases, phonemes tend to have more probability in the interval [0,0.02), then the probability decreases in the intervals: [0.02,0.07), [0.07, 0.2) and [0.2, 0.995). For phonemes harder to pronounce, the interval [0.995, 1) tends to be the one that increases the most. The interval between [0.25, 0.995) tends to have smaller probability and tends to be similar to an uniform probability distribution, for these reasons we chose these intervals [2] in order to quantize the SSS pdf.

4.3.2.B Transforming Quantized pdf in a Sample Set

As previously mentioned, the NN outputs a quantized version of the SSS pdf of a phoneme in a particular context. In other words, it outputs 5 numbers that correspond to a probability of obtaining a particular SSS score in each of the 5 intervals presented in 4.3.2.A.

The quantized pdf can be seen as pdf composed of a sum of 5 uniform distributions, which their area corresponds to the probability of each of the output neurons. Then, to obtain a sample set that is representative of the quantized pdf from the NN we take the following steps:

1. Make an array of length 100 (it just has to be sufficiently large), with values from 1 to 5 that represent the output of the Neural Network.

   Each value of the array is chosen randomly in which: the probability of giving an element the value 1 is equal to the probability determined by the output neuron 1, and the same logic is applied for values 2, 3, 4 and 5.

2. Substitute the values of 1 to 5 in the array for SSS. Because the values of the auxiliary array represent an interval of SSSs (4.3.2.A).

   In order to choose the new value in the array we first check the previous value, then go to the correspondent SSS interval and finally sample a number from an uniform distribution with the same limits of that interval. The sampled number is the new value of the array.

   We do this step for the 100 elements of the array.

In the end, we obtain an SSS sample set representative of the quantized pdf of the phonemes, that we can use in the subsystem 3.

[2]In the section 5.1.1, we justify in more detail and with experiments the choice of intervals for each of the output neurons.

4.3.3 Subsystem 3 - Compute sentence's SSS pdf from the phoneme's SSS pdf

The inputs of this subsystem are: the list of phonemes groups (output of subsystem 1) and their respective SSS sample sets (output of subsystem 2). The output is an SSS sample set, that is representative of the utterance's SSS pdf.

As previously mentioned, the app first computes the SSS for the phonemes and then using a function (f_S) it computes the SSS for the utterance. Our strategy is to determine the sentence SSS pdf for the sentence using f_S and the phonemes SSS pdf.

To calculate the utterance SSS sample set we use the Monte Carlo Simulation. Where we sample an SSS for each of its phonemes, then we compute the score for the sentence, with f_S. Then, we repeat this process multiple times storing the sentences scores in an array, which then gives us the approximated pdf for the sentence (it is to be noted that if we do two different Monte Carlo Simulations for the same sentence we will get slightly different results, because the problem has a probabilistic nature so the pdf might be a little different for different simulations but the error should not be much different) .

Once we obtain the pdf for the sentences we compute the sentences difficulty by calculating the expected value of the scores in the pdf.

Algorithm 4.1: Monte Carlo Simultation - Obtain utterance SSS sample set

```
$N_{runs}, i = 200, 0$
$utt\_scores = [\,]$
for $\underline{i < N_{runs}}$ do
    $sample_{ph}, j = [\,], 0$
    for $\underline{j < len(utterance_{phonemes})}$ do
        $tmp\ sample\ from\ pdf_{phonemes}$
        $sample_{ph}.append(tmp)$
        $j = j + 1$
    $utt_{scr} = f_s(sample_{ph})$
    $utt_{scores}.append(utt_{scr})$
    $i = i + 1$
return $utt_{scores}$
```

4.3.3.A Compute DS for the phonemes

From the SSS sample set of the phonemes, we have to obtain a score that is representative of the phonemes difficulty.

In this case, first we define a threshold in the SSSs in order to separate erroneous phonemes from the correctly pronounced ones. Then, we compute the score using the formula 4.1, which represents the probability of the phoneme being perfectly pronounced by a user, with a certain proficiency level.

$$DS_p = \frac{number\ of\ correct\ scores}{total\ number\ of\ scores\ for\ the\ phoneme} \tag{4.1}$$

The interval that we considered for a phoneme to be correctly pronounced was between: [0,0.02). We chose this interval mainly because it corresponds to the first output neuron of the NN. The DS for the phonemes can be in the interval [0,1], when the difficulty score equals to 0 means that the phoneme, in that context is very hard to pronounce and the contrary when the difficulty score is equal to 1.

4.3.3.B Compute Difficulty Score for utterance

As previously discussed, the DS of the utterance should be representative of the SSS sample set of the utterance, for users with a particular proficiency level. In this case, we considered that the DS should represent the central tendency of the SSS sample set. Therefore, we use equation 4.2 in order to obtain the DS for the utterance.

$$DS_u = \frac{\sum_{n=1}^{N} x_{iu}}{N} \tag{4.2}$$

Where u represents an utterance, i represents one of the SSS results obtained by one user of the app, x_{iu} represents the value of the score obtained by that user and N the number of samples in the sample set. This DS has a range of [0;100], in which 0 means that the utterance is very difficult and 100 means the utterance is very easy.

4.3.4 Evaluation

In order to evaluate how the system predicts the DS for the phonemes, we will use the Test Set B. For all its n-grams, we will compute their DS using the formula 4.1 and then we will compare it with the DS predicted by subsystem 2.

In order to evaluate how the system predicts the DS for the utterances, we will use the Test Set A. For all its utterances, we will compute their DS using the formula 4.2 and then we will compare it with the DS predicted by the system.

The performance of the system for both outputs will be evaluated using the following metrics:

- **Mean Absolute Error:** It is a scale dependent metric and represents the average absolute value of the error.

$$MAE = \frac{\sum_{i=1}^{N} |y_i - \hat{y}_i|}{N} \tag{4.3}$$

- **R2-Score:** It is a scale independent metric. That can have negative numbers but has a maximum of 1, which represents a perfect prediction.

$$R2 - Score = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \overline{y})^2} \tag{4.4}$$

In the equations 4.3 and 4.4 y_i represents the real DS, $\hat{y}_i$ the DS predicted by the system and $\overline{y}$ the mean of the real values.

It is to be noted, however, that when we evaluate the NN output (which forecasts the pdf for the phonemes), we present the values of 4.3 and 4.4 for each of the elements of the output vector and also an average of these values which represents the overall performance of the system.

5 Experiments and Results

Contents

In order to tune the model we conducted several experiments. In this chapter, we will present some of these experiments and discuss the results obtained for each of them and what they represent for our model. In addition, we will present and discuss the final results obtained.

5.1 Experiments

The method presented in Chapter 4 has multiple parameters that need to be tuned in order to optimise the model. In this section, we will discuss some experiments that we performed in order to tune these parameters and obtain an optimised model.

5.1.1 Phoneme's SSS pdf quantization

As previously mentioned, the algorithm has two ways of obtaining a sample set representative of an n-gram: in the case we have the n-gram in the associative array, we use the sample set previously stored and in the case we do not have the n-gram, we use an NN to predict the pdf.

The pdf that the NN forecasts is quantized in several intervals. In this experiment, to find the appropriate intervals for our problem, we changed how the algorithm generates the n-gram SSS sample set. In this case, we obtained the sample set directly from the corpus [1] and then we computed the quantized pdf according to the established intervals (in order to simulate the output of the NN). Then, we generated the sample set for the phoneme, using the same method discussed in Chapter 4.

In order to find the most suitable intervals for the phonemes' SSS pdf, we used this version of the algorithm to calculate the DS for all the utterances in the corpus. We used different intervals for the quantization and we compared it to the "control" case, where we did not apply quantization [2].

In the first test, we used the "control case" and intervals with the same width in order to represent the pdf like in the NN output. In the Table 5.1, we present the results obtained for the different intervals and according to the size of the n-gram.

[1] In this case we did not follow the division of the data presented in the previous chapter, because it was not necessary for this test. Therefore, when we obtain the sample set for an n-gram, it corresponds to the scores of the n-gram in all the corpus.

[2] We extracted directly from the corpus without doing the other two steps described in the previous paragraph.

	1 phonemes		3 phonemes		5 phonemes		7 phonemes	
	MAE	R2-Score	MAE	R2-Score	MAE	R2-Score	MAE	R2-Score
Control Case	5.32	-0.32	3.1	0.75	2.21	0.87	1.92	0.89
40 intervals	5.44	-0.32	3.25	0.71	2.31	0.86	2.05	0.88
20 intervals	5.35	-0.41	3.25	0.71	2.44	0.85	2.05	0.88
10 intervals	5.97	-0.75	4.07	0.55	3.18	0.76	2.96	0.78
5 intervals	9.22	-4.02	8.35	-1.04	7.87	-0.48	7.98	-0.42

Table 5.1: Results obtained for the first test, according to the number of intervals and the number of phonemes in the n-gram.

From the Table 5.1 we can observe that increasing the number of intervals and the number of phonemes in the n-gram has a positive effect on the results. Despite this fact, it is better for our model to have a reduced number of intervals, because it represents a smaller number of output neurons in the Neural Network, which makes it easier to predict. On the other hand, it is important to have a quantization that is adequate to use as input in the Monte Carlo Simulation.

From Chapter 3, we can observe that the phonemes pdf follow a certain pattern. More specifically, we can observe that there is more probability of obtaining a score close to 0 or 1 than an intermediate score. We can also note that the phonemes SSS pdf has a tendency of decreasing between the intervals 0 and 0.2. Therefore, instead of using intervals with the same length we can choose them in order to accommodate this information. In the Table 5.2, we present some of the intervals tested in order to reduce the number of output neurons necessary in the NN.

In this table, we present the results according to the limits established for the intervals. It is to be noted, that for Table 5.2 we specified the limits in the first column. For example, in the third line of the table we specify three intervals for the quantized pdf: [0,0.02), [0.02,0.995) and [0.995,1].

	Number of	3 phonemes		5 phonemes		7 phonemes	
	intervals	MAE	R2	MAE	R2	MAE	R2
Control Case	——	3.1	0.75	2.21	0.87	1.92	0.89
(0;0.2;0.995;1)	3	6.24	-0.05	5.61	0.33	5.4	0.38
(0;0.02;0.2;0.995;1)	4	3.94	0.57	3.19	0.75	2.89	0.79
(0;0.02;0.07;0.2;0.995;1)	**5**	**3.24**	**0.7**	**2.46**	**0.84**	**2.24**	**0.86**
(0;0.02;0.07;0.2;0,6;0.995;1)	6	3.37	0.7	2.48	0.84	2.24	0.86
(0;0.02;0,07;0.2;0.6;0.84;0.995;1)	7	3.37	0.7	2.42	0.85	2.15	0.87

Table 5.2: Results obtained for the second test, according to the intervals and the number of phonemes in the n-gram.

From the data presented in Table 5.2, we can infer that we should train the NN with five output neurons and they should represent the following intervals: [0, 0.02), [0.02, 0.07), [0.07, 0.2), [0.2, 0.995) and [0.995,1].

5.1.2 Neural Network Architectures

In this chapter we will present some of the types of NN architectures that we experimented and the results obtained with their optimised versions.

In subsystem 2, we used a Neural Network in order to forecast the SSS pdf of the phonemes. In order to find the best NN for our model, we tested several types of architectures. These architectures had in common some aspects such as: they received as input the encoded phonemes plus the proficiency level of the user and they had the same output layer. This layer had five neurons and a softmax activation function.

The first architecture we tested was a Feed Forward Neural Network. We chose to experiment using this architecture, because even though they are one of the simplest architectures, they still achieve a lot of success in many machine learning tasks [17] [18]. Using this NN we followed the same type of encoding presented in Chapter 4.

The second architecture was a Recurrent Neural Network (RNN) with an LSTM layer as the input layer (Figure 5.1). Since our input data is sequential in nature we also experimented using an RNN, which is usually used in these types of tasks [19] [20] [21]. In this architecture, we used as time-steps each of the n-gram's phonemes.

The third NN was an RNN-LSTM with an attention mechanism (Figure 5.2). NNs with attention mechanisms are relatively new but already very successful in many areas, including Natural Language Processing [22] [23] [24]. Furthermore, a similar approach was used in Vec2Read [1], that we presented in Chapter 2. In our approach, however, we did not used an hierarchical NN, because we were using phonemes as inputs, instead of a full sentence, which can be hierarchically divided. So our attention mechanism focused on which phonemes the model should pay attention in order to predict the pdf.

In the Table 5.3, we present the results obtained for the three architectures, previously mentioned [3].

[3] In the table we present the result architecture after tuning the NN parameters in order to obtain the best results.

	3 phonemes		5 phonemes		7 phonemes	
	MAE	R2-Score	MAE	R2-Score	MAE	R2-Score
Feed Forward Neural Network	**0.06**	**0.61**	**0.068**	**0.54**	**0.076**	**0.47**
RNN	0.065	0.52	0.071	0.47	0.083	0.36
RNN + attention mechanism	0.067	0.51	0.076	0.42	0.087	0.33

Table 5.3: Results obtained for each of the optimised NN architectures, according to the size of the n-gram. Tested in the data set: Test Set B.

After analysing the results presented in Table 5.3, we can observe that the Feed Forward Neural Network has a better performance than the other architectures.

Other conclusion that we can derive from the Table 5.3 is that the NN predicts better the pdf for smaller n-grams.

Although, the network obtains better results for these cases, we can not select the size of the n-gram just based on these results. As we mentioned in section 5.1.1, for subsystem 2 is better to have larger n-grams, so to optimise the system we have to find a compromise between the subsystems 2 and 3.

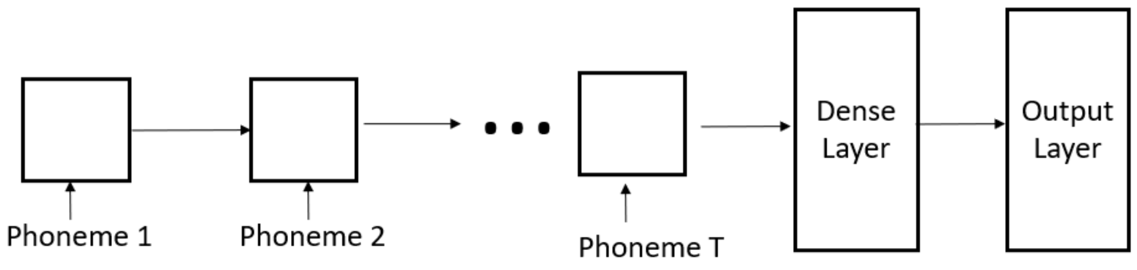

Figure 5.1: Architecture of the NN with an LSTM as input layer.

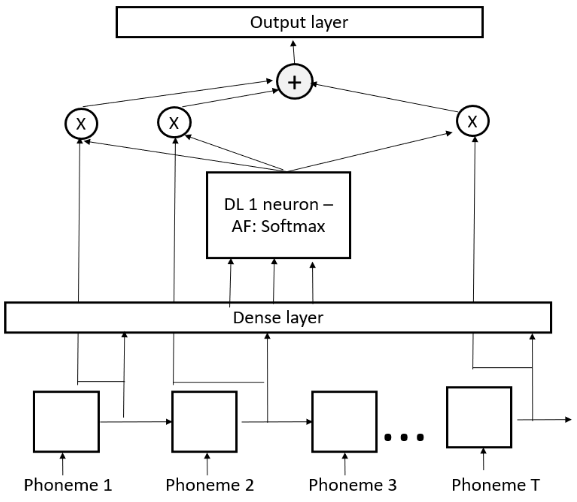

Figure 5.2: Architecture of the NN with an LSTM as input layer and attention mechanisms.

5.1.3 Evaluator of the utterance pdf

The ”evaluator of the utterance SSS sample set” (Figure 4.2), should obtain the score for the sentence that is representative of the generated SSS sample set. Therefore, in order to select an appropriate evaluator, we computed the DS for the utterances in the Test Set B (Figure 3.13) with different evaluators. In the table 5.4 we present the results obtained, with different evaluators.

	Description	Formula	MAE	R2-Score
Evaluator 1	Expected Value	$\frac{1}{N}\sum_{n=1}^{N}(x_{iu})$	6.50	0.53
Evaluator 2	Median	——-	8.34	0.51
Evaluator 3	Probability of correct pronunciation	$\frac{number\ of\ correct\ utterances}{number\ of\ utterances}$	10.26	0.64
Evaluator 4	Mean of the square	$\frac{1}{N*F_4}\sum_{n=1}^{N}(x_{iu})^2$	8.3	0.58
Evaluator 5	Mean of the cube	$\frac{1}{N*F_5}\sum_{n=1}^{N}(x_{iu})^3$	8.73	0.6

Table 5.4: Results obtained using different evaluators.

If we take into account the R2-Score we find that the best evaluators would be the number 2 and 5. However, in this particular case, the main metric that we use to evaluate which one we should use is the Mean Absolute Error. Using this metric we infer that the evaluator that gives us the most precise results is the Expected Value (Equation 4.2).

5.2 Final Results

The system that we developed has two outputs: the difficulty score for the utterance and the difficulty score associated to its phonemes. Therefore, in order to evaluate the system, we have to evaluate both these outputs. In this section, we will present the tests that we performed to evaluate how the optimised system performs for both of these outputs.

5.2.1 Phoneme Difficulty

As mentioned in Chapter 4, the DS for the phonemes is calculated using the SSS sample set obtained from Subsystem 2 and applying the equation 4.1.

Due to the fact that the error from this subsystem comes from the Neural Network. In this section we will evaluate the NN. More specifically, how it is able to forecast the pdf for unseen data. For that reason, we will use the Test Set B.

In this test we will use the NN to predict the quantized SSS pdf that corresponds to an n-gram and proficiency level and then we will obtain the real quantized pdf that corresponds to that n-gram and proficiency level from the Test Set B. We perform this experiment in every single n-gram in Test Set B and in the end we compare the predicted and real values with the evaluation method, described in section 4.3.4. In the Table 5.5, we present the results obtained from this test.

Output Neuron Number	3 Phonemes		5 Phonemes		7 Phonemes	
	MAE	R2-Score	MAE	R2-Score	MAE	R2-Score
1 - [0, 0.02]	**0.077**	**0.81**	0.091	0.73	0.105	0.68
2 - [0.02, 0.07]	0.068	0.4	0.077	0.38	0.083	0.3
3 - [0.07, 0.2]	0.054	0.51	0.06	0.46	0.063	0.38
4 - [0.2, 0.995]	0.057	0.66	0.063	0.56	0.076	0.42
5 - [0.995, 1]	0.042	0.64	0.047	0.55	0.05	0.55

Table 5.5: Results obtained for the train, dev and test set of the Neural Network

In the Table 5.5, we highlighted how the NN determined the Difficulty Score for the phonemes. As we can observe, the model is able to attribute very competently a difficulty score for the phonemes. However, it is not as efficient in forecasting the values of the other output neurons.

5.2.2 Utterance Difficulty

In order to evaluate the overall System, we use the Test Set A, which we have not used so far whether to train the NN, or to store data in the associative array. Therefore, this can be seen as new data for the system and we can use it to evaluate how the model performs with unseen data.

To evaluate the model, we use our system to generate the SSS sample sets that correspond to the utterance and proficiency level of every single sample in the Test Set A. Then, we extract the real sample sets for each of the utterance (and corresponding proficiency level) present in the Test Set A. Using the equation 4.2 we compute the DS associated with the generated and real sample set. Finally, we evaluate the model using the evaluation method described in the section 4.3.4.

We performed this test with a modified version of the algorithm (first line of Table 5.6) and with the regular version of the system (second line of Table 5.6). In the modified version, we just used the NN to compute the DS for the utterance, not using the associative array (Figure 4.4). In the Table 5.6, we present the results obtained from this test.

Output Neuron Number	3 phonemes		5 phonemes		7 phonemes	
	MAE	R2-Score	MAE	R2-Score	MAE	R2-Score
Without the associative array	6.35	0.55	6.34	0.55	6.90	0.42
With the associative array	**6.3**	**0.55**	6.35	0.56	6.7	0.43

Table 5.6: Results obtained for overall system, according to the size of the n-gram

After analysing the results presented in Table 5.6, we can observe that using the associative array does not modify significantly performance of the system. Although, this is accurate for this particular Test Set A (because the utterances in this set do not have a lot of n-grams in common with the ones in the associative array), for an utterance with a lot of n-grams in common with the associative array this is no longer truthful. In these cases, the associative array improves the performance of the system, as demonstrated in the section 5.1.1.

Secondly, we can observe that for this particular output, the size of the n-gram does not significantly change the performance of the system. But, as we observed in section 5.2.1, for the phonemes' DS having an n-gram with three phonemes is preferable.

Additionally, we can compare the results obtained for the phonemes difficulty score and for the utterance. Observing the highlighted fields [4] in the Tables 5.5 and 5.6, we can infer that, although, the system is competently predicting the difficulty scores for the sentence, it predicts more accurately the phonemes DS than the utterance DS.

[4]The highlighted fields correspond to the optimised parameters in the model.

6 Conclusion

Contents

6.1 Conclusions

In this project, our objective was to develop a system capable of automatically assessing the pronunciation difficulty of a certain input text.

Therefore, we developed a system that takes as input a certain text and the proficiency level of a user. Then, it outputs a score representative of the difficulty that a user with that particular proficiency level would have pronouncing the utterance. Additionally, the system evaluates the difficulty score for each of the sentence's phonemes.

In order to train the model, we had a database with several utterances and the evaluation obtained by multiple users with different proficiency levels.

In order to develop the system we tried to forecast the pdf for the phonemes. This pdf was obtained using a Neural Network, that we trained using ELSA's dataset. To generate a sample set representative of the utterance SSS pdf we used a Monte Carlo Simulation. This simulation had as input each of the phonemes pdf and used ELSA's formula to compute a sample set that is representative of the utterance pdf.

Finally, after obtaining the sample set for the text we used the formulas described in Chapter 4 to compute the Difficulty Scores for the sentence and its phonemes.

Since, the system has two outputs we had to test how the system performs for both of them. For each of these outputs we used two different subdivisions of the data sets.

To test how it evaluates the DS for the phonemes we compute the scores for a set of phonemes, using the NN. Then we compared the score obtained to the real one. For this output we can observe that the system is able to forecast the phoneme score according to its context and the user proficiency level.

To test how the system evaluates the DS for the utterances we compute the scores for a set of utterances. Then we compared the score obtained to the real one. For the sentence output we can observe that the system is capable of differentiating between difficult and easy utterances, but is more accurate predicting phonemes DS than utterances DS.

6.2 System Limitations and Future Work

In this section, we will discuss some of the system limitations and how we could improve it in Future Work.

As previously mentioned, the system evaluates more accurately the difficulty for the utterance's phonemes than for the utterance itself. We could improve this aspect of our model, if we improve the performance of the NN, because the results from the network will influence the result for the phonemes DS and for the utterance DS.

Although the NN efficiently predicts the value correspondent to the output neuron 1, for the rest of the output neurons the NN is not as efficient, which will affect the evaluation of the sentence DS. If we observe the histograms of the phoneme's scores in Chapter 3, we verify that there are substantially more samples scores around the zero value than in any other interval. This causes that some of the score intervals have a smaller probability and a simple deviation in the data, produces larger variations in the ground truth, which affects the training of the NN and its predictive capabilities.

In the future, to improve the efficiency of the system the main objective should be to increase the capacity of the NN to forecast the pdf for n-grams of larger size. This could be achieved using better encoding techniques capable of better incorporating external features to the input data. Once we obtain an NN more capable of obtaining a pdf for larger n-grams, the model would be able to better assess a DS for the sentence.

Other aspect that we could improve is the way we evaluate the utterance SSS sample set in order to obtain the respective DS. We could change the formula that we use to obtain the utterance DS to a more complex formula that better reflects the difficulty score.

Furthermore, in the future we could change the algorithm so it can evaluate the difficulty score of a sentence not only for Vietnamese students, but also for students with other native languages.

Bibliography

[1] Ion Madrazo Azpiazu and Maria Soledad Pera. Multiattentive recurrent neural network architecture for multilingual readability assessment. *Transactions of the Association for Computational Linguistics*, 7:421–436, 2019.

[2] Deborah Hwa-Froelich, Barbara W Hodson, and Harold T Edwards. Characteristics of vietnamese phonology. *American Journal of Speech-Language Pathology*, 2002.

[3] *The American heritage dictionary (4th ed.)*. Boston: Houghton Mifflin Company, 2000.

[4] Nahed Rajaâ Ghlamallah. British and american phonology: Variation or contrast.

[5] Soenjono Dardjowidjojo. *English phonetics & phonology for Indonesians*. Yayasan Pustaka Obor Indonesia, 2009.

[6] Peter Roach. *English phonetics and phonology paperback with audio CDs (2): A practical course*. Cambridge university press, 2009.

[7] Giang Tang. Cross-linguistic analysis of vietnamese and english with implications for vietnamese language acquisition and maintenance in the united states. *Journal of Southeast Asian American Education and Advancement*, 2(1):3, 2007.

[8] Rita Wong. Teaching pronunciation: Focus on english rhythm and intonation. language in education: Theory and practice, no. 68. 1987.

[9] Luu Thi Kim Nhung. A brief comparison of vietnamese intonation and english intonation and its implications for teaching english intonation to vietnamese efl learners. *VNU Journal of Foreign Studies*, 26(3), 2010.

[10] Thomas Stainsby and Ian Cross. The perception of pitch. *The Oxford handbook of music psychology*, pages 47–58, 2009.

[11] Jeesoo Bang and Gary Geunbae Lee. Determining sentence pronunciation difficulty for non-native speakers. In *Speech and Language Technology in Education*, 2013.

[12] Scott A Crossley, David B Allen, and Danielle S McNamara. Text readability and intuitive simplification: A comparison of readability formulas. *Reading in a foreign language*, 23(1):84–101, 2011.

[13] Scott A Crossley, Jerry Greenfield, and Danielle S McNamara. Assessing text readability using cognitively based indices. *Tesol Quarterly*, 42(3):475–493, 2008.

[14] J Peter Kincaid, Robert P Fishburne Jr, Richard L Rogers, and Brad S Chissom. Derivation of new readability formulas (automated readability index, fog count and flesch reading ease formula) for navy enlisted personnel. Technical report, Naval Technical Training Command Millington TN Research Branch, 1975.

[15] Orphée De Clercq and Véronique Hoste. All mixed up? finding the optimal feature set for general readability prediction and its application to english and dutch. *Computational Linguistics*, 42(3):457–490, 2016.

[16] Kevyn Collins-Thompson and James P Callan. A language modeling approach to predicting reading difficulty. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: HLT-NAACL 2004*, pages 193–200, 2004.

[17] Jan A Botha, Emily Pitler, Ji Ma, Anton Bakalov, Alex Salcianu, David Weiss, Ryan McDonald, and Slav Petrov. Natural language processing with small feed-forward networks. *arXiv preprint arXiv:1708.00214*, 2017.

[18] Yoav Goldberg. Neural network methods for natural language processing. *Synthesis Lectures on Human Language Technologies*, 10(1):1–309, 2017.

[19] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[20] Dani Yogatama, Chris Dyer, Wang Ling, and Phil Blunsom. Generative and discriminative text classification with recurrent neural networks. *arXiv preprint arXiv:1703.01898*, 2017.

[21] Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. Recent trends in deep learning based natural language processing. *ieee Computational intelligenCe magazine*, 13(3):55–75, 2018.

[22] Ru Peng, Zhitao Chen, Tianyong Hao, and Yi Fang. Neural machine translation with attention based on a new syntactic branch distance. In *China Conference on Machine Translation*, pages 47–57. Springer, 2019.

[23] Yuxuan Sun, Keying Chen, Lin Sun, and Chenlu Hu. Attention-based deep learning model for text readability evaluation. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2020.

[24] Changshun Du and Lei Huang. Text classification research with attention-based recurrent neural networks. *International Journal of Computers Communications & Control*, 13(1):50–61, 2018.