PARLA: mobile application for English pronunciation
A supervised machine learning approach

Davide Berdin
{davide.berdin.0110}@student.uu.se

Master Thesis
Department of Information Technology

January 18, 2016

To my family that never stopped believing in me

Abstract

Acknowledgements

Contents

List of Figures

Chapter 1

Introduction

The pronunciation is one of the hardest part of learning a language among all the other components, such as grammar rules and vocabulary. To achieve a good level of pronunciation, non native speakers have to study and constantly practice the target language for a incredible amount of hours. In most cases, when students are learning a new language, the teacher is not a native speaker in which implies that the pronunciation may be influenced by the country where he or she comes from, since it is a normal consequence of second learning language [10]. In fact, [11] states that the advantages of having a native speaker as a teacher, lies in the superior linguistic competences, especially the usage of the language more spontaneously in different communication situations. Pronunciation falls into those competences underlying a base problem in teaching pronunciation at school.

The basic questions we tried to answer in this work are:
1) Why is pronunciation so important?
2) What are the most effective methods for improving the pronunciation?
3) What is the research state-of-art and what can we do to make it better?

The first question is fairly easy to answer. There are two reasons to claim why pronunciation is important: *(i)* it helps to acquire the target language faster and *(ii)* being understood. Regarding the first point, earlier a learner masters the basics of pronunciation, the faster it will become fluent. The reason is because *critical listening* with a particular focus on hearing the sounds will lead to gain fluency in speaking the language. The second point is **crucial** when working with other people, especially at these days where both in school and business the environment is often multicultural. Pronunciation mistakes may lead the person to being misunderstood affecting the results of a project for example.

With these statements in mind, [12] gives suggestions on how a learner can effectively improve the pronunciation. Four important ways are depicted: *Conversation* is the most relevant approach to improve pronunciation, although, a supervision of an *expert guidance* that corrects the mistakes is fundamental during the process of learning. At the same time, learners have to be pro-active to have conversation with other native speakers in such a way to constantly practicing. *Repetitions* of pronunciation exercises is another important factor that will help the learner to be better in speaking. As last, *Critical listening*, that we also mentioned above, amplify the opportunity of learning the way native speakers pronounce words. In particular, for a learner is important to understand the difference when he or she is pronouncing a certain sentence with the one said by the native speaker. This method is very effective and is important for understanding the different sounds of the language and how a native speaker is able to reproduce them [13].
An important factor while learning a second language is to have a feedback about improvements. Teachers are usually responsible to judge the way the learners' progress. In fact, when teaching pronunciation, often it is used to draw the intonation and the stress of the words in such a way that the learner is able to see how the utterances should be pronounced. The *British Council* shows this practice [14]. The usage of visual feedbacks is the key of learning pronunciation and it is the main feature of this research.

In the computer science field, some works have been previously done regarding pronunciation. For instance, [15] helps learners to acquire the tonal sound system of Mandarin Chinese through a mobile game. Another example is [16] in which the application provides a platform where learners of Chinese language can interact with native speakers and challenging them to a competition of pronunciations of Chinese tones.

The idea behind this project is based on the fact that people need to keep practicing their pronunciation to have a significant improvement as well as they need immediate feedbacks to understand if they are going in the right direction or not. The approach we used is based on these two factors and we designed the system to be as useful and portable as possible. The mobile application is where the user will test the pronunciation, a server through a machine learning technique will compute the similarity between the user's pronunciation and the native speaker's one and the results will be displayed on the phone.

We started collecting data from *American Native Speakers* in which we asked to pronounced a set of most used idioms and slangs. Each candidate had to repeat the same sentence several times trying to be as consistent as possible. After we gathered the data, a preprocessing step is due since we are seeking for specific features such as voice-stress, accent, intonation and formants. This part has been done using an external tool called **FAVE-Extract** in which it uses **PRAAT**[17] to analyze the sound. At this point, the next step is processed differently when treating native speaker files because we use manually defined the correct *phonemes* for each sentence. This step is called **force alignment** in which it is estimated the beginning and the end of when a phoneme is pronounced by the speaker. For non-native speakers we used the phonemes that we extracted using the speech recognition system.

The machine learning part is divided in two: the first consists in using the library called **CMU Sphinx 4** with an acoustic model trained with all the data we collected from the native speakers. This library is a **HMM-based** system with multiple searching systems written in Java. To estimate the overall error between the native pronunciation and the user, we use a method called **Word Error Rate** (WER), a common method metric for measuring the performance of a speech recognition system. The second part consists in using a **Gaussian Mixture Model** (GMM) that we used to predict the vowels pronounced by the user. The result should help the user to better understand *how close* is his/her vowels pronunciation compared with the native ones.

After the server has computed the speech recognition extracting the phonemes and predicted the similarity of vowels, the system creates graphs that will be used in the mobile application as feedback. In this way the user has a clear understanding of how he/she should **adjust** the way the utterance have to be pronounced.

Chapter 2

Sounds of the General American English

In the *General American English* there are 41 different sounds in which can be structured by the way they are produced. In 2.1 is shown the kind of sounds with the respective number of possible productions. Each type will be described into a dedicated section of this thesis. An important factor is the way of how the *constriction* of the flow of air is made. In fact, to distinguish between *consonants*, *semivowels* and *vowels*, the *degree* of constriction is checked. Instead, for *sonorant* consonants the air flow is continuous with no pressure. *Nasal* consonants have an occlusive consonant made with a lowered velum allowing the airflow in the nasal cavity. The *continuant* consonants are produced without blocking the airflow in the oral cavity.

Type	**Number**
Vowels	18
Fricatives	8
Stops	6
Nasals	3
Semivowels	4
Affricates	2
Aspirant	1

Table 2.1: Type of English sounds

2.1 Vowel production

Generally speaking, when a vowel is pronounced, there is no air-constriction in the flow. This means that the articulators like the tongue, lips and the uvula do not touch allowing the flow of air from the lungs. The consonants instead have another pattern when producing them. Moreover, to produce each vowel, the mouth has to make a different shape in such a way that the resonance is different. 2.1 shows the way the mouth, the jaw and the lips are combined in a such a way to produce the acoustic sound of a vowel.

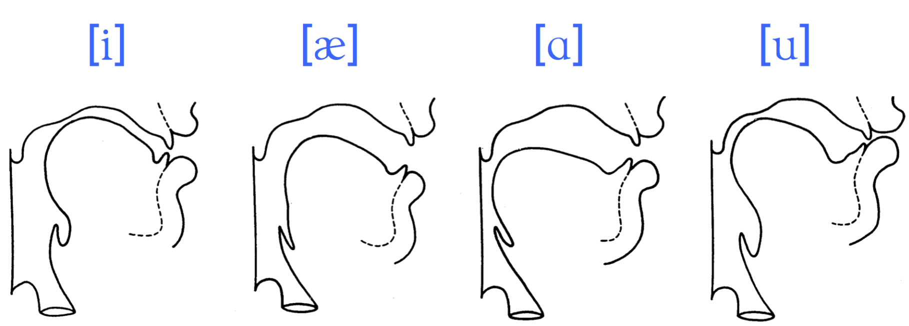

Figure 2.1: Vowels production [1]

2.1.1 Vowel of American English

There are 18 different vowels in American English that can be grouped by three different sets: the **monopthongs**, the **diphthongs**, and the **schwa's** - or reduced vowels.

/iʸ/	iy	beat	/ɔ/	ao	bought	/ɑʸ/	ay	bite
/ɪ/	ih	bit	/ʌ/	ah	but	/ɔʸ/	oy	Boyd
/eʸ/	ey	bait	/oʷ/	ow	boat	/ɑʷ/	aw	bout
/ɛ/	eh	bet	/ʊ/	uh	book	[ə]	ax	about
/æ/	ae	bat	/u/	uw	boot	[ɨ]	ix	roses
/ɑ/	aa	Bob	/ɝ/	er	Bert	[ɚ]	axr	butter

Figure 2.2: Example of words depending on the group [1]

The first column shows some examples of monopthongs. A *monopthong* is a clear vowel sound in which the utterance are fixed at both the beginning and at the end. The central part of the picture represents the diphthongs. A *diphthong* is the sound produced by two vowels when they occur within the same syllable. In the last column are depicted some examples of reduced vowels. *Schwa's* refers to the vowel sound that stays in the mid-central of the word. In general, in English, the schwa is found in unstressed position.

2.1.2 Formants

A *formant* is the resonant frequency of a vocal track that resonate the loudest. In a spectrum graph, formants are represented by the peaks. In 2.3 it is possible to see how the three first formants are defined by the peaks. The pictures is the *envelope* of a spectrogram of the vowel **[i]**. Frequencies are the most relevant information to determine which vowel has been pronounced. In general, within a spectrum graph there may be a different number of formants, although the most relevant are the first three and they are named **F1**, **F2** and **F3**.

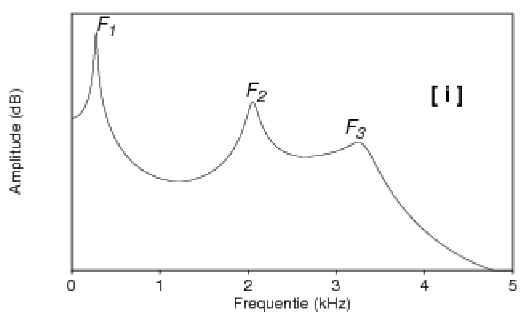

Figure 2.3: Spectral envelope of the [i] vowel pronunciation. F1, F2 and F3 are the first 3 formants [2]

The frequencies produced by the formants are highly dependent on the tongue position. In fact, formant *F1*'s frequencies are produced when the tongue is either in a *high* or *low* position, whereas formant *F2* when the tongue is in either *front* or *back* position and formant *F3* when the tongue is doing *Retroflexion*. **Retroflection** is more present when pronouncing the consonant *R*

2.1.3 Vowel duration

The duration of a vowel is the time that taken when pronouncing it. The duration is measured in *centiseconds* and in English[1] the different lengths are defined by certain rules. In general, the length of *lax vowels* such as /ɪ e æʌ ɒ u ɘ/ are short whereas *tense vowels* like /iː ɑː ɔː uː ɜː/ including diphthongs /eɪ aɪ ɔɪ əʊ aʊ ɪə ea ʊə/ have a variable length but longer than lax vowels [3]. In 2.4 is shown an example of time-length of some vowels. In General American English, the length of vowels are not as distinctive as in the *RP*[2] pronunciation. In some American accents, to express an emphasis the length of vowels can be extended.

[1] In Icelandic as well

[2] More commonly referred as the Standard English in the UK

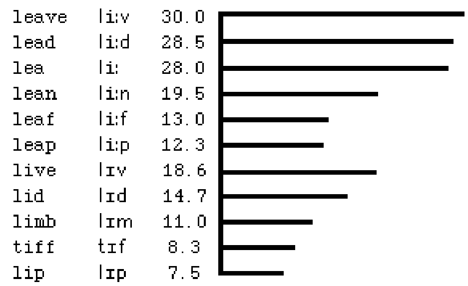

Figure 2.4: RP vowel length [3]

2.2 Fricative Production

A **fricative** is a consonant sound that is produced by narrowing the cavity causing a friction as the air goes through it [18]. There are eight fricatives in American English divided in two categories: *Unvoiced* and *Voiced*. These two categories are often called *Non-Strident* and *Strident* that means that there is a constriction behind the alveolar ridge.

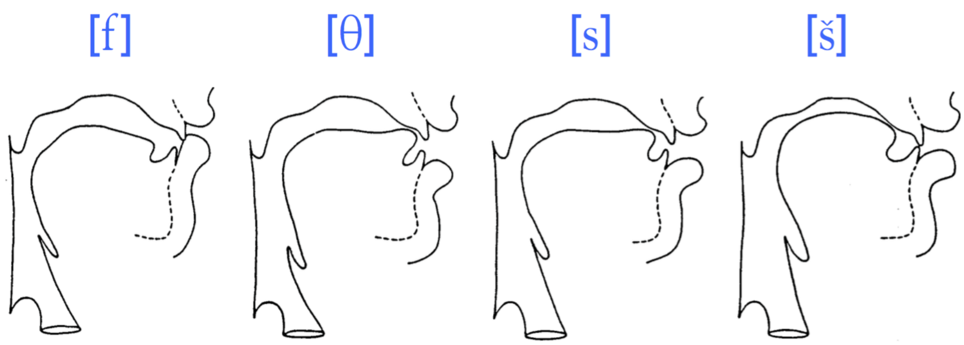

Figure 2.5: Fricative production [1]

In 2.6 it is possible to see some examples of these two categories. Each consonant also belongs to a specific articulation position. In fact, each figure in 2.5 represents a specific articulation position. From left to right we have: *Labio-Dental* (Labial), *Interdental* (Dental), *Alveolar* and *Palato-Alveolar* (Palatal).

Type	Unvoiced			Voiced		
Labial	/f/	f	fee	/v/	v	v
Dental	/θ/	th	thief	/ð/	dh	thee
Alveolar	/s/	s	see	/z/	z	z
Palatal	/š/	sh	she	/ž/	zh	Gigi

Figure 2.6: Fricative examples of productions [1]

2.3 Affricate Production

An **affricate** consonant is produced by stopping the airflow first and then release it similar to a fricative. The result is also considered a *turbulence noise* since the produced sound has a sudden release of the constriction. In English there only two affricate phonemes, as depicted in 2.7.

Voiced			Unvoiced		
/ǰ/	jh	judge	/č/	ch	church

Figure 2.7: Affricative production [1]

2.4 Aspirant Production

An **aspirant** consonant is a strong outbreak of breath produced by generating a turbulent airflow at glottis level. In American English exists only one aspirant consonant and it is the /h/, for instance in the word *hat*.

2.5 Stop Production

A **Stop** is a consonant sound formed by stopping the airflow in the oral cavity. The stop consonant is also known as *plosive* which means that when the air is released it creates a small *explosive* sound [19]. The occlusion can come up in three different variance as shown in 2.8: from left to right we have a *Labial* occlusion, the *Alveolar* occlusion and the *Velar* occlusion. The pressure built up in the vocal tract, determine the produced sound depending on which occlusion is performed.

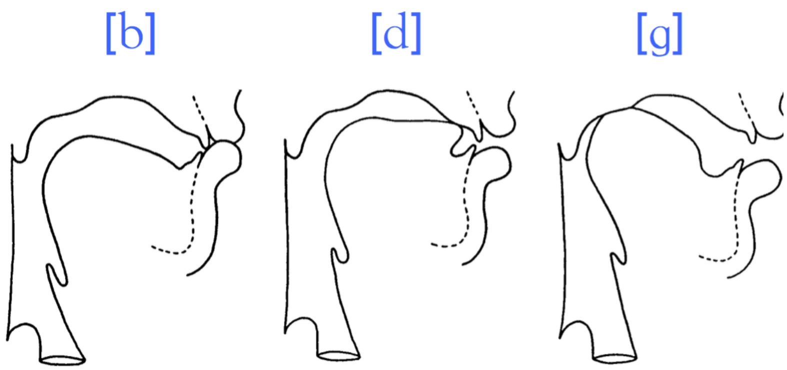

Figure 2.8: Stop production [1]

In American English there are six stop consonants, as represented in 2.9. As for the fricative consonants, the two main categories are the *Voiced* and *Unvoiced* sounds. Although, a particularity of the Unvoiced stops is that they are typically *aspirated* whereas in the Voiced ones there is a *voice-bar* during the closure movement. These two particularities are very useful where analyzing the formants because the frequencies are very well distinguished allowing a classification system to better understand the difference between stop phonemes.

Type	Voiced			Unvoiced		
Labial	/b/	b	bought	/p/	p	pot
Alveolar	/d/	d	dot	/t/	t	tot
Velar	/g/	g	got	/k/	k	cot

Figure 2.9: Stop examples of production [1]

2.6 Nasal Production

A **Nasal** is an occlusive consonant sound that is produced with lowering of the soft palate *(lowered velum)* at the back of the mouth, allowing the airflow to go out through the nostrils [20]. Because the airflow escapes through the nose, the consonants are produced with a closure in the vocal tract. 2.11 shows the three different positions to produce a nasal consonant. From left to right we have *Labial*, *Alveolar* and *Velar*.
Due to this particularity, the frequencies of nasal *murmurs* are quite similar. If we take a look on the spectrogram

in 2.10, it is possible to notice that nasal consonants have a high similarity. In a classification system, this can be a problem.

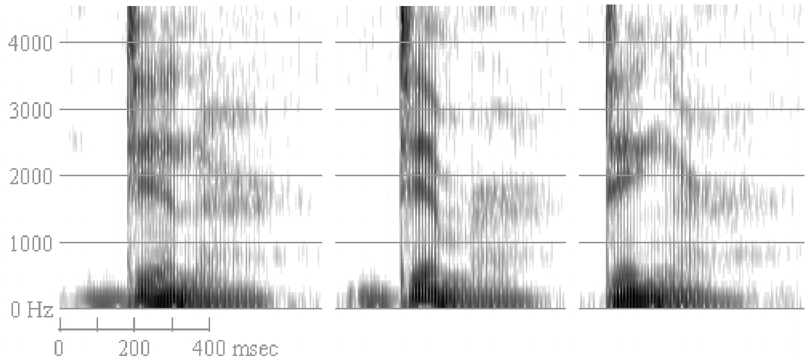

Figure 2.10: Nasal Spectrograms of **dinner**, **dimmer**, **dinger** [4]

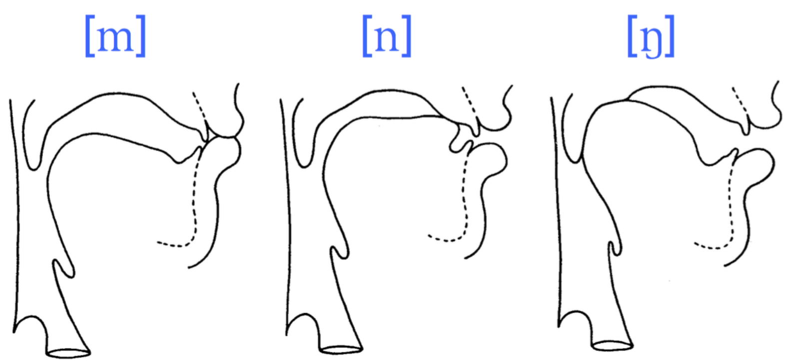

Figure 2.11: Nasal production [1]

Since the sound produced by a nasal is produced with an occlusive vocal tract, each consonant is **always attached** to a vowel and it can form an entire syllable. Although, in English, the consonant /**ŋ**/ always occur immediately after a vowel. In 2.12 are shown some examples of nasal consonants divided by articulation position.

Type	Nasal		
Labial	/m/	m	me
Alveolar	/n/	n	knee
Velar	/ŋ/	ng	sing

Figure 2.12: Nasal examples of production [1]

2.7 Semivowels Production

A **semivowel** is a sound that is very close to a vowel sound but it works more likely as a syllable boundary rather than a core of a syllable [21]. A typical example of semivowels in English are the **y** and **w** in words *yes* and *west*. In the *IPA* alphabet they are written /j/ and /w/ and they correspond to the vowels /iː/ and /uː/ in the words *seen* and *moon*. In 2.14 there are some examples of semivowels production.
The sound is produced by making a constriction in the oral cavity without having any sort of air turbulence. To achieve that, the articulation motion is slower than other consonants because the laterals[3] form a complete closer combined with a tongue tip. In this way the airflow has to pour out using the sides of the constriction.

[3]They are a pair of upper teeth that are located laterally from the central incisors [22]

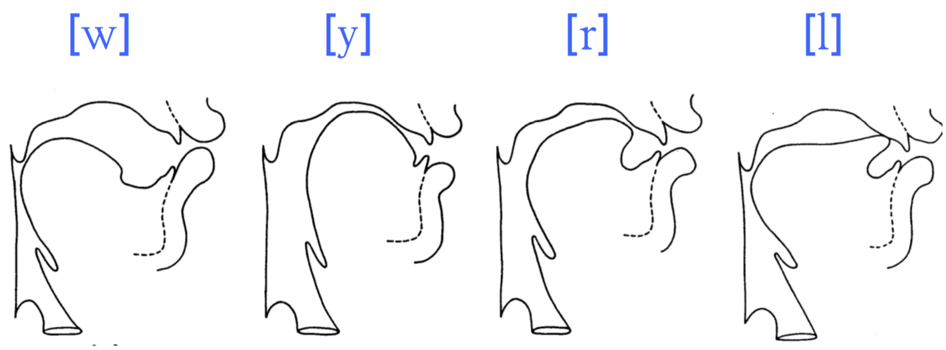

Figure 2.13: Semivowel production [1]

In American English there are four semivowels and they are depicted in 2.13. An important fact of semivowels is that they are always close to a vowel. Although, the /l/ can form an entire syllable by itself when there is no stress in a word.

Type	Semivowel			Nearest Vowel
Glides	/w/	w	wet	/u/
	/y/	y	yet	/i/
Liquids	/r/	r	red	/ɝ/
	/l/	l	let	/o/

Figure 2.14: Semivowel examples of production [1]

Acoustic Properties of Semivowels

Semivowels have some properties that are taken into account when doing any sort of analysis. In fact, /w/ and /l/ are the semivowels that are more confusable because both are characterized by a *low* range of frequencies for both formants *F1* and *F2*. Although, the /w/ can be distinguished by the *rapid falloff* in the F2 spectrogram whereas /l/ has more often a *high frequency energy* compared to /w/. The **energy** is the relationship between the *wavelength* and the *frequency*. So, having a high energy means that there is a high frequency value and a small wavelength [23]. The semivowel /y/ is characterized by having a very low frequency value in formant F1 and a very high in formant F2. The /r/ instead is presented with a very low frequency value of formant F3.

2.8 The Syllable

The definition of the **syllable** can be divided in two sub-definition: one from the phonetic point of view and one from the phonological point of view.
In phonetic analysis, the syllable is a basic unit of speech in which they *"are usually described as consisting of a centre which has little or no obstruction to airflow and which sounds comparatively loud; before and after that centre (...) there will be greater obstruction to airflow and/or less loud sound"* [24]. Taking the word *cat* (/kæt/) as example, the **centre** is defined by the vowel /æ/ in which takes place only a little obstruction. The surrounding *plosive* consonants (/k/ and /t/) the airflow is completely blocked [25].
A phonological definition of the syllable establishes that it is *"a complex unit made up of nuclear and marginal elements"*[26]. In this context, the vowels are considered the **Nuclear** elements or syllabic segments whereas the **Marginal** ones are the consonants or non-syllabic segments [25]. Considering the word *paint* (/peɪnt/) as example, the nuclear element is defined by the diphthong /eɪ/ whereas /p/ and /nt/ are the marginal elements.

2.8.1 Syllable Structure

In the phonological theory, the syllable can be decomposed in a hierarchical structure instead of a linear one. The structure starts with the σ letter in which represents not only the root, but the syllable itself. Immediately after,

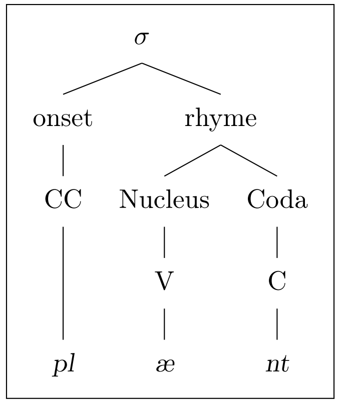

Figure 2.15: Tree structure of the word **plant**[4]

there are two *branches* called **constituents** that they represents the *Onset* and the *Rhyme*. The left branch includes any consonants that precede the vowel (or Nuclear element), whereas the right branch includes both the nuclear element and any consonants (or Marginal elements) that potentially could follow it.
Usually, the rhyme branch is further split in two other branches represented by the **Nucleus** and the **Coda**. The first on represent the nuclear element in the syllable. The second one instead, subsumes all the consonants that follow the Nucleus in the syllable [25]. In 2.15 there is a representation of the syllable structure based on the word *plant*.

[4]**C** means *Consonant* whereas **V** means *Vowel*

Chapter 3

Acoustics and Digital Signal Processing

In the past decade, digital computers have significantly helped *signal processing* to quantify a finite number of bits. The flexibility inherited from digital elements allowed the usage of a vast number of techniques in which had been not possible to implement in the past. Nowadays, digital signal processor have been used to perform multiple operations, such as *filtering*, *spectrum estimation* and many others algorithms [27].

3.1 Speech signals

The **speech** is the human way of communication. The protocol used in communication is based on a syntactic combination of different words taken from a very large vocabulary. Each word in the vocabulary is composed by a small set of vowels and consonants that combined with a phonetic units form a spoken word.
When a word is pronounced[1], a sounds is produced causing the air particles to be excited at a certain vibration rate. The source of our voice is due to the vibration of the vocal cords. The resultant signal is a *non-stationary* but it can be divided in segments since each phoneme has a common acoustic properties. In 3.1 is possible to notice how the pronounced words have a different shape as well as when the intensity of the voice is higher/lower during the pronunciation.

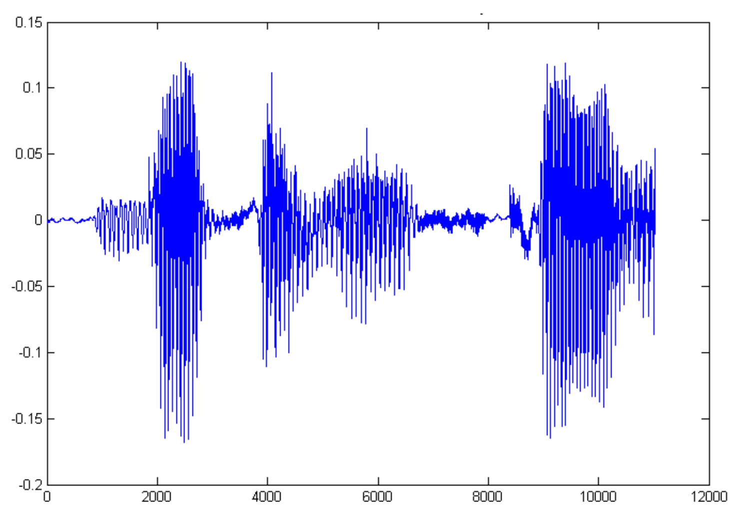

Figure 3.1: Example of a speech sound. In this case, the sentence **This is a story** has been pronounced [5]

The simplest form of sound is the *sinusoid* and it is the easiest waveform to describe because it corresponds to a **pure tone**. A pure tone consist in a waveform that consists only on one frequency. Other examples are the *cosine* or *sine* waves.

[1] 2 explains in details how phonemes are pronounced

3.1.1 Properties of Sinusoids

A sinusoid, is a simple waveform represented by a up and down movement. There are three important measures that has to be taken into consideration when defining the shape of the sinusoid: *amplitude*, *frequency* and *phase*.

Amplitude

The amplitude, from a sound point of view, corresponds to the *loudness* whereas in the soundwave it corresponds to the amount of **energy**. In general, to measure the amplitude, we use the unit called **deciBels** (dB) in which it is measured using a logarithmic scale relative to a standard sound [28].

Frequency

Frequency is the number of cycles per unit of time[2].To define cycle, we can think of an oscillation that starts from the middle line, goes to the maximum point, down to the minimum and get back to the middle point. The unit of measure of the frequency is calculated in **Hertz** (Hz). Also, if we calculate the time taken for one cycle, we estimate the so called **period**.
Frequency plays a fundamental role with the *pitch*. In fact, changing the number of oscillations but keeping the same waveform, we are able to increase or decrease the level of the pitch.

Phase

The **phase** measures the starting point position of the waveform. If the sinusoids start at the very minimum of the wave, the value of the phase is π radians whereas starting from the top of the wave it will have a phase of *zero*. When two sounds do not have the same phase, it is possible to perceive the difference in the time scale since one of the two is delayed compared to the other. When comparing two signals, there is the need to obtain a *"phase-neutral"* that means the comparison is made taking only into account Amplitude and Frequency. This method is called **autocorrelation** of the signals.

3.1.2 Spectrograms

A **Spectrogram** is the visual representation of an acoustic signal [29]. Basically, a Fourier Transformation is applied to the sound, in such a way to obtain the set of waveforms extracted form the original signal and separate their frequencies and amplitudes. The result is typically depicted in a graph with degrees of amplitude with a *light-dark* representation. Since amplitude represents the *energy*, having a darker shade means that the energy is more intense in a certain range of frequencies - lighter when there is low energy. In 2.10 there is an example of the spectrogram.
The visual feedback of the spectrogram is highly dependent from the **window size** of the Fourier Analysis. In fact, different sizes affect the levels of frequencies and time resolution.
If the window size is *short*, the adjacent **harmonics** are distorted but the time resolution is better [29]. An harmonic is an integer multiple of the fundamental frequency or component frequencies. This is helpful when we are looking for the *formant structure* because the striations created by the spectrogram highlights the individual pitch periods. On the other hand, a *wider* window size, helps to locate the harmonics because the band of the spectrogram are narrower.

3.2 Fourier Analysis

Fourier Analysis is the process that decompose a periodic waveform into a set of sinusoids having different amplitudes, phases and frequencies. Yet, if we add those waveforms again, we will obtain the original signal. The analysis has been involved in many scientific applications and the reason is due to the following transform properties:

- Linear transformation - the relationship between two modules is kept
- Exponential function are eigenfunctions of differentiation [30]
- Invertible - derived from the linear relationship

[2] In general, a unit of time is considered a single second

In signal processing, the Fourier analysis is used to isolate singular components of a complex waveform. A set of techniques consist in using **Fourier Transformation** on a signal in such a way to be able to manipulate the data in the easiest way possible but at the same time we have to be capable of inverting the transformation [31]. In the next subsections we describe the fundamental steps for manipulating a signal.

Sampling

Sampling is the process where a continuous signal is periodically measured every **T** seconds [27].
Consider a sound signal that varies in time a continuous function $s(t)$. For every T seconds, we need to measure the value of the function. This frame of time is called the *sampling interval* [32]. To calculate the sequence a sampled function is given as follow: $s(nT), \forall$ integer values of n. Thus, the *sampling rate* is the average number of samples obtained in a range of $T = 1sec$ [6]. An example of sampling is shown in 3.2.

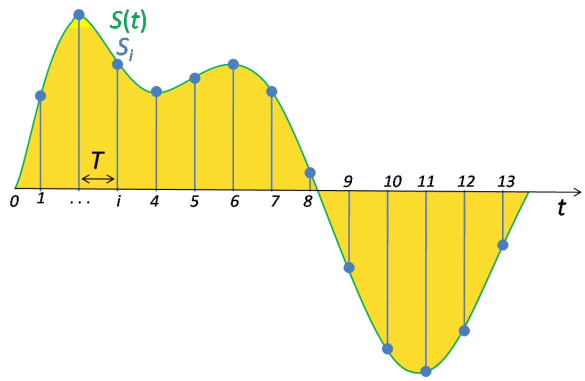

Figure 3.2: Example of signal sampling. The green line represents the continuous signal whereas the samples are represented by the blue lines [6]

As we mentioned above, using the Fourier Analysis we need to be able to reconstruct the original signal from the transformed one. To be able to, the **Nyquist-Shannon** theorem states that the sampling rate has to be larger as twice as the maximum frequency of the signal, in order to rebuild the original signal [33].
The *Nyquist sampling rate* is defined by the following equation:

$$f_s > f_{Nyquist} = 2f_{max} \tag{3.1}$$

Quantization

To finalize the transformation from a continuous signal to a discrete one, we need to *quantized* the signal in such a way to obtain a finite set of values. Unlike sampling in which permits to reconstruct the original signal, quantization is an irreversible operation that introduce a loss of information.
Consider x be the sampled signal and x_q the quantized one where x_q can be expressed as the signal x plus the error e_q. From here we have:

$$x_q = x + eq \Leftrightarrow e_q = x - x_q \tag{3.2}$$

Given the equation above, we can restrict the range of error to $-q/2... + q/2$ because we will not make a larger error than the half of the quantization step. From a mathematical point of view, the error-signal is a random signal with an uniform probability distribution between the range of $q/2and + q/2$, giving the following [34]:

$$p(e) = \begin{Bmatrix} \frac{1}{q} & for\frac{-q}{2} \leq e < \frac{q}{2} \\ 0 & otherwise \end{Bmatrix} \tag{3.3}$$

Given this reason, the quantization error also called quantization noise.

Windowing Signals

Speech sound is a **non-stationary** signal where its properties (amplitude, frequency and pitch) rapidly change over time [35]. Due to the quick changes of those properties, it makes hard to use *autocorrelation* or *Discrete Fourier*

Transformation. In 2 we highlighted the fact that phonemes have some invariant properties for a small period of time. Having said that, it is possible to apply methods that will take *short windows* (pieces of signal) and process them. This window is also called **frame**. Typically, the shape of this window is *rectangular* because one of the most used methods are the *Hanning* and *Hamming* in which the window covers the whole amplitude spectrum between a range. In 3.3 there is an example on how the Hamming window is taken from a signal. The rectangle called *Time Record*, is the frame that is extracted and processed by the windowing function.

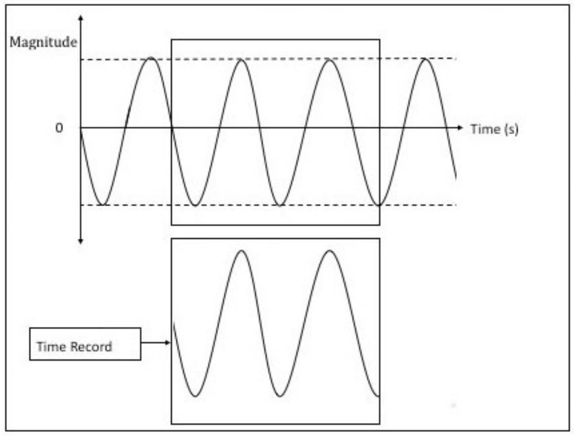

Figure 3.3: Hamming window example on a sinusoid signal

Hann Function

This is one of the most used windowing method in signal processing. The function is discrete and it is defined by 3.4a. The method is a linear combination of the *rectangular function* defined by 3.4b. Starting from the *Euler's formula*, it is possible to inject the rectangular equation as in 3.4c. From here, given the properties of the *Fourier Transformation*, the spectrum of the window function is defined as in 3.4d. Combining the spectrum with 3.4b we obtain 3.4e in which the signal modulation factor *disappears* when the windows are moved around time 0.

$$w(n) = 0.5\left(1-\cos\left(\frac{2\pi n}{N-1}\right)\right) \tag{3.4a}$$

$$w_r = \mathbf{1}_{[0,N-1]} \tag{3.4b}$$

$$w(n) = \frac{1}{2}\,w_r(n) - \frac{1}{4}e^{\mathrm{i}2\pi\frac{n}{N-1}}w_r(n) - \frac{1}{4}e^{-\mathrm{i}2\pi\frac{n}{N-1}}w_r(n) \tag{3.4c}$$

$$\hat{w}(\omega) = \frac{1}{2}\hat{w}_r(\omega) - \frac{1}{4}\hat{w}_r\left(\omega+\frac{2\pi}{N-1}\right) - \frac{1}{4}\hat{w}_r\left(\omega-\frac{2\pi}{N-1}\right) \tag{3.4d}$$

$$\hat{w}_r(\omega) = e^{-\mathrm{i}\omega\frac{N-1}{2}}\frac{\sin(N\omega/2)}{\sin(\omega/2)} \tag{3.4e}$$

The reason why this windowing method is one of the most diffuse is due to the *low aliasing*

Zero Crossing Rate

Zero crossing is the point of the function where the sign changes from a positive value to a negative one or vice versa. The method of counting the zero crossings is widely used in speech recognition for estimating the *fundamental* frequency of the signal. The zero-crossing rate is the rate of this positive-negative changes. Formally, it is defined as follow:

$$ZCR = \frac{1}{T-1}\sum_{t=1}^{T-1}\left\{\begin{matrix}1 & s_t s_{t-1} < 0\\ 0 & otherwise\end{matrix}\right\} \tag{3.5}$$

where s is the signal of length T.

The Discrete Fourier Transform

Before to jump into the definition of the Discrete Fourier Transformation (DFT), we need to introduce the Fourier Transformation (FT) from the mathematical point of view. The FT of a continuous-signal $x(t)$ is defined by the following equation:

$$X(\omega) = \int_{-\infty}^{\infty} x(t)e^{-j\omega t}dt, \qquad \omega \in (-\infty, \infty) \tag{3.6}$$

The discrete operation allows us to transform the equation above from an infinite space in a finite sum as follows:

$$X(\omega_k) = \sum_{n=0}^{N-1} x(t_n)e^{-j\omega_k t_n}, \qquad k = 0, 1, 2, \ldots, N-1 \tag{3.7}$$

where $x(t_n)$ is the *amplitude* of the signal at time t_n (sampling time). T is the sampling period in which the transformation is applied. $X(\omega_k)$ is the *spectrum* of the complex value x at frequency ω_k. Ω is the sampling interval defined by the *Nyquist-Shannon* theorem whereas N is the number of samples.

The motivation behind the DFT is that we want to move the signal from the *Time or space domain* to the *Frequency domain*. This allows us to analyze the spectrum in a simpler way. 3.4 shows the transformation.

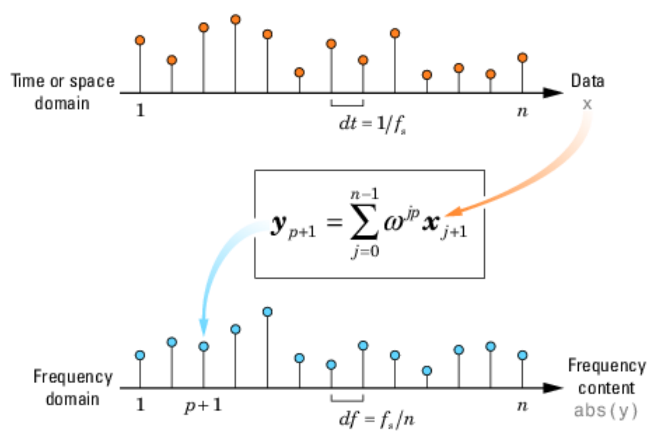

Figure 3.4: DFT transformation [7]

Chapter 4

Speech Recognition

Speech recognition is a sub-field of machine learning in which allows a computer program to extract and recognize words or sentences from a human being language, and converting them back to a machine language. Advance techniques nowadays, permits to understand natural speech for executing tasks. Google Voice Search[1] and Siri[2] are two examples of very advance speech recognition softwares with the capability of understanding natural language.

4.1 The Problem of Speech Recognition

Human languages are very complex and different among each other. Despite they might have a well-structured grammar, automatically recognition is still a very difficult problem since people have many ways to say the same thing. In fact, spoken language is different from written one because the articulation of verbal utterance is less strict and complicated.
The environment in which the sound is taken has a big influence on the speech recognition software because it introduces a *unwanted* amount of information in the signal. For this reason it is important that the system is capable of *identifying* and *filtering out* this surplus of information [36].

Another interesting set of problems are related to the speaker itself. Each person has a different body that means there are a variety of components that the recognition system has to take care of in such a way to be able to understand correctly. Gender, vocal tracts, speaking style, speed of the speech, regional provenience are fundamental parts that have to be taken in consideration when building the *acoustic model* for the system. Despite these features are unique for each person, there some common aspects that will be used to construct the model. The acoustic model represents the relationship between the acoustic signal of the speech and the phonemes related to it.

Ambiguity represents the major concern since natural languages have inherited it. In fact, it may happen that in a sentence we are not able to discriminate which words are actually intended [36]. In speech recognition there are two types of ambiguity: *homophones* and *word boundary ambiguity*.
Homophones refers to those words that are spelled in a different way but they **sound** the same. Generally speaking, these words are not correlated to each other but it happened that the sound is equivalent. Word boundary ambiguity instead, it *occurs when there are multiple ways of grouping phones into words*[36].

4.2 Architecture

Generally speaking, a speech recognition system is divided in three main components: the **Feature Extraction** (or Front End), the **Decoder** and the **Knowledge Base** (KB). In 4.1 the KB part is represented by the three sub-blocks called *Acoustic Model*, *Pronunciation Dictionary* and *Language Model*. The *Front End* takes as in input the voice signal where it is analyzed and converted in the so called *Features Vectors*. This last is the set of common properties that we discussed in 2. From here we can say that $\mathbf{Y}1 : N = y_1, ..., y_N$ where Y is the set of features vectors.
The second step consists in feeding the *Decoder* with vectors we obtained from the previous step, attempting to find the sequence of words $\mathbf{w}1 : L = w_1, ..., w_L$ that have most likely generated the set Y[8]. The decoder tries to find the likelihood estimation as follows:

[1] https://www.google.com/search/about/

[2] http://www.apple.com/ios/siri/

$$\widehat{w} = \underset{w}{arg\,max}\ P(\mathbf{w}|\mathbf{Y}) \tag{4.1}$$

The $P(w|Y)$ is difficult to find directly[3], but using Bayes' Rules we can transform the equation above in

$$\widehat{w} = \underset{w}{arg\,max}\ P(\mathbf{Y}|\mathbf{w})P(\mathbf{w}) \tag{4.2}$$

in which the probability $P(Y|w)$ and $P(w)$ are estimated by the *Knowledge Base* block. In particular, the *Acoustic Model* is responsible to estimate the first one whereas the *Language Model* estimates the second one.
Each word $\mathbf{w}$ is decomposed in smaller components called *phones*, representing the collection of phonemes $\mathbf{K}_w$ (see 2). We can describe the *pronunciation* as $\mathbf{q}^{(w)}_{1:K_w} = q_1, ..., q_{K_w}$. The likelihood estimation of the sequence of phonemes is calculated by a **Hidden Markov Model** (HMM). In the section, a general overview of HMM is given. We are not going to discuss a particular model because every speech recognition system uses a variation of the general HMM chain.

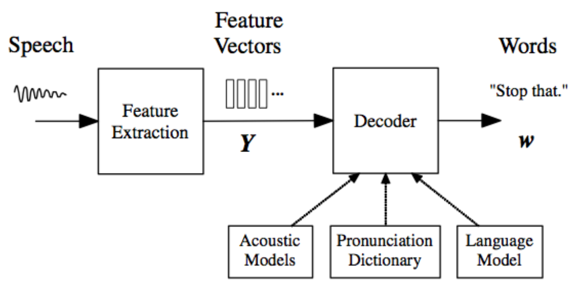

Figure 4.1: HMM-Based speech recognition system [8]

4.3 Hidden Markov Model

A definition given by [38] is the following: *"An Hidden Markov Model is a finite model that describes the probability distribution over an infinite number of possible sequences"*. Each sequence is determined by a set of *transition probabilities* in which describes the transitions among states. The **observation** (or outcome) of each state is generated based on the associated probability distribution. From an *outside* perspective, the *observer* is only able to see the outcome and not the state itself. Hence, the states are considered **hidden** which leads to the name Hidden Markov Model [39].

An HMM is composed by the following elements:

- The number of states (N)
- The number of observations (M), that becomes infinite if the set of observations is contiguous
- The set of transition probabilities, $\Lambda = \{a_{ij}\}$

  The set of probabilities is defined as follow:

$$a_{ij} = p\{\, q_{t+1} = j \,|\, q_t = i\,\},\ \ 1 \leq i, j \leq N, \tag{4.3}$$

where q_t is the state we are currently in and a_{ij} represent the transition from state i to j. Each transition should satisfy the following rules:

$$a_{ij} \geq 1,\ \ 1 \leq i, j \leq N, \tag{4.4a}$$
$$\sum_{j=1}^{N} a_{ij} = 1,\ \ 1 \leq j \leq N \tag{4.4b}$$

[3]There is discriminate way of finding the estimation directly as described in [37]

For each state S we can define the probability distribution $S = \{s_j(k)\}$ as follow:

$$s_j(k) = p\{\, o_t = v_k \,|\, q_t = j \,\},\;\; 1 \leq j \leq N,\; 1 \leq k \leq M \tag{4.5}$$

where v_k is the k^{th} observation whereas o_t is the outcome. Furthermore, $b_j(k)$ must satisfy the same stochastic rules described in 4.4.

A different approach is made when the number of observations is infinite. In fact, we are not going to use a set of discrete probabilities but instead a continuous probability density function. Given that, we can define the parameters of the density function by approximating it by a weighted sum of M Gaussian distributions φ [39]. We can describe the function as follow:

$$s_j(o) = \sum_{m=1}^{M} c_{jm}\varphi(\mu_{jm}, \Sigma_{jm}, o_t) \tag{4.6}$$

where c_{jm} is the weighted coefficients, μ_{jm} is the mean vector and Σ_{jm} is the covariance matrix. The coefficients should satisfy the stochastic rules in 4.4.
We can then define the initial state distribution as $\pi = \{\pi_i\}$ where

$$\pi_i = p\{q_I = i\,\},\;\;\; 1 \leq i \leq N \tag{4.7}$$

Hence, to describe the HMM with the discrete probability function we can use the following compact form

$$\lambda = (\Lambda, S, \pi) \tag{4.8}$$

whereas to denote the model with a continuous density function, we use the one described in 4.9

$$\lambda = (\Lambda, c_{jm}, \mu_{jm}, \Sigma_{jm}, \pi) \tag{4.9}$$

4.3.1 Assumptions

The theory behind HMM requires three important assumptions: the **Markov assumption**, the **stationarity assumption** and the **output independence assumption**.

The Markov Assumption

The Markov assumption assumes that the following state depends only from the state we are currently in as given in 4.3. The result model is also referred as *first order* HMM. Generally speaking though, the decision of the next coming state might depend on **n** previous states, leading to a n^{th} HMM order model. In this case, the transition probabilities is defined as follow:

$$a_{i_1 i_2 \ldots i_n j} = p\{\, q_{t+1} = j \,|\, q_t = i_1,\, q_{t-1} = i_2, ..., \, q_{t-k+1} = i_k \,\},\;\;\; 1 \leq i_1, i_2, ..., i_k, j \leq N \tag{4.10}$$

The Stationary Assumption

The second assumption states that the transition probabilities are *time-independent* when the transitions occur. This is defined by the following equation for any t_1 and t_2:

$$p\{\, q_{t+1} = j \,|\, q_{t_1} = i \,\} \;=\; p\{\, q_{t_2+1} = j \,|\, q_{t_2} = i \,\} \tag{4.11}$$

The Output Assumption

The last assumption says that the current observation is statistically independent from the previous observations. Let's consider the following observations:

$$O = o_1, o_2, ..., o_T \tag{4.12}$$

Now, recalling 4.8, it is possible to formulate the assumption as follow:

$$p\{\, O \,|\, q_1, q_2, ..., q_T, \lambda \,\} \;=\; \prod_{t=1}^{T} p\{\, o_t \,|\, q_t,\, \lambda \,\} \tag{4.13}$$

4.4 Evaluation

The next step in the HMM algorithm is the *evaluation*. This phase consists in estimating the likelihood probability of a model when it produces that output sequence. Generally speaking, there are two famous algorithms that have been extensively used: **forward** and **backward** probability algorithms. In the next two subsections, we report the two algorithms that either one can be used.

4.4.1 Forward probability algorithm

Let us consider the equation 4.13 where the probabilistic output estimation is given. The major drawback of this equation is that the computational cost is exponential in T because the probability of O is calculated directly. It is possible to improve the previous approach by *caching* the calculations. The cache is made using a *lattice* (or trellis) of states where at each time step, the α value is calculated by summing all the states at the previous time step [9].

The α value (or forward probability) can be calculated as follow:

$$\alpha_t(i) \ = \ P(o_1, o_2, ..., o_t, q_t = s_i|\lambda) \tag{4.14}$$

where s_i is the state at time t.
Given that, we can define the forward algorithm in three steps as follow:

1. Initialization:

$$\alpha_1(i) = \pi_i b_i(o_1), 1 \leq i \leq N \tag{4.15}$$

2. Induction step:

$$\left(\sum_{i=1}^{N} \alpha_t(i) a_{ij} \right) b_j(o_{t+1}) \ where \ 1 \leq t \leq T-1, \ 1 \leq j \, N \tag{4.16}$$

3. Termination:

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i) \tag{4.17}$$

The key of this algorithm stays in 4.16 where for each state s_j the α value contains the probability of the observed sequence from the beginning to time t. Given the previous algorithm, we can now calculate the new complexity. The direct algorithm has a complexity of $2TN^T$ whereas the new one is N^2T.

4.4.2 Backward probability algorithm

This algorithm is very similar to the previous one with the only difference when calculating the probability. Instead of estimating the probability as in 4.14, the backward algorithm estimates the likelihood of *"the partial observation sequence from* $t+1$ *to* T*, starting from state* s_i *"*[9].

The probability is calculated with the following equation:

$$\beta_t(i) \ = \ P(o_{t+1}, o_{t+2}, ..., o_T \,|\, q_t = s_i, \, \lambda) \tag{4.18}$$

The usage of either one depends on the type of problem we need to face.

4.5 Viterbi algorithm

The main goal of this algorithm is to discover the sequence of hidden states that are more likely to be produces given a sequence of observations. This block is called **decoder** (see 4.1 for reference). The *Viterbi algorithm* is one of the most used solution for finding a *single best sequence* for a given set of observations [9]. What makes this algorithm suitable for this problem, is the similarity with the forwarding algorithm with the only difference that

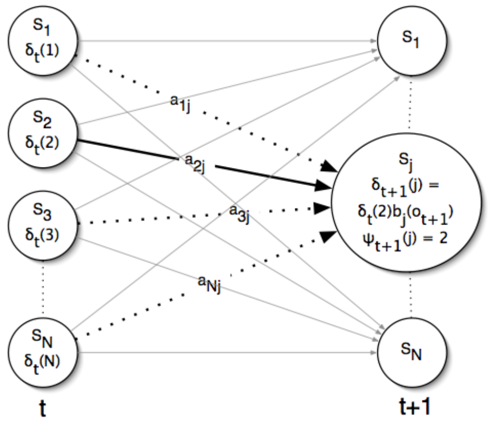

Figure 4.2: The recursion step [9]

instead of summing the transition probabilities at each step, it calculates the **maximum**. In 4.2 it is shown how the maximization estimation is calculated during the recursion step.
Let's define the probability of the most likely sequence for a given partial observation:

$$\delta_t(i) = \max_{q_1,q_2,\ldots,q_{t-1}} P(q_1, q_2, ..., q_t = s_i, o_1, o_2, ..., o_t \mid \lambda) \tag{4.19}$$

From here, we can define the steps of the algorithm as follow:

1. Initialization:

$$\delta_1(i) = \pi_i b_i(o_1),\ 1 \leq i \leq N, \phi_1(i) = 0 \tag{4.20}$$

2. Recursion:

$$\delta_t(j) = \max_{1\leq i\leq N} [\delta_{t-1}(i)a_{ij}]\, b_j(o_t),\ 2 \leq t \leq T,\ 1 \leq j \leq N, \tag{4.21a}$$

$$\psi_t(j) = \arg\max_{1\leq i\leq N} [\delta_{t-1}(i)a_{ij}],\ 2 \leq t \leq T,\ 1 \leq j \leq N, \tag{4.21b}$$

3. Termination:

$$P^* = \max_{1\leq i\leq N} [\delta_T(i)] \tag{4.22a}$$

$$q_t^* = \arg\max_{1\leq i\leq N} [\delta_T(i)] \tag{4.22b}$$

4. Backtracking:

$$q_t^* = \psi_{t+1}(q_{t+1}^*),\ t = T-1, T-2, ..., 1 \tag{4.23}$$

As previously stated, the Viterbi algorithm maximizes the probability during the recursion step. After that, the resulting state is used as a *back-pointer* in which during the backtracking step, the best sequence will be found. In 4.3 is depicted how the backtracking step works.

4.6 Maximum likelihood estimation

The last part of the model is represented by the *Learning* phase, in which the system is able to decide what it the final word pronounced by a user. With the usage of HMM models, it is possible to extract one or more sequences of states. The last piece of the puzzle is to estimate the sequence of words. To do so, a typical speech recognition

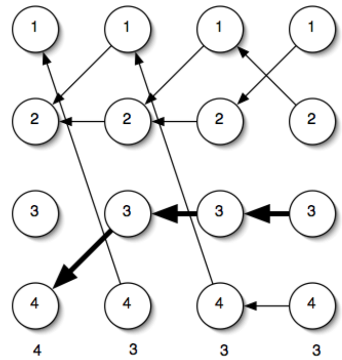

Figure 4.3: The backtracking step [9]

system uses the *Maximum Likelihood estimation* (MLE).

Given a sequence of n *independent* and *identical* observations $x_1, x_2, ..., x_n$. Assuming that the set of samples comes from a probability distribution with an *unknown density function* called $f_0(x_1, ..., x_n)$. The function belongs to a family of a certain kind of distributions in which θ is the *parameters vector* for that specific family.

Before using MLE, a *joint density function* must be specified first for all observations. Given the previous set of observation, the joint density function can be denoted as follow:

$$f(x_1, x_2, ..., x_n|\theta) = f(x_1|\theta) \times f(x_2|\theta) \times ... \times f(x_n|\theta) \tag{4.24}$$

Now, consider the same set of observations as a *fixed* parameters whereas θ is allowed to change without any constraint. From now on, this function will be called **likelihood** and denoted as follow:

$$L(\theta \sim x_1, x_2, ..., x_n) = f(x_1, x_2, ..., x_n|\theta) = \prod_{i=1}^{n} f(x_i|\theta) \tag{4.25}$$

In this case, $\sim$ indicates a simple separation between the parameters function and the set of observations. Often, there is a need to use the *log* function; that is transform the likelihood as follow:

$$ln\, L(\theta \sim x_1, x_2, ..., x_n) = \sum_{i=1}^{n} ln\, f(x_i|\theta) \tag{4.26}$$

To estimate the log-likelihood of a single observation, it is necessary to calculate the average of 4.26 as follow:

$$\hat{l} = \frac{1}{n} ln\, L \tag{4.27}$$

The *hat* in 4.27 indicates that the function is an estimator. From here we can define the actual MLE. This method estimates the θ_0 by finding the value of θ that returns the maximum value of $\hat{l}(\theta \sim x)$. The estimation is defined as follow if the maximum exists:

$$\hat{\theta}_{mle} \subseteq \{arg\,\underset{\theta}{max}\; \hat{l}\; (\theta \sim x_1, x_2, ..., x_n)\} \tag{4.28}$$

The MLE corresponds to the so called *maximum a posteriori estimation* (MPE) of *Bayes rule* when a uniformed prior distribution is given. In fact, θ is the MPE that maximize the probability. Given the Bayes' theorem we have:

$$P(\theta|x_1, x_2, ..., x_n) = \frac{f(x_1, x_2, ..., x_n|\theta)P(\theta)}{P(x_1, x_2, ..., x_n)} \tag{4.29}$$

where $P(\theta)$ is the prior distribution whereas $P(x_1, x_2, ..., x_n)$ is the averaged probability of all parameters. Due to the fact that the denominator of the Bayes' theorem is independent from θ, the estimation id obtained by maximizing $f(x_1, x_2, ..., x_n|\theta)P(\theta)$ with respect of θ.

4.7 Gaussian Mixture Model

A Gaussian mixture model is a probabilistic model where the it is assumed that the set of points come from a *mixture model*, in particular from a fixed number of *Gaussian distributions* where the parameters are *unknown*. This approach can be thought of a generalization of the clustering algorithm called *k-means* where we are looking for the **covariance** and the center of the Gaussian distribution and not only the centroids [40]. There are different way for fitting the mixture model, but we are going to focus in particular to the one where the expectation-maximization is involved (see 4.6).

Let the following equation defining a weighted sum of N Gaussian densities component:

$$p(\mathbf{x}|\lambda) = \sum_{i=1}^{N} w_i \, g(\mathbf{x}|\mu_i, \Sigma_i) \tag{4.30}$$

where $\mathbf{x}$ defines the set of features (data-vector) of continuous values. The sequence $w_i = 1, ..., N$ represents the set of mixture weights whereas the function $g(\mathbf{x}|\mu_i, \Sigma_i)$, $i = 1, ..., N$ defines the Gaussian densities component. The following equation specifies each Gaussian component's form:

$$g(\mathbf{x}|\mu_i, \Sigma_i) = \frac{1}{(2\pi)^{D/2} \, |\Sigma_i|^{1/2}} exp\left\{ -\frac{1}{2}(\mathbf{x} - \mu_\mathbf{i})' \; \Sigma_i^{-1} \; (x - \mu_\mathbf{i}) \right\} \tag{4.31}$$

where $\mu_\mathbf{i}$ is the mean vector and Σ_i is the covariance matrix. Given that, we can assume that the mixture satisfy the constrain that $\Sigma_{i=1}^{N} \; w_i = 1$.

With the notation in 4.32, we can now define the complete GMM since all the component densities are parameterize by the covariance matrices, the mean vectors and the mixture weights [41]

$$\lambda = \{w_i, \mu_\mathbf{i}, \Sigma_i\} \;\; i = 1, ..., N \tag{4.32}$$

Let's break down the variants in 4.32. The choice of model configuration highly depends on the available dataset. In fact, to estimate the GMM parameters we have to determine the covariance matrix Σ_i. This can be either full rank or constrained to be diagonal. In the first case, all rows and columns are linearly independent and all the values are taken into account, whereas in the second case, we consider only the values in the diagonal. The covariance matrix is not the only parameter that needs to be carefully chosen. In fact, the *number of components* in general refers to the amount of possible *"clusters"* in the dataset.

It is important to note that in speaking recognition, it is allowed to assume that size of the acoustic space of the spectral. The spectral is referred to the phonetic events as we described in 2. In fact, these acoustic classes have well defined features that allows the model to distinguish one phoneme from another. For the same reason, GMM is also used in *speaker recognition* in which the vocal tracts spectral is taken into account to distinguish a speaker from another [42].

Continuing with the speaker recognition example, we can think of the spectral shape i as an acoustic class in which can be represented by the mean μ_i of the $i - th$ component density. The variation in the spectrum can be defined as the covariance matrix Σ_i. Also, a GMM can be views as a Hidden Markov Model with a single state assuming that the feature vectors are independent as well as the observation density from the acoustic classes is a Gaussian mixture [41] [43].

Chapter 5

Implementation

In this chapter we explain the infrastructure that performs all the necessary steps to produce an efficient feedback. A general overview is given and for each section, we describe in particular the tools as well as the way we manipulated the data in order to obtain the information useful for the user. The chapter is divided in two parts: the first part focuses on the back-end and the services we used to extract the features we described in 4. The second part describe the front-end, that is, the *Android*[1] application (called **PARLA**[2]) with a particular focus on the feedback page and the general usage.

5.1 General architecture

In 5.1 is shown the general architecture of the infrastructure. The flow displays only the *pronunciation testing* phase:

1) User says the sentence using the internal microphone of the smartphone (or through the headset)
2) The application sends the audio file to the *Speech Recognition service*
3) The result of step 2 is sent to the *Gaussian Mixture Model service*
4) The result of step 3 is sent back to the application where a *Feedback page* is displayed
5) A short explanation for each chart is given to the user
6) Back to step 1

5.2 Server

5.3 Data collection

5.3.1 Data pre-processing

5.3.2 Training the Speech Recognition model

5.3.3 Training the Gaussian Mixture Model

- BIC, AIC and other things here

[1] https://www.android.com
[2] https://github.com/davideberdin/PARLA

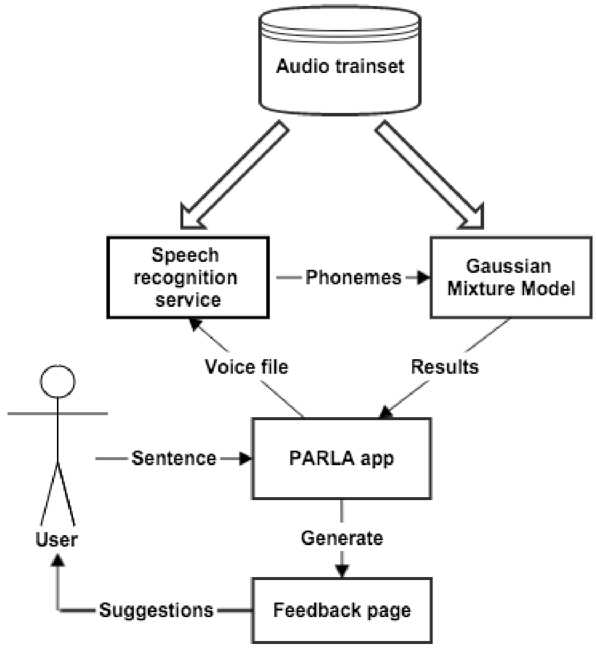

Figure 5.1: General architecture of the infrastructure

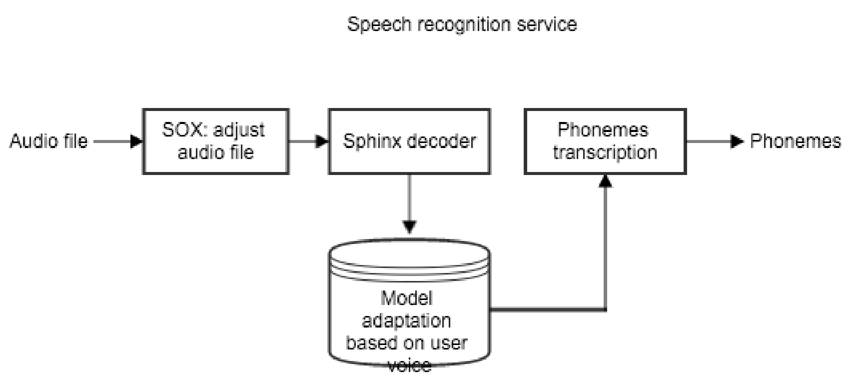

Figure 5.2: Architecture of the Speech recognition service

5.4 Android application

5.4.1 Layouts

5.4.2 Feedback layout

5.4.3 Usage procedure

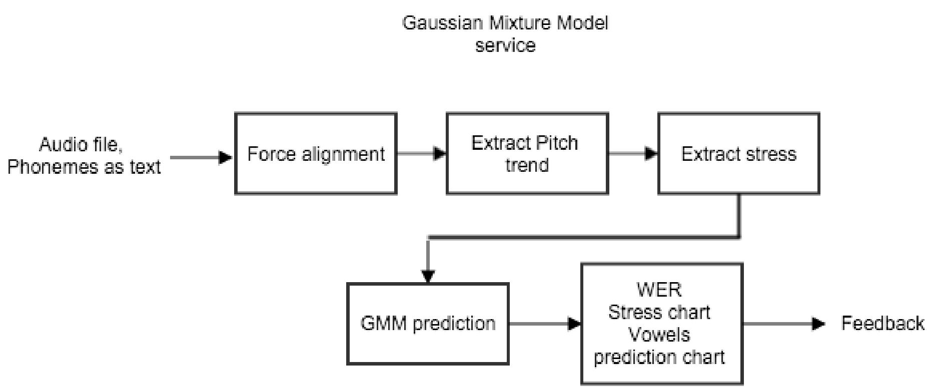

Figure 5.3: Architecture of the Gaussian Mixture Model service

Chapter 6

User studies and Evaluation

Chapter 7

Conclusions

Chapter 8

Future Works

Bibliography

[1] J. Glass and V. Zue, "6.345 automatic speech recognition," Spring 2003. http://ocw.mit.edu, (Massachusetts Institute of Technology: MIT OpenCourseWare), (Accessed 23 Sep, 2015). License: Creative Commons BY-NC-SA.

[2] "The spectrum of acousting," 2015. accessed 2015-09-28. Available: http://www.hum.uu.nl/uilots/lab/courseware/phonetics/basics_of_acoustics_2/formants.html.

[3] "Rp vowel length: some details," 2015. accessed 2015-10-08. Available: https://notendur.hi.is/peturk/KENNSLA/02/TOP/VowelLength0.html#lengths.

[4] "How do i read a spectrogram ?," 2015. accessed 2015-10-28. Available: https://home.cc.umanitoba.ca/~robh/howto.html.

[5] "Example of autocorrelation," 2015. accessed 2015-10-28. Available: http://www.eng.usf.edu/~lazam2/Project/sht_time_timedom/xmp_acr.htm.

[6] "File:signal sampling.png," 2015. accessed 2015-11-08. Available: https://en.wikipedia.org/wiki/File:Signal_Sampling.png.

[7] "Discrete fourier transform (dft)," 2015. accessed 2015-11-08. Available: http://www.mathworks.com/help/matlab/math/discrete-fourier-transform-dft.html.

[8] M. Gales and S. Young, "The application of hidden markov models in speech recognition," *Foundations and trends in signal processing*, vol. 1, no. 3, pp. 195–304, 2008.

[9] "Hidden markov model tutorial," 2015. accessed 2015-11-08. Available: http://digital.cs.usu.edu/~cyan/CS7960/hmm-tutorial.pdf.

[10] T. M. Derwing and M. J. Munro, "Second language accent and pronunciation teaching: A research-based approach," *Tesol Quarterly*, pp. 379–397, 2005.

[11] P. Medgyes, "When the teacher is a non-native speaker," *Teaching English as a second or foreign language*, vol. 3, pp. 429–442, 2001.

[12] A. Gilakjani, S. Ahmadi, and M. Ahmadi, "Why is pronunciation so difficult to learn?," *English Language Teaching*, vol. 4, no. 3, p. p74, 2011.

[13] M. Rost and C. Candlin, *Listening in language learning*. Routledge, 2014.

[14] "Word stress - british council," 2015. accessed 2015-09-28. Available: https://www.teachingenglish.org.uk/article/word-stress.

[15] D. Edge, K.-Y. Cheng, M. Whitney, Y. Qian, Z. Yan, and F. Soong, "Tip tap tones: mobile microtraining of mandarin sounds," in *Proceedings of the 14th international conference on Human-computer interaction with mobile devices and services*, pp. 427–430, ACM, 2012.

[16] A. Head, Y. Xu, and J. Wang, "Tonewars: Connecting language learners and native speakers through collaborative mobile games," in *Intelligent Tutoring Systems*, pp. 368–377, Springer, 2014.

[17] P. Boersma and D. Weenink, "{P} raat: doing phonetics by computer," 2010.

[18] "What are fricatives ?," 2015. accessed 2015-10-28. Available: http://www.pronuncian.com/Lessons/default.aspx?Lesson=9.

[19] "Consonants: Stops," 2015. accessed 2015-10-28. Available: http://facweb.furman.edu/~wrogers/phonemes/phono/stop.htm.

[20] "Nasal speech sound," 2015. accessed 2015-10-28. Available: http://www.britannica.com/topic/nasal-speech-sound.

[21] P. Ladefoged and I. Maddieson, "The sounds of the world's languages," *Language*, vol. 74, no. 2, pp. 374–376, 1998.

[22] "Maxillary lateral incisor," 2015. accessed 2015-10-28. Available: https://en.wikipedia.org/wiki/Maxillary_lateral_incisor.

[23] "Syllable, stress & accent," 2015. accessed 2015-10-28. Available: http://hubblesite.org/reference_desk/faq/answer.php.id=73&cat=light.

[24] P. Roach and E. Phonetics, "Phonology: A practical course," *Cambridge UP Cambridge*, 2000.

[25] "What is the relationship between wavelength, frequency and energy?," 2015. accessed 2015-10-28. Available: http://www.personal.rdg.ac.uk/~llsroach/phon2/mitko/syllable.htm.

[26] J. Laver, *Principles of phonetics.* Cambridge University Press, 1994.

[27] S. J. Orfanidis, *Introduction to signal processing.* Prentice-Hall, Inc., 1995.

[28] "Properties of sinusoids," 2015. accessed 2015-10-28. Available: http://web.science.mq.edu.au/~cassidy/comp449/html/ch03s02.html.

[29] "So what is a spectrogram anyway?," 2015. accessed 2015-11-08. Available: https://home.cc.umanitoba.ca/~robh/howto.html.

[30] L. C. Evans, "Partial differential equations and monge-kantorovich mass transfer," *Current developments in mathematics*, pp. 65–126, 1997.

[31] L. R. Rabiner and B. Gold, "Theory and application of digital signal processing," *Englewood Cliffs, NJ, Prentice-Hall, Inc., 1975. 777 p.*, vol. 1, 1975.

[32] M. Weik, *Communications standard dictionary.* Springer Science & Business Media, 2012.

[33] "Sampling and quantization," 2015. accessed 2015-11-08. Available: https://courses.engr.illinois.edu/ece110/content/courseNotes/files/?samplingAndQuantization.

[34] "Digital signals - sampling and quantization," 2015. accessed 2015-11-08. Available: http://rs-met.com/documents/tutorials/DigitalSignals.pdf.

[35] "Windowing signal processing," 2015. accessed 2015-11-08. Available: http://www.cs.tut.fi/kurssit/SGN-4010/ikkunointi_en.pdf.

[36] M. Forsberg, "Why is speech recognition difficult," *Chalmers University of Technology*, 2003.

[37] M. Gales, "Discriminative models for speech recognition," in *Information Theory and Applications Workshop, 2007*, pp. 170–176, IEEE, 2007.

[38] S. R. Eddy, "Hidden markov models," *Current opinion in structural biology*, vol. 6, no. 3, pp. 361–365, 1996.

[39] "Definition of hidden markov model," 2015. accessed 2015-09-08. Available: http://jedlik.phy.bme.hu/~gerjanos/HMM/node4.html.

[40] "Gaussian mixture models," 2015. accessed 2015-11-08. Available: http://scikit-learn.org/stable/modules/mixture.html.

[41] D. A. Reynolds, T. F. Quatieri, and R. B. Dunn, "Speaker verification using adapted gaussian mixture models," *Digital signal processing*, vol. 10, no. 1, pp. 19–41, 2000.

[42] D. A. Reynolds, "A gaussian mixture modeling approach to text-independent speaker identification," 1992.

[43] D. Reynolds, R. C. Rose, *et al.*, "Robust text-independent speaker identification using gaussian mixture speaker models," *Speech and Audio Processing, IEEE Transactions on*, vol. 3, no. 1, pp. 72–83, 1995.

***Elwood**: It's 106 miles to Chicago, we got a full tank of gas, half a pack of cigarettes, it's dark and we're wearing sunglasses.*
***Jake**: Hit it.*

The Blues Brothers