

A Diphone-Based Maltese Speech Synthesis System

Daniel Magro

Supervisor(s): Dr Claudia Borg and Dr Andrea DeMarco



Faculty of ICT
University of Malta

May 2019

*Submitted in partial fulfillment of the requirements for the degree of B.Sc. ICT in
Artificial Intelligence (Hons.)*

FACULTY/INSTITUTE/CENTRE/SCHOOL ICT

DECLARATIONS BY UNDERGRADUATE STUDENTS

Student's I.D. /Code 484497M

Student's Name & Surname Daniel Magro

Course Bachelor of Science in Information Technology (Honours) (Artificial Intelligence)

Title of Long Essay/Dissertation

A Diphone-Based Maltese Speech Synthesis System

Word Count 15,500

(a) Authenticity of Long Essay/Dissertation

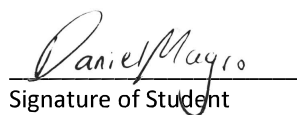
I hereby declare that I am the legitimate author of this Long Essay/Dissertation and that it is my original work.

No portion of this work has been submitted in support of an application for another degree or qualification of this or any other university or institution of higher education.

I hold the University of Malta harmless against any third party claims with regard to copyright violation, breach of confidentiality, defamation and any other third party right infringement.

(b) Research Code of Practice and Ethics Review Procedures

I declare that I have abided by the University's Research Ethics Review Procedures.


Signature of Student

DANIEL MAGRO
Name of Student (in Caps)

30/05/2019
Date

Abstract:

In Malta, there are 7,100 vision-impaired (1.9% of the Maltese population), and over 24,000 illiterate (6.4% of the Maltese population), Maltese speakers [1]. These people are unable to consume any content written in Maltese, be it a book, a news article, or even a simple Facebook post. This dissertation sets out to solve that problem by creating a Text to Speech (TTS) system for the Maltese language.

While there has been work in the area, at the time of writing there are no available TTS systems for Maltese, thus almost the entire system had to be built from scratch. In light of this, a Diphone-Based Concatenative Speech System was chosen as the type of synthesiser to implement. This was due to the minimal amount of data needed, requiring less than 20 minutes of recorded speech.

A simple ‘Text Normalisation’ component was built, which converts integers between 0 and 9,999 written as numerals to their textual form. While this is far from covering all the possible forms of Non-Standard Words (NSWs) in Maltese, the modular nature in which it was built allows for easy upgrading in future work. A ‘Grapheme to Phoneme (G2P)’ component which then converts the normalised text into a sequence of phonemes (basic sounds) that make up the text was also created, based on an already existing implementation by Crimsonwing [34].

Three separate ‘Diphone Databases’ were made available to the speech synthesiser. One of these is the professionally recorded English Diphone database FestVox’s ‘CMU US KAL Diphone’¹ [8]. The second and third were created as part of this work, one with diphones manually extracted from the recorded carrier phrases in Maltese, the other with diphones automatically extracted using Dynamic Time Warping (DTW). The Time Domain - Pitch Synchronous OverLap Add (TD-PSOLA) concatenation algorithm was implemented to string together the diphones in the sequence specified by the G2P component.

On a scale of 1 to 5, the speech synthesised when using the diphone database of manually extracted diphones concatenated by the TD-PSOLA algorithm was scored 2.57 for naturalness, 2.72 for clarity, and most important of all, 3.06 for Intelligibility by evaluators. These scores were higher than those obtained when using the professionally recorded English diphone set.

¹http://www.festvox.org/dbs/dbs_kal.html

Acknowledgements:

I would like to express my sincere gratitude to my supervisors Dr Claudia Borg and Dr Andrea DeMarco for their guidance and support throughout this dissertation. I'd also like to thank my family for their support throughout my studies, as well as my friends and colleagues. Last but not least, I'd like to thank my cats for keeping me company during my late night writing sessions.

Contents

1	Introduction	1
1.1	Problem Definition	1
1.2	Motivation	1
1.3	Approach	2
1.4	Aims and Objectives	3
1.5	Dissertation Layout	3
2	Background	4
3	Literature Review	5
3.1	Review of Existing Technologies	5
3.2	Choosing a Unit Size	6
3.3	Converting Graphemes to Phonemes	7
3.4	Generating Carrier Text and Recording Diphones	8
3.5	Diphone Extraction and Segmentation	9
3.6	Concatenation of Diphones	10
3.7	Prosody Modification	12
3.8	Front-End and Handling Non-Standard Words	12
3.9	Evaluation	13
4	Implementation	14
4.1	Text Normalisation - Non-Standard Word Handling	15
4.2	G2P - Converting Graphemes to Phonemes	17
4.3	Diphone Database	17
4.3.1	Generating Carrier Material	18
4.3.2	Recording Carrier Material	19
4.3.3	Diphone Extraction	20
4.4	Concatenation of Diphones	23
5	Evaluation	27
5.1	Evaluation Plan	27
5.1.1	Evaluating the Diphone Databases and the Overall Performance of the Synthesiser	27
5.1.2	Evaluating the Diphone Concatenation Algorithm	28

5.1.3	Evaluating the Front End	29
5.2	Evaluation Results and Discussion	29
5.2.1	Text Normalisation and G2P	29
5.2.2	Diphone Databases and Overall Synthesis	30
5.2.3	Diphone Concatenation Algorithm	31
6	Conclusions and Future Work	33
6.1	Future Work	34
6.1.1	Core Functionality	34
6.1.2	Usability and Accessibility	35
6.2	Discussion	36
	Appendix A - The Evaluation Survey	42

List of Figures

1	The sound wave of the word ‘ <i>sar</i> ’, with phones labelled at the top and diphones at the bottom	5
2	The sound wave of the word ‘ <i>sur</i> ’, with phones labelled at the top and diphones at the bottom	5
3	Block Diagram with all the components involved in the entire TTS system	14
4	Carrier Phrases that will be used during recording	19
5	Visual representation of Phones and Diphones	19
6	Uploading Recorded carrier phrases (.wav) and their transcripts (.txt) to WebMAUS Basic for forced alignment of phonemes	20
7	The result of WebMAUS Basic’s Forced Alignment tool shown in EMU-webApp	21
8	Diphone list specifying how diphones are to be extracted from carrier phrases	22
9	A graphical explanation of the TD-PSOLA algorithm showing the pitch being increased i.e. the period being reduced in (a) and the pitch being lowered i.e. the period being increased in (b). Reproduced from [23]. . . .	26
10	Question Type 1 of the Evaluation Survey, asked for Phrases 1 and 2 of each Diphone Database	42
11	Question Type 2 of the Evaluation Survey, asked for Phrase 3 of each Diphone Database	43
12	Question Type 3 of the Evaluation Survey, asked after 3 Phrases are presented from each Diphone Database	43
13	Question Type 4 of the Evaluation Survey, asked after presenting two versions of the same Phrase, one using simple concatenation, the other with TD-PSOLA	44

List of Tables

1 The results of the evaluation survey 31

List of Abbreviations and Acronyms

ARPABET	Advanced Research Projects Agency alphaBET
CE	Categorical Estimation test
DB	database
DRT	Diagnostic Rhyme Test
DSP	Digital Signal Processing
DTW	Dynamic Time Warping
EGG	Electroglottograph
FITA	Foundation for Information Technology Accessibility
G2P	Grapheme to Phoneme
HMM	Hidden Markov Model
HNH	Harmonic Plus Noise Model
IPA	International Phonetic Alphabet
MFCC	Mel-Frequency Cepstral Coefficient
MLP	Multi-Layer Perceptron
MPM	McLeod Pitch Method
NLP	Natural Language Processing
NSW	Non-Standard Word
OLA	OverLap Add
OOV	Out of Vocabulary
PDA	Pitch Detection Algorithm
RMS	Root Mean Square
SVM	Support Vector Machine
TD-PSOLA	Time Domain - Pitch Synchronous OverLap Add
TTS	Text to Speech

1 Introduction

The aim of this thesis is to transform any given string of text in the Maltese Language into a clear and intelligible synthesised voice. Different well-established speech synthesis methods will be explored, and one will be selected based on the resources available (linguistic datasets), computational resources available, amount of training data required and typical output quality. Furthermore, some problems which are common to many speech synthesis systems will be addressed, for example, handling of Non-Standard Words (NSWs).

1.1 Problem Definition

Text to Speech (TTS), or Speech Synthesis, is the conversion of a written text in a particular language into speech.

Work on the topic has been carried out in a PhD thesis by P. Micallef [36], however the developed TTS system was not found to be available. The Foundation for Information Technology Accessibility (FITA) also collaborated with Crimsonwing between 2009 and 2012 to develop a Maltese TTS system [34]. This TTS system, however, requires an account to download², for which registrations seem to be closed at the time of writing (“You are not allowed to do this.” pops up after trying to register). Furthermore, the solution is 1.1 GB in size, and is only available for Windows, which severely limits its portability to different platforms, especially smartphones. Therefore, there are no available TTS synthesis systems for Maltese.

Maltese is a low-resourced language, meaning very few resources exist for it. There are currently no databases of recorded and labelled Maltese speech available. The Maltese TTS system must be designed with these very minimal resources in mind.

1.2 Motivation

Speech Synthesis Systems, in general, are not just a convenience for the many but also an enabler for certain members of society. For one, it gives the 7,100 vision-impaired (1.9% of Maltese) and over 24,000 illiterate Maltese speakers (6.4% of Maltese) [1], the ability to consume written text from newspaper articles, or any other written media for that matter.

Secondly, speech synthesis is so important nowadays as speech is becoming an increasingly more convenient and effective way for people to interact with their smartphone, smart

²<https://fitamalta.eu/projects/maltese-speech-engine-synthesis-erdf-114/download-maltese-speech-engine/>

speaker (such as Google Home), or other smart devices.

1.3 Approach

As mentioned in Section 1.1, while two speech synthesisers for Maltese have been built in the past, these are not available to download and use. As was also mentioned in Section 1.1, Maltese is a low-resourced language, so the majority of components needed to make up a TTS system will need to be built from scratch.

With this low-resource constraint in mind, Neural Network based solutions were ruled out, as they require hours of data to produce satisfactory results. On the other hand, Concatenative speech synthesisers require far less data to produce good quality speech. In order to build a Concatenative speech synthesiser, the following components are required:

1. Text Normalisation - the component which converts the user's text input to a normalised form, for example converting numerals to their textual form or expanding abbreviations.
2. Grapheme to Phoneme (G2P) - the component which converts the normalised text to sequence of phonemes (a sequence of basic sounds that make up the text when spoken).
3. Speech Unit Database - a database of speech units (very short extracts, usually shorter than a second, from recorded speech) that can be concatenated to synthesise speech.
4. Speech Unit Concatenation Algorithm - the algorithm which smoothly joins together the speech units loaded from the speech unit database as determined by the G2P algorithm.

No text normalisation programs for Maltese could be found, so one would need to be built from scratch. This, however, does not need to be an advanced one, and a simple one with basic features will suffice. A G2P program for Maltese was found. Crimsonwing offers a rule-based implementation of a G2P program for Maltese [10].

No Speech Unit Databases of any speech unit size were found for Maltese. This means that a procedure for designing carrier text, recording the speech and extracting the speech units from the recordings would need to be defined. While no speech unit databases were available for Maltese, there are of course databases for units extracted from other languages, such as English. One such example is FestVox's 'CMU US KAL Diphone'³ [8]. Using this database is certainly unideal, however can serve as a guide to a diphone's

³http://www.festvox.org/dbs/dbs_kal.html

boundaries, or possibly, even as a prototype database. No complete, open-source speech unit or audio concatenation algorithms were found either. This meant that one would also have to be built from scratch. A concatenation algorithm which produces notably good quality speech while being a light-weight implementation and relatively simpler to implement is Time Domain - Pitch Synchronous OverLap Add (TD-PSOLA) [46].

1.4 Aims and Objectives

The principal aim of this thesis is to develop a speech synthesis system which given any Maltese text, can produce an intelligible audio output. This aim can be subdivided into smaller objectives as follows:

- Building a front-end for the speech synthesis system which normalises text. It is not within the scope of this thesis to cater for every NSW that can appear in the Maltese language, but rather to build a front-end such that further capabilities can be added at a later stage. To demonstrate the capabilities of the front-end, handling of integers written in their numeral form will be implemented.
- Building a component which can convert any body of Maltese text to a sequence of speech units that will make up the intended speech.
- Collecting a database of human voices which can be processed such that they can be used within the chosen speech synthesis system.
- Building a lightweight, efficient and simple back-end for a speech synthesis system, that apart from being able to synthesise text, can also serve as a solid foundation for any future work in the area. The principal component of the back-end will be the Speech Unit Concatenation Algorithm.

1.5 Dissertation Layout

In Section 2 a background on some of the essential terms and concepts used throughout this dissertation is given. In Section 3 the literature relating to the topic is discussed and reviewed on its relevance to the problem being tackled. In Section 4 the design of the implemented TTS system is described from a high-level, after which the implementation of the individual components is explained. In Section 5 the implemented TTS system is evaluated to measure its performance. Section 6 extracts conclusions on whether the work carried out achieved the objectives of this dissertation, and also mentions improvements that could be implemented in future work.

2 Background

A grapheme is the smallest “distinctive visual unit” of an alphabet, i.e. letters [38]. A phoneme is to speech, what the grapheme is to an alphabet, it is the most basic unit of speech/sound in a language [38]. Typically, each grapheme corresponds to a phoneme, however there are exceptions. For example, in Maltese, the letter (grapheme) ‘għ’ has no associated phoneme as it is silent, however, it may affect the phonemes generated from surrounding graphemes.

The word ‘*sar*’ can be converted into its phonemic transcription, ‘S AH R’. This transcription essentially states what sound each letter (grapheme) in the word makes, or rather, what series of basic sounds (phonemes) make up a given word. This phonemic transcription is determined by a process known as Grapheme to Phoneme (G2P). These ‘building blocks’ of speech are called speech ‘units’. If all the phonemes that make up the word ‘*sar*’ were to be strung together (in this case the phonemes ‘S’, ‘AH’ and ‘R’), the result would sound very similar to what the word actually sounds like. This type of system is called a concatenative speech synthesiser with phonemes as its speech unit, thus a ‘phoneme-based’ concatenative speech synthesiser. Units in a concatenative speech system or a unit-selection speech system are the standard utterances which will be stored in the ‘speech database’ and be chained together when generating speech. These units are commonly phonemes, diphones, half-phones, allophones or syllables [27].

Consider the sound waves of the words ‘*sar*’ (S AH R), shown in Figure 1, and ‘*sur*’ (S UH R), shown in Figure 2. Even though the ‘S’ and ‘R’ phonemes are identical in both words, and thus should sound identical, in Figures 1 and 2 the parts of the waveforms that contain those phonemes are noticeably different, especially at the phonemes’ edges. This occurs due to the effects of coarticulation, i.e. since the phoneme in the middle is different, the phonemes surrounding it change accordingly when they transition. The centre of a singular phone is the most stable part of that phone, whereas the ‘edges’, or the areas where a transition from one phone to the next occurs, is the area that machines find hardest to produce.

As opposed to phonemes, diphones capture this coarticulation better, as they start from the most stable part of the first phone, cover the transitionary area where coarticulation happens, and end on the most stable part of the next phone [31]. It is because of this that diphones are so often used as a unit [31].

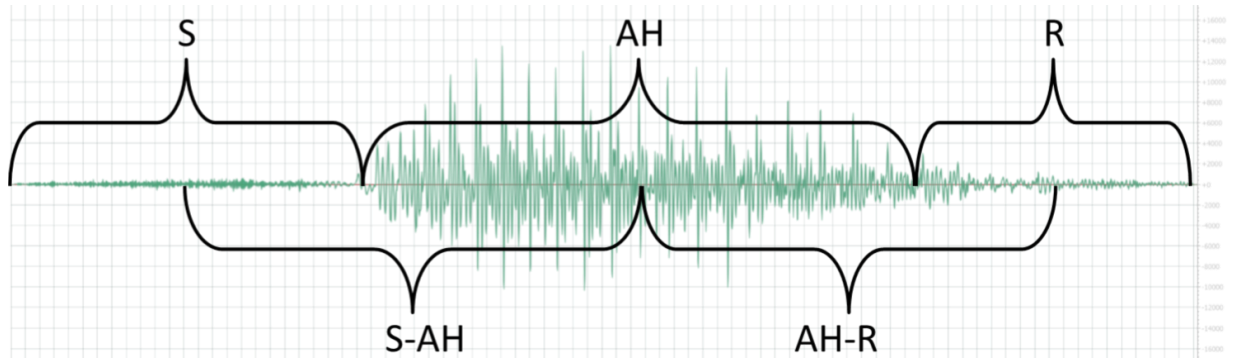


Figure 1: The sound wave of the word ‘sar’, with phones labelled at the top and diphones at the bottom

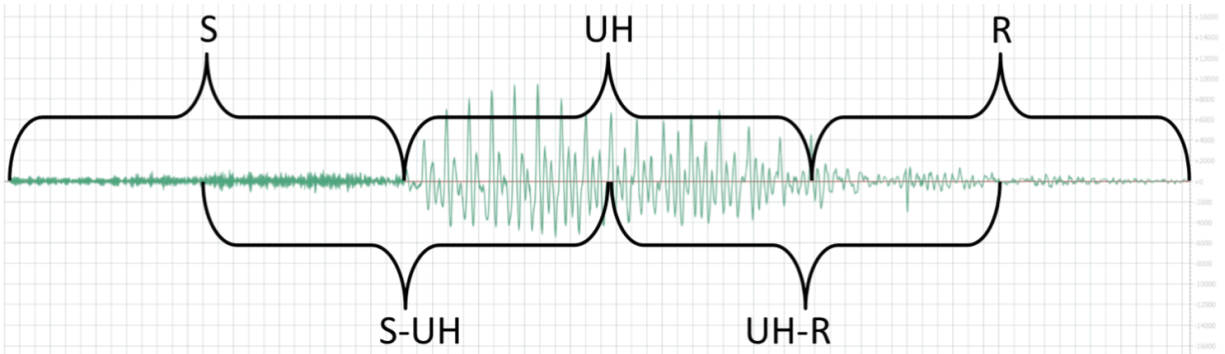


Figure 2: The sound wave of the word ‘sur’, with phones labelled at the top and diphones at the bottom

3 Literature Review

3.1 Review of Existing Technologies

The current state-of-the-art in speech synthesis lies in the field of Deep Neural Network-based, generative speech synthesis systems. One such example is Google DeepMind’s WaveNet, which is capable of producing superb quality speech. Such systems however are not feasible to use in this scenario given the hours of data needed for training (upwards of 50 hours of labelled speech data) [48, 51].

Unit Selection based Speech Synthesisers are known to obtain very respectable results, generating “natural-sounding synthesized speech” [24]. Two major types of Unit Selection systems exist, those with ‘Large Speech Databases’, containing multiple utterances for each unit [24], and those with a single instance of each unit [31]. Unit Selection Concatenative Speech Synthesisers using a Large Speech Database will have a very large database of speech

units (utterances), which gives the system a plethora of utterances to choose from, rather than just one utterance of a particular unit. Since such a large database of speech exists, it is much more often that a desired utterance with the correct and intended prosodic characteristics is found, without the need for waveform modification. Since the units chosen by the Viterbi algorithm will likely be as desired, they will require little to no signal processing, resulting in an output with correct intonation and thus a more natural sounding result [24]. The units are traditionally chosen by the Viterbi algorithm, however improvements of the system exist where the units are selected by a Hidden Markov Model (HMM) [52, 45].

It must be noted that no ‘large speech recording database’ for Maltese exists as of yet, which makes this type of system unfeasible for the Maltese language at this stage. There are techniques for building such large speech databases automatically from audio books. Such methods, however, still require a 1-2 hour speech database for initial training of the acoustic models used to extract the speech units from the audio books [37]. If such a database were to be obtained, there are 12 audio books in Maltese offered by Merlin Publishers⁴.

As opposed to the aforementioned solutions, ‘Concatenative Speech Synthesisers’ with single utterances of each unit require far less data, requiring less than 20 minutes of recorded speech, which then needs to be segmented. Such synthesisers have also, in their time, more than once been referred to as the state-of-the-art in high quality synthesis [20].

3.2 Choosing a Unit Size

In Concatenative Speech Synthesisers, a decision needs to be made on what unit size to use, be it a phone, diphone, half-phone or a syllable. Syllable units have been shown to produce the highest quality results, especially for languages that are ‘syllable centred’. Since syllables are longer than diphones or phones, more coarticulation is captured. Furthermore, there will be less points of concatenation for the same text, which will result in higher quality, more natural sounding text [27, 6]. This idea can be further extended to polysyllabic units, which display the same advantages over shorter units [49].

Using larger units comes at a cost of course, that is, the larger a unit is, the more distinct units are needed for complete coverage of a language. Solutions that combat this problem do exist, one of which is Approximate Matching of Units, however such a solution is meant to fill in gaps where a rarely used unit might be missing, and not for the majority

⁴<https://fitamalta.eu/maltese-audiobooks-by-merlin-publishers/>

of units [39]. Other solutions involve either finding the closest matching diphone (backing-off), or extending the boundaries of the adjacent two diphones to fill in the missing diphone [16, 26].

While ‘Global Speech Databases’ do exist, such as GlobalPhone [43], and units from one language can be used for similar sounding and related languages, if units from a language which is not very similar are used, the results will sound off and “silly” [7]. From the Maltese dictionary, 32.4% of words have Arabic origins, 52% have Sicilian/Italian origins and 6% have English origins [11]. It can thus prove quite challenging to pick and choose speech units from speech unit databases for different languages to sound natural when used for Maltese.

3.3 Converting Graphemes to Phonemes

One of the first steps in the Front-End of Concatenative Speech Synthesisers is the conversion of a body of text, formally a sequence of graphemes, into a “phonemic transcription” [10], or a sequence of phones that make up the sound of the text. This process is known as Grapheme to Phoneme (G2P).

G2P programs usually use a large set of G2P, letter to sound, rules to convert graphemes to phonemes. These rules are very context sensitive, i.e. their use depends not only on the grapheme itself, but also on the surrounding graphemes [36]. G2P implementations in commercial Text to Speech (TTS) systems usually contain a pronunciation dictionary of the most frequently used words, and then fall back to a rule based implementation for Out of Vocabulary (OOV) words [36].

A G2P program for Maltese was created by Crimsonwing for their implementation of a Maltese TTS system [10]. This program makes use of a ‘rule based’ approach, using over 100 Maltese grapheme to phoneme rules [36, 10]. This method is reported to achieve 98.5% accuracy when compared to human transcription [36]. Using such a rule based approach allows the G2P program to handle unseen, OOV words, something that a dictionary alone is not capable of.

Once the graphemes have been converted to phonemes, the International Phonetic Alphabet (IPA) phoneme symbols must then be converted to Advanced Research Projects Agency alphaBET (ARPABET) symbols. This step is required as IPA symbols are not “computer friendly” [29], and are thus represented in ARPABET by one or two upper case letters [29]. For example, the sound of the grapheme ‘x’ in ‘*xbiek*’ (a voiced palato-alveolar fricative) is represented as a ‘ɣ’ in IPA, but as a simple ‘ZH’ in ARPABET.

Unlike the English language, where multiple heteronyms exist, such as read (as in ‘I read books’ - present simple) and read (as in ‘I read a book yesterday’ - past simple), Maltese has a “low degree of heterography” [10], meaning very few words that are spelt the same are pronounced differently.

3.4 Generating Carrier Text and Recording Diphones

With this information, it is reasonable to choose to develop a Diphone Based Speech Synthesiser. A syllable is too long of a unit, as too many utterances need to be recorded for decent coverage of a language. On the other hand, there are 41 phones in the Maltese language, meaning no more than 1681 (41^2) diphones need to be recorded for complete coverage. Out of those, only about 1349 (around 80%) diphones are actually found in a Maltese text corpus [10].

When choosing a word from which a specific diphone will be extracted, it is important to, whenever possible, have that diphone in the middle of the word, since “it takes time for the human articulation system to start” [31]. There seems to be no clear cut answer on how the text which the speakers will read out should be generated [31]. The only hard requirement is of course that the “carrier words” are chosen such that each one contains the desired transition of a particular diphone [13].

One option is to have ‘free text’, which consists of natural sentences. One advantage of this is that prosody is captured in the extracted diphones, so the result should sound more natural [10]. Another option is to use ‘rainbow text’, which is manually prepared by an expert to have at least one instance of each diphone. Sentences are usually nonsensical for monotony and consistency [10]. Yet another option is to use computer generated nonsense words. This results in carrier words with the optimal diphone placement for best quality extraction. However, this requires expert phoneticians to read, who themselves may find it hard to utter some of the words [31].

By choosing to store a single utterance of each diphone, and thus recording a single carrier word, 1,349 words need to be recorded, 1619 counting any ‘pad words’. Even at a very slow speaking rate of 100 words per minute, recording these 1,349 diphones would take no longer than 17 minutes.

A. Stan et al. concluded that, to obtain the best quality when recording speech for use in synthesisers, speech should be recorded in a hemi-anechoic chamber, using high quality microphones recording at an oversampled rate of 96 kHz. The recordings should then be

downsampled to 48 kHz “for noise reduction”. The speech still retains a very good quality at 32 kHz [45].

Since the speaker’s voice may vary over one recording session, or if multiple recording sessions take place, it is important to normalise the recordings in some way. The Root Mean Square (RMS) power can be calculated over all the words, so that a “modification factor” is produced which can be applied to the words needing normalisation [31].

3.5 Diphone Extraction and Segmentation

The task of manually selecting the part of each carrier word’s waveform which correspond to a particular diphone by ear would take days, which gives rise to the need for an automatic labelling method, or at very least a tool which helps with efficiently and correctly segmenting the diphones.

WebMAUS Basic⁵ is a web app which, given an audio file, and a textual transcription of what is said in the audio file, will produce an alignment of the audio with the phonemes in the audio [42, 28], and can be graphically displayed in EMU-webApp⁶ [14]. This greatly facilitates the process of determining during which part of a carrier word a diphone occurs, by having an indication of what the phoneme boundaries are.

One of the earlier methods of automatic labelling was HMM based, however this was a supervised method, and needed a large speech database of pre-labelled units. A more modern spin on this method pairs HMMs with Support Vector Machines (SVMs), however still needs a manually segmented and hand-labelled speech database for training [22].

An alternative approach came about shortly after which does not necessarily require a pre-labelled database. This approach compares the recorded audio for a sentence in free text, with the audio output of another, already existing, concatenative speech synthesiser [33]. This comparison is done using the Dynamic Time Warping (DTW) algorithm. The DTW algorithm works very similarly to the Levenshtein String Distance algorithm, however measures the distance between any two time series, and computes the optimal global alignment of the two series [33]. DTW however, takes $O(n^2)$ (quadratic) time and space [41]. When taking into consideration the 1,600 diphones that need to be extracted, this becomes a very large and complex task. FastDTW is a near-optimal approximation of DTW which runs in linear time, i.e. has $O(n)$ time and space complexity, all the while producing results within 1% error of the optimal path [41]. An open-source python library

⁵<https://clarin.phonetik.uni-muenchen.de/BASWebServices/interface/WebMAUSBasic>

⁶<https://ips-lmu.github.io/EMU-webApp/>

exists which implements FastDTW, named ‘fastdtw’⁷ [2].

The DTW algorithm can be applied to automatically extract diphones from Maltese carrier phrases, using a pre-existing diphone database for English, such as FestVox’s ‘CMU US KAL Diphone’⁸ [8]. The quality of speech produced by an English diphone database will likely be inferior to a Maltese one, as Maltese and English are, of course, distinct, pronunciation-wise.

Multi-Layer Perceptron (MLP) based labelling methods also exist, however these are usually used as refinement on the borders of the segments, rather than as an initial pass [30]. Other methods include using a “syllable rate” timing (in the case of syllable based systems) to segment the recording, the output of which is compared to a rule-based or manually segmented signal, and the segmentation is refined using a HMM. This can be further paired with “Vowel Onset Point detection”, however this is primarily a syllable specific concept [25].

3.6 Concatenation of Diphones

One of the most important features of a good concatenation algorithm is the matching of “pitch, phase, amplitude, and frequency envelope” from one speech unit/diphone to the next for natural sounding speech [47].

The most efficient concatenation algorithms are ‘overlap-add methods’. Such methods work by selecting certain frames from the two diphones to be concatenated, processing those frames, and “recombining” them “with an OverLap Add (OLA) algorithm”, such that the pitches of the two diphones match up and thus, creating continuity. Time Domain - Pitch Synchronous OverLap Add (TD-PSOLA) is a widely used OLA method that produces good results while being notably computationally efficient [47].

Another concatenation algorithm is the Harmonic Plus Noise Model (HNM), which is a parametric method that produces higher quality recordings than TD-PSOLA, however, TD-PSOLA can be implemented in a much more reasonable time frame [46].

In order for OLA algorithms to work, the sound waves that are to be combined need to be pitchmarked [23]. Pitch marking marks a certain part of a wave, say the crest/peak, at every period. A pitch-period is thus the period of a wave between two consecutive pitchmarks. This is of very high importance for OLA algorithms as not only does it

⁷<https://pypi.org/project/fastdtw/>

⁸http://www.festvox.org/dbs/dbs_kal.html

specify which samples are to be manipulated (those at the beginning and end of each pitch-period), but also makes it possible to calculate the pitch/frequency of the wave at each pitchmark, using the formula:

$$f = \frac{1}{T}$$

Where f is the frequency/pitch and T is the period (time elapsed between one pitchmark and the next).

For speech, the optimal method for pitchmarking is through the use of an Electroglogtograph (EGG) [31]. An EGG is a specialised device that measures the contact between the vocal folds (the glottis) [32]. Data from the EGG is synced with the recorded audio so as to extract pitchmarks at the peak of each period [31]. While not ideal, pitchmarks can alternatively be extracted from waves using Pitch Detection Algorithms (PDAs) such as Auto-Correlation [19], ‘YIN’ [15] and McLeod Pitch Method (MPM) [35]. Praat⁹ is an open source program made to facilitate working with speech, which includes a PDA based on Auto Correlation which can be run via Praat scripts [9]. aubio¹⁰ is another open source audio toolbox which, amongst other features, has implementations of various PDAs including ‘YIN’ [12].

In TD-PSOLA, after pitch marking, each pitch-period is “windowed” (or ‘tapered’), such that the window is “centred on the region of maximum amplitude” and samples close to the centre of the window remain the same, whereas the amplitude of samples towards the edges of the window are scaled down the further they are from the centre [23]. Some popular window functions are Triangle, Hamming, Hanning and Blackman [18]. For speech synthesis, the Hann function, or the Hanning Window, is a popular choice [23].

Once all the pitch periods have been extracted from a wave and tapered, they can be reconstructed into a single wave by moving each pitch-period making up the wave closer together creating an overlap (to increase the wave’s pitch), or further apart from each other by adding a short pause between the pitch periods (to decrease the wave’s pitch). By how much the pitch-periods are moved depends on the target pitch. The target pitch is usually the pitch of the preceding wave, so that there are no jumps in pitch, resulting in a smoother transition [23].

⁹<http://www.fon.hum.uva.nl/praat/>

¹⁰<https://aubio.org/>

3.7 Prosody Modification

Prosody Modification can be carried out on the output waveform in order to add intonation to synthesised speech, make speech sound more natural and even possibly add hints of emotion. Prosody is made up of microprosody and macroprosody. Microprosody refers specifically to the prosody of a basic “individual speech sound”, whereas macroprosody refers to the prosody or intonation present over a larger body of text, as applied by the speaker [17].

DTW is one technique which can be applied to the output speech waveform to modify the prosody of the synthesised text. In order to make the algorithm faster and require less memory, instead of using DTW on the amplitudes of the speech signals, it can be used on the Mel-Frequency Cepstral Coefficients (MFCCs) of the “speech frames”. This method has been shown to be very effective with simpler intonation tasks, such as making a sentence declarative or interrogative. It was discovered that intensity, pitch and phoneme duration are the major features which influence prosody, whereas intensity has a relatively minor influence on the macroprosody [17].

3.8 Front-End and Handling Non-Standard Words

Text Normalisation, or Non-Standard Word (NSW) processing, deals with two major problems, detection of NSWs in text, and transformation of NSWs into a sequence of graphemes which can then be handled by the G2P component [5]. It also deals with other problems such as whether to pronounce the number 3 as a cardinal or an ordinal number, i.e. whether to say ‘*tlieta*’ or ‘*tielet*’ [5]. This process of Text Normalisation is regarded as one of the greatest challenges when implementing a speech synthesis system for a new language [45]. This problem is even more present in Maltese where the number 5 can be read as ‘*ħamsa*’ or ‘*ħamest*’ when used as a cardinal number.

One method to build a front-end is to go through the input text, applying G2P rules on standard words. When a NSW is met, it is first classified using a classification tree to determine its type, and then it is normalised/expanded according to the ‘procedure’ defined for that subclass [5]. These procedures are very often rule based and hand written, however this is often not sufficient. A different, more novel, approach for detection and handling of NSWs exists, that is, through use of language models [44].

3.9 Evaluation

There doesn't seem to be a 'de facto' way to evaluate TTS systems, and this is understandable as the quality of the result cannot be directly quantified, rather it is usually a qualitative evaluation that can be gathered. Nevertheless there do exist several quantitative evaluations, however most times it simply stems from a score given by the evaluator, which is quite subjective [4].

The evaluation can be focused on one specific feature or aspect of the TTS system. For example one can choose to ask for evaluators' score of the pronunciation of the system, ignoring other aspects such as the naturalness, speed or quality [3]. This method would make sense if the focus of this thesis was to compare how different algorithms affect pronunciation, however this is not the case.

The Diagnostic Rhyme Test (DRT) is a test for intelligibility. The DRT plays two pairs of words that are the same, except for a single consonant, to the evaluator, who must choose which word is the correct one. The Categorical Estimation test (CE) evaluates naturalness. The CE test asks evaluators for a score of factors such as the speed, pronunciation, quality and speed. These tests also uncovered that evaluators achieved better scores in DRT in subsequent tries, as they get accustomed to the synthesised voice [40].

The method that evaluates best what this thesis aims to achieve is very similar to the CE. Evaluators are played a number of synthesised texts and they are told to score the audio on criteria such as "intelligibility, naturalness and overall voice quality" on a scale of 1 to 5 [50].

In this section, the various types of TTS systems were reviewed, with a focus on their feasibility as a solution for Maltese. Following that, all the components of Concatenative Speech Synthesisers were studied in depth; namely unit sizes, text normalisation, G2P, building diphone databases and diphone concatenation. Finally, several methods for evaluating TTS systems were reviewed on how well they evaluate what this dissertation is trying to accomplish.

4 Implementation

Given the possible solutions for Text to Speech (TTS) systems and their requirements, the solution that is most feasible and sensible given what resources are currently available for the Maltese Language would be a Diphone-Based Concatenative Speech Synthesis system.

The block diagram in Figure 3 graphically explains the components involved in the TTS system and the flow of data.

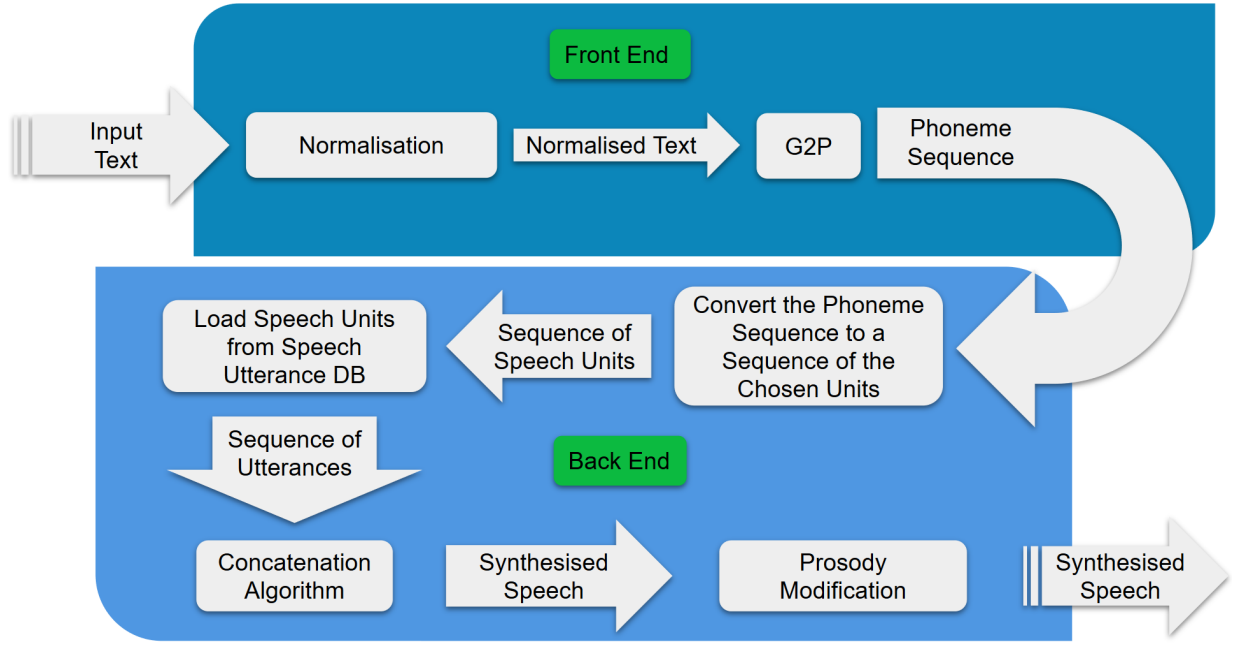


Figure 3: Block Diagram with all the components involved in the entire TTS system

The speech synthesis system has two principal components as follows: The front-end, or the Natural Language Processing (NLP) component, handles the user's raw input text and prepares it for the back end.

1. The front-end first scans the input for any Non-Standard Words (NSWs), i.e. text which the Grapheme to Phoneme (G2P) process is not natively capable of converting into phonemes, and replaces them with their textual equivalent. This step is called 'Text Normalisation' and its responsibility is converting non-word tokens (e.g. *hemm xi 15*) to their textual form (e.g. *hemm xi ħmistax*).
2. After that is done, the G2P process is run, which converts the normalised text from the previous step into a phonemic transcription. So for example, '*dan hu eżempju*' is converted to '*dən u ɛzempju*'. This sequence of phonemes is then passed to the

back-end of the speech synthesis system.

The back-end, or the Digital Signal Processing (DSP) component, converts the sequence of phonemes (or in this case, diphones), into a sequence of sounds making up the intended speech waveform.

1. The back-end first converts the phonemic transcription from a sequence of International Phonetic Alphabet (IPA) symbols (e.g. ‘dɛn ʊ ɛzɛmpjʊ’) into a sequence of Advanced Research Projects Agency alphaBET (ARPABET) symbols (e.g. ‘[[D, AH, N], [UH], [EH, Z, EH, M, P, Y, UH]]’).
2. The sequence of phonemes is then converted into a sequence of the chosen units, in this case from a sequence of phonemes to a sequence of diphones (e.g. ‘[[D-AH, AH-N, N-#], [UH-#], [EH-Z, Z-EH, EH-M, M-P, P-Y, Y-UH, UH-#]]’). The ‘#’ symbols have been added as silence symbols, indicating a pause.
3. All the diphones from the previous step are loaded from the Diphone Database into memory.
4. The Diphone Concatenation algorithm then handles smoothly joining these diphones, such that their transitions sound seamless. At the end of this step, the speech waveform is generated.
5. Finally, Prosody Modification may be applied to the speech waveform to add intonation or expression/emotion to the synthesised speech. This will not, however, be implemented in this system.

4.1 Text Normalisation - Non-Standard Word Handling

Text Normalisation is the first component of the speech synthesis system. The aim of this step is to convert the user’s raw, text input into a format which can be handled better by the G2P component. As will be explained in Section 4.2, the G2P component can deal with any string of words, however, is unable to correctly handle NSWs, such as the following:

- numbers in their numeral form - e.g. 190
- dates and time - e.g. 27/04/2019 18:00
- abbreviations - e.g. UE (Unjoni Ewropea) or MT (Malta)
- symbols and measurements - e.g. €50 or 25 km

The job of Text Normalisation is to convert a NSW such as ‘1984’ to its textual form, in this example, *elf disa’ mija erbgħa u tmenin*. Handling of NSWs is not the main focus of this thesis, and thus, out of the mentioned NSWs, only handling of integer numerals was implemented. Handling of other NSWs can easily be added in the future, as will be

explained in this section.

Once the user's input has been received by the system, the Front-End splits the input text into tokens by spaces. Each token is then sent to the Text Normalisation component. This component first determines what type of token it is dealing with. For example, if all the characters of a token are digits, then the token is an integer. Further checks could be added at this stage, as other NSW handlers are added. If the token is not a NSW, then it is returned the same way it was received. If the token is determined to be a NSW, then its appropriate normalisation procedure is called.

Handling of integers between 0 and 9,999 was implemented. The normalisation procedure is as follows:

1. If the number is between 0 and 9, then another function that deals with units is called. This function maps each unit digit to its textual form - e.g. 9 is converted to *disgha*.
2. If the number is between 10 and 99:
 - (a) The tenth and unit are extracted from the integer.
 - (b) If the number is less than or equal to 19, it is converted automatically to its textual form by the function that deals with tenths - e.g. 17 is converted to *sbatax*.
 - (c) If the unit of the number is 0, it is converted automatically to its textual form by the function that deals with tenths - e.g. 40 is converted to *erbgħin*.
 - (d) If the number is any other number, the function is called recursively with the unit. The output of that is concatenated with an ' u ' and the tenth which is converted to its textual form by the function that deals with tenths - e.g. 25 is converted to *ħamsa u ghoxrin*.
3. If the number is between 100 and 999:
 - (a) The hundredth, tenth and unit are extracted from the integer.
 - (b) If the unit of the number is 0, the textual form of the number is:
the textual form of the hundredth + ' u ' + the textual form of the tenth - e.g. 450 is converted to *erba' mija u ħamsin*.
 - (c) If the number is any other number, the textual form of the number is:
the textual form of the hundredth + ' ' + the return of the recursive call to the function without the hundredth - e.g. 567 is converted to *ħames mija sebgħa u sittin*.
4. If the number is between 1000 and 9999:

- (a) The thousandth, hundredth, tenth and unit are extracted from the integer.
- (b) The textual form of the number is:
the textual form of the thousandth + ‘ ’ + the return of the recursive call to
the function without the thousandth - e.g. 2345 is converted to *elfejn tliet mija
ħamsa u erbħin*.

Due to the implementation of the G2P program used, Out of Vocabulary (OOV) words do not need to be handled during Text Normalisation, as will be explained in Section 4.2.

4.2 G2P - Converting Graphemes to Phonemes

This subsection deals with converting any given body of text, formally a sequence of graphemes, into a “phonemic transcription” [10], or a sequence of phonemes. This process is known as G2P.

In order to implement G2P for this implementation of a TTS system, Crimsonwing’s G2P program was used. This G2P program, given any text file of Maltese words, will return another text file where each word is replaced by its phoneme sequence, represented in IPA symbols. Crimsonwing’s G2P program uses a rule based approach, using letter-to-sound rules [10], as opposed to a pronunciation dictionary. Since a rule-based approach does not need a database of known words, it can handle OOV words, thus eliminating the need for them to be normalised in the ‘Text Normalisation’ step. The python script ‘MalteseG2P’ was written which writes the user’s input (graphemes) to a text file, runs the Crimsonwing G2P program on the text file and reads the output (phonemes) from the output text file. Once the graphemes have been converted to phonemes, the IPA phoneme symbols are read, and converted to ARPABET through the use of an IPA:ARPABET symbol dictionary. Only the IPA to ARPABET conversions for phonemes present in the Maltese language were implemented, as specified by Farrugia [21].

4.3 Diphone Database

Next, a Diphone Database was created. This stores a recording of almost every possible Diphone, and will consist of at most 1,681 diphones, of which around 1,450 are actually present in spoken Maltese.

4.3.1 Generating Carrier Material

A Python function was written which, given the list of Maltese phonemes, returns a list of all diphones in Maltese. This list of diphones is then passed to another function which finds a carrier word for each diphone in the list.

This function first shuffles the list of diphones, and then generates 12 word carrier phrases as follows:

1. The first word of each phrase is a random pad word, which is there to allow for articulation to start.
2. Next, a diphone is popped from the list of diphones, and a carrier word is found for it using the ‘find_carrier_word_for_diphone’ function (This function will be described after the end of this list).
3. The last word is again set to a random pad word.

This method ensures that there will always be at least one recording of each diphone, and the randomness ensures that the carrier material will be nonsensical, and will thus be read in a monotonous and consistent voice, thus eliminating unwanted intonation and expression in speech.

A python script called ‘MLRSCorpusToWordPhoneme.py’ was written. This function first goes through the entire MLRS corpus¹¹, and extracts every unique Maltese word, resulting in around 700,000 words. Each word is then converted to a phonemic transcription using the G2P function as described in Section 4.2. Next, the IPA symbol phonemic transcriptions are converted to lists of ARPABET symbols. Finally, the 700,000 ‘word, phoneme(IPA), phoneme(ARPABET)’ pairs are written to disk as a .csv file. The aforementioned function which finds carrier words for diphones looks through the phonemic transcriptions in this .csv file to find a suitable carrier word for any given diphone.

Once all the carrier phrases have been generated, they are written to disk in two ways:

1. As a single text file with the index of the phrase, the carrier phrase and the diphone that will be extracted from each carrier word, as shown in Figure 4. This format is useful for when the carrier phrases are being recorded as all the information is in one place.
2. As a series of ‘transcript’ text files, where each transcript file stores the carrier phrase that will be recorded in the associated audio file, as shown in Figure 5. This format is useful for when the recorded carrier phrases are uploaded to the ‘WebMAUS Basic’ forced alignment tool; this will be explained further in Section 4.3.3.

¹¹<http://mlrs.research.um.edu.mt/index.php?page=corpora>

```

1 0
2 ['pad', 'Y-UH', 'M-N', 'L-#', 'N-IH', 'N-#', 'M-EH', 'IH-M', 'AH-N', 'AH-F', 'IH-T', 'pad']
3 ['nifxu', 'bjut', 'tomna', 'skrapel', 'knisja', 'skrun', 'tithemmedx', 'qadima', 'staneg', 'hra']
4
5 1
6 ['pad', 'P-Y', 'Y-AH', 'HH-EH', 'S-EH', 'T-K', 'M-P', 'K-EH', 'W-IH', 'SH-R', 'IH-L', 'pad']
7 ['ssaġġarniex', 'tipjip', 'fwejjah', 'nithemmdu', 'hsejjes', 'nitkafkaf', 'ċenpula', 'rkejjen']
8
9 2
10 ['pad', 'IH-#', 'AO-M', 'D-IH', 'M-#', 'AO-SH', 'Y-IH', 'EH-M', 'IH-HH', 'IH-S', 'K-#', 'pad']
11 ['jitkarmsu', 'lbiesi', 'tomna', 'nadif', 'ċrum', 'qoxra', 'nkoljix', 'denb', 'rejħiet', 'knis']
12

```

Figure 4: Carrier Phrases that will be used during recording

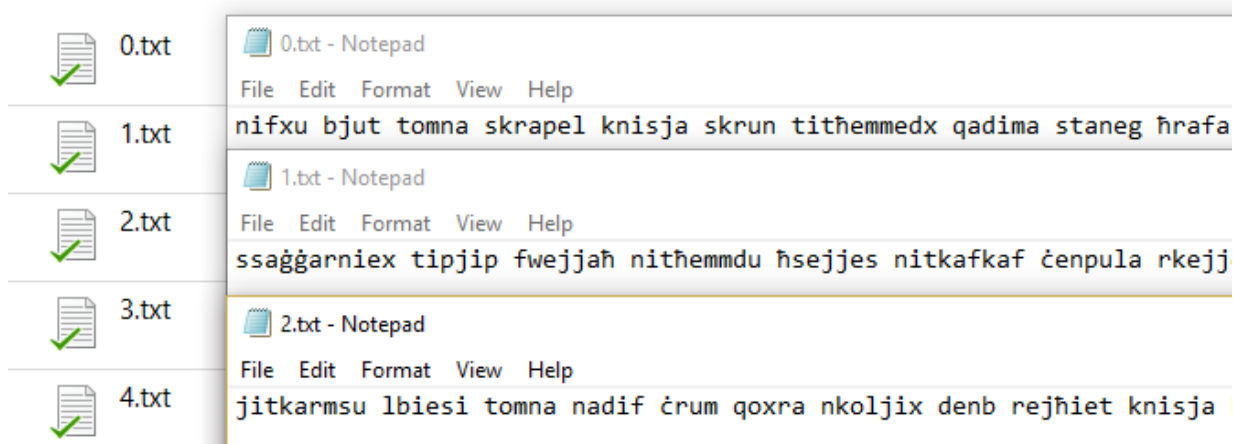


Figure 5: Visual representation of Phones and Diphones

4.3.2 Recording Carrier Material

Recording was carried out in a very quiet room, after midnight, to minimise the background noise from the street such as traffic, construction and other people. The only device present in the room was the smartphone used for recording the phrases. A smartphone is ideal because modern phones come equipped with a rather high quality microphone. They also have a touch screen, which minimises clicks from a traditional mouse. The phone was configured to record .wav files at 16kHz, and to incrementally name audio files starting from 0 and counting up. The generated list of all carrier phrases was printed. This was chosen over reading the prompts from a display so as to minimise any electronic hum. The speaker was seated at about arms length from the recording device, so as not to be speaking directly into the microphone to avoid pops in the audio. The paper with the prompts was stood upright on the table, so as to eliminate any noise that might come from fumbling the pages.

During recording, the speaker was instructed to keep a constant pose throughout, to

stay the same distance apart from the microphone. They were also instructed to give a slight pause after pressing record on the touch screen, and before pressing ‘stop recording’ after finishing each phrase. They were told to, most important of all, speak loud and clear, pay attention to enunciating every phoneme present in the word and to give adequate pauses between each word.

4.3.3 Diphone Extraction

Once the carrier phrases had been recorded, the recordings were segmented to isolate the diphones they were meant capture. The diphones are extracted using two distinct methods as follows, thus creating two separate diphone databases.

Manually Extracting Diphones The first method is extracting the diphones manually, using the ‘WebMAUS Basic’ web service for audio to phoneme forced alignment, together with EMU-webApp for visualisation of the audio aligned with the phonemes. The recorded carrier phrases, together with their associated transcript are uploaded to WebMaus Basic, as shown in Figure 6. The language is set to Maltese, and the service is run. A ‘TextGrid’ file is generated for each pair of audio and transcript files. These TextGrid files are then opened in EMU-webAPP. As shown in Figure 7, this interface shows the start time, duration, and end time of each phoneme in the audio. While these are not always perfect, and require some fine-tuning, this combination of WebMAUS Basic and EMU-webApp prove very helpful in determining the boundaries of each diphone.



Figure 6: Uploading Recorded carrier phrases (.wav) and their transcripts (.txt) to WebMAUS Basic for forced alignment of phonemes

For vowels the midpoint of each phoneme is used as the boundary of the diphone, whereas for consonants, the closure is preferred. As each diphone boundary was deter-

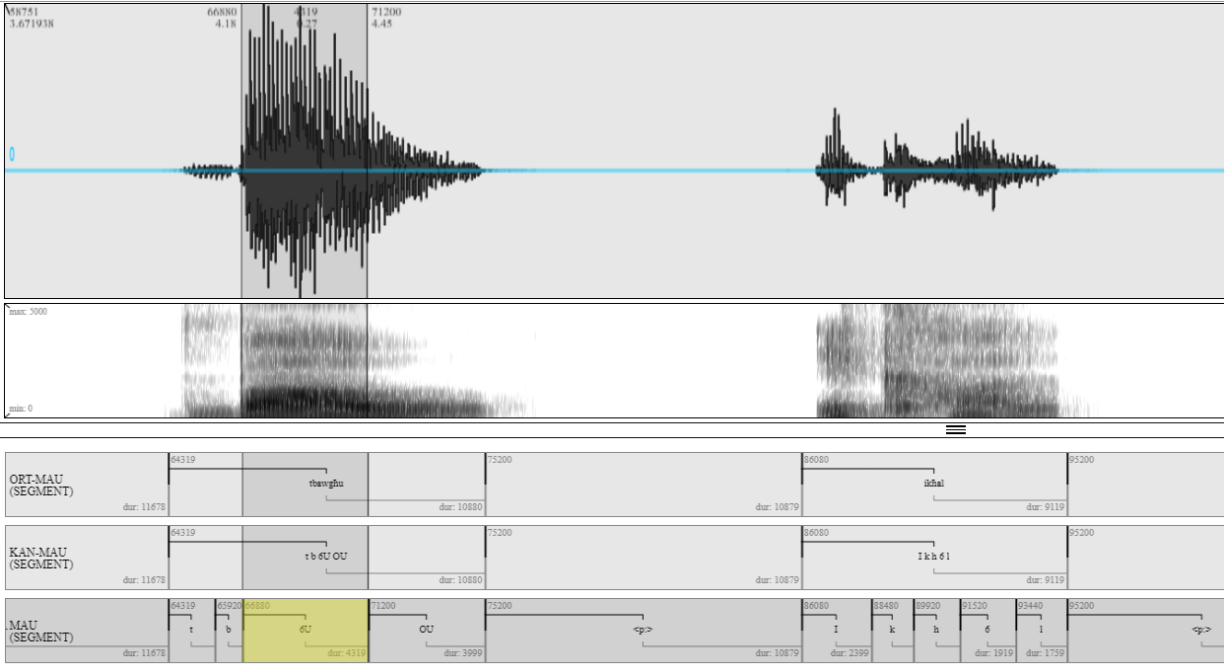


Figure 7: The result of WebMAUS Basic's Forced Alignment tool shown in EMU-webApp

mined, a text file was filled in with the following details:

- The diphone that will be extracted (e.g. Y-UH)
- From which .wav file (carrier phrase recording) the aforementioned diphone will be extracted (e.g. 0.wav)
- The timestamp of the start boundary of the diphone in seconds (e.g. 1.85)
- The timestamp of the end boundary of the diphone (e.g. 1.99)

An example of this is shown in Figure 8. This process of precisely determining diphone boundaries was found to take 20 to 25 minutes per carrier phrase of 10 diphones. As more diphone boundaries were determined, the process becomes progressively easier, as the part of a wave associated with a particular phoneme becomes more familiar. The waveforms for 't' and 's', for example, are quite recognisable, as are the waveforms for vowels and 'b' and 'p', amongst others. Being able to recognise what part of the wave exactly corresponds to a phoneme speeds up the process of fine-tuning the forced alignment, and thus speeds up the boundary determination process. A python script was created which takes this specification text file as input, loads the specified carrier phrase, extracts the audio between the specified time stamps, and saves it to disk in the diphone database as per the specified name.

In total, 58 diphones were extracted, enough to synthesise a few phrases to be able to evaluate the system. To record all the theoretically possible 1,681 diphones in Maltese,

1	Y-UH	0	1.85	1.99
2	M-N	0	3.21	3.32
3	L-#	0	4.80	4.91
4	N-IH	0	5.48	5.57
5	N-#	0	7.05	7.17
6	M-EH	0	8.06	8.21
7	IH-M	0	9.23	9.36
8	AH-N	0	10.28	10.37
9	AH-F	0	11.40	11.54
10	IH-T	0	12.33	12.40
11				
12	P-Y	1	3.22	3.34
13	Y-AH	1	4.55	4.65
14	HH-EH	1	5.44	5.53

Figure 8: Diphone list specifying how diphones are to be extracted from carrier phrases

assuming 2.5 minutes per diphone (rounding up), it would take about 70 hours to have complete coverage of Maltese. Alternatively, the output of the forced alignment tool can be used directly without refinement, however this will likely produce rather poor results.

Automatically Extracting Diphones using Dynamic Time Warping (DTW)

The second method uses DTW to automatically extract a diphone from its carrier word. The ‘CMU US KAL Diphone’ database will be used as a comparison for aligning, since there is currently no available Maltese diphone database. The python library ‘fastdtw’ [2] was used for an $O(n)$ (linear time and space) implementation of DTW.

A python script was created which goes through the CarrierPhrases.txt file, such as that shown in Figure 4, which contains a list of all carrier phrases and the list of diphones contained in each carrier phrase. The script extracts the name of the recording that contains the carrier phrase, e.g. 0.wav, and the diphones in that phrase, e.g. [Y-UH, M-N, L-#, ..., AH-F, IH-T]. The path of the carrier phrase is determined from the name of the recording. A loop then goes over every diphone in the phrase. FastDTW is then applied to the entire carrier phrase and the current diphone, but from the CMU US KAL diphone database. FastDTW takes no more than 20 seconds to extract a diphone from a 15 second carrier phrase, whereas standard DTW takes more than 3 minutes, sometimes crashing due to not having enough memory. Finally, the path generated from DTW is read, and applied

to the carrier phrase to extract the diphone, which is saved to disk in an appropriate folder, as its name (e.g. Y-UH.wav).

The quality of the synthesised speech when using the ‘CMU US KAL Diphone’ database, the manually extracted diphones and the DTW extracted diphones will be evaluated, serving as a comparison between which method works better for Maltese given what is available.

4.4 Concatenation of Diphones

Once the phonemic transcription has been generated from the G2P step, this is converted into a sequence of diphones. The specified diphones are then loaded from the diphone database into memory.

Diphone Concatenation, as the name suggests, deals with concatenating the individual diphones to form the target speech, as a single audio wave. The concatenation algorithm is of course not as simple as simply sticking one diphone after the other, some form of processing must be carried out so that the flow of the audio is smooth, and there are no noticeable jumps in pitch, intonation or volume in the speech.

While several methods exist for concatenation, Time Domain - Pitch Synchronous OverLap Add (TD-PSOLA) was implemented as it is known to reliably produce satisfactory results while being both a lightweight algorithm and relatively simpler to implement than other parametric concatenation algorithms [46].

The first step in the TD-PSOLA algorithm is pitchmarking all the waves to be concatenated. Two distinct methods were implemented that solve this task.

The first method involves calling ‘aubio’, an open-source library, with the file path of the .wav file and the pitchmark detection algorithm as parameters. ‘YIN’ was chosen as the pitchmark detection method.

The second method uses ‘Praat’, another open-source project, and calls a script written in Praat scripting language that calculates the pitchmarks of the wave passed as a parameter. The output is stored in a text file, from where it is loaded back into the program. As opposed to the aubio implementation, Praat uses Auto-Correlation.

While ‘YIN’ is an improvement over the older Auto-Correlation method, from a visual inspection, Praat seemed to produce better results than aubio. This may be due to the fact that Praat’s parameters are fine tuned towards speech, whereas aubio has a more general use case. Both programs, apart from the timestamps of each pitchmark, also return the frequency of the wave at each pitchmark.

The pitchmarks extracted from either aubio or Praat will be timestamps. In order to easily manipulate the waves from Python, these values are converted to samples. This conversion is easily computed using the following formula:

$$sample = time \times sampling\ rate$$

Following that, a ‘two-period window’ is extracted from every wave. These two-period windows can be used later to compensate for a loss in duration when overlap adding. This is simply done by selecting a pitchmark from the middle, and extracting two periods/cycles (i.e. until the pitchmark after the next).

The frequency of the first wave to be concatenated is calculated. The frequency of a wave is calculated by averaging the frequency at each pitchmark as calculated by Praat or aubio. All subsequent waves must approximately match this frequency in order for the transition between one diphone and the next to sound seamless. A loop then goes over every remaining diphone to be concatenated, and does the following:

1. The diphone is segmented into smaller waves, according to its pitchmarks.
2. Each ‘sub-wave’ from the previous step is tapered using a hamming window.
3. The frequency of the diphone is calculated.
4. The periodic times of the first wave and the current diphone are calculated. This is done using the formula

$$T = \frac{1}{f}$$

where T is the periodic time and f is the frequency of the wave

5. The periodic times are each converted into a number of samples. This is again done using the formula

$$sample = time \times sampling\ rate$$

6. The difference in periodic time between the two waves (in samples) is computed;
 - If the first diphone’s frequency and the current diphone’s frequency are equal, then they are simply concatenated together.
 - If the first wave’s frequency is lower than the current diphone’s frequency, zeros are added between each period of the current diphone to lower its frequency and thus match it to that of the first wave, as shown in part (b) of Figure 9.
 - If the first wave’s frequency is higher than the current diphone’s frequency, OverLap Add (OLA) is carried out on each period of the current diphone to raise its frequency and thus match that of the first wave, as shown in part (a) of Figure 9.

When adding zeroes between waves: The number of zeroes (in samples) that are added between each period (sub-wave) is the difference between the periodic times of each wave (in samples).

The modified diphone is computed by adding periods (sub-waves) from the original diphone and the pad of zeroes after them one by one, until the final number of samples is within ‘half a period worth of samples’ of the original diphone. This is done to preserve the original duration of the diphone.

When doing OverLap Add (OLA): The number of samples that are overlapped between each period (sub-wave) is the difference between the periodic times of each wave (in samples).

The overlap added diphone is computed as follows:

1. The final diphone is initially set to the first period of the diphone.
2. The number of samples that are to be overlapped are removed from the end of the final diphone, and stored elsewhere temporarily.
3. The same number of samples are removed from the beginning of the next period of the diphone, and again stored elsewhere, temporarily.
4. An element-wise summation of the two temporary arrays is computed.
5. The final diphone is set to the final diphone (from the last iteration) + the summation + the next period; ‘+’ here refers to concatenation.
6. The process from Step 2 to Step 5 iterates until all periods have been processed.

The number of samples that were lost during the OLA process is calculated as the difference between the initial number of samples and the number of samples in the overlap added wave. The number of ‘two period windows’ that need to be added to the beginning of the modified diphone is calculated by dividing the number of samples lost by the length of the two period window for the diphone. The two period window for the current diphone is tapered and prepended as many times as specified by the previous calculation. This is done to maintain the original duration of the diphone.

In this section, the design of the developed TTS system was first described at a high-level. A Diphone-Based Concatenative Speech System was chosen as the most appropriate solution, using the TD-PSOLA algorithm as the diphone concatenation algorithm. The implementation of each individual component was then described in detail, and a detailed procedure of how a diphone database for Maltese was built from scratch was provided.

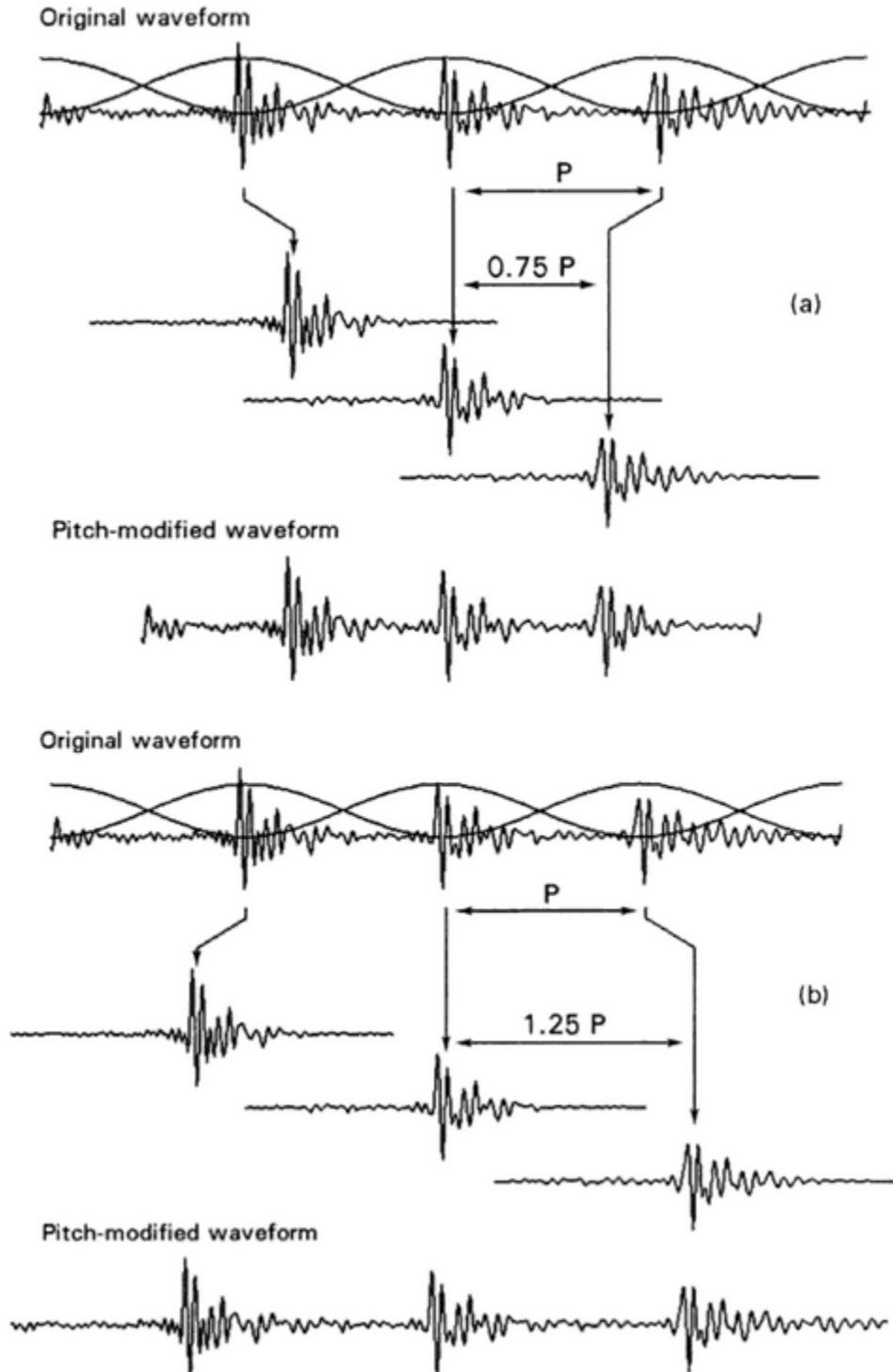


Figure 9: A graphical explanation of the TD-PSOLA algorithm showing the pitch being increased i.e. the period being reduced in (a) and the pitch being lowered i.e. the period being increased in (b). Reproduced from [23].

5 Evaluation

Since 3 separate Diphone Databases were built and made available to the speech synthesiser, the performance of each individual database was evaluated. A diphone concatenation algorithm (Time Domain - Pitch Synchronous OverLap Add (TD-PSOLA)) was also implemented, and evaluated. Finally, a simple Text Normalisation algorithm was created for Maltese, which was also tested and evaluated.

5.1 Evaluation Plan

Since the developed speech synthesiser is made up of many components, these were evaluated individually, then also as part of a complete system.

5.1.1 Evaluating the Diphone Databases and the Overall Performance of the Synthesiser

The three diphone databases (DBs) that the developed speech synthesiser is capable of using are:

1. The FestVox English ‘CMU US KAL’ Diphone DB [8]
2. The Diphone DB built by using Dynamic Time Warping (DTW) with ‘CMU US KAL’ to automatically extract diphones from Maltese carrier phrases.
3. The Diphone DB built by manually extracting diphones from the Maltese carrier phrases with the help of ‘WebMAUS Basic’ and ‘EMU-webApp’.

The ‘CMU US KAL’ Diphone DB was primarily used as a benchmark database. If one of the diphone databases extracted from Maltese carrier phrases is found to outperform the US diphone set, that would mean that the process outlined in Section 4.3 produces high quality diphone databases, and thus is capable of synthesising speech at a good quality.

The following four phrases were arbitrarily chosen, and synthesised from each of the three diphone databases:

1. *Il-kompjuter jaf jitkellem.*
2. *Jien għandi 21 sena’.*
3. *Jien jisimni Daniel!*
4. *Int x’jismek?*

A survey based on the Categorical Estimation test (CE) as described by Rashad et al. in [40] was created as follows:

1. Phrases 1 and 2 synthesised using all 3 databases (referred to as voices within the survey to avoid technical terms) with the question shown in Appendix A - The Evaluation Survey - Figure 10 for each speech sample.
2. Phrase 3, again synthesised using all 3 databases, with the question in Appendix A - The Evaluation Survey - Figure 11 after each speech sample.
3. After 3 Phrases synthesised from each diphone database were presented, the questions shown in Appendix A - The Evaluation Survey - Figure 12 was asked.

This question was asked to allow users to choose which database they feel produces the best sounding speech without having to think about specific features such as intelligibility, clarity and naturalness. Furthermore, it allows users to give a qualitative evaluation of the different diphone databases.

The Survey was created using Google Forms¹² and is available at <https://forms.gle/tPChMGpj2ReLP81E9>; excerpts of which are shown in Appendix A - The Evaluation Survey. The survey was shared primarily with university students, and also within Maltese Facebook groups. Wherever the survey was shared, respondents were asked to only fill the survey in if they are fluent in Maltese, to listen to the speech carefully and wear headphones if possible.

From the scores obtained in question for naturalness, clarity and intelligibility, a weighted average will be calculated for each diphone database for comparison. The overall score will be calculated as follows:

$$score = 3 \times intelligibility + 1.5 \times clarity + naturalness$$

The reasoning behind these weightings is that intelligibility is the main focus of this thesis, and should thus be the most significant determiner of the performance of the system. Clarity was also weighted slightly higher as synthesising clear speech is also a main aim of this work, and this factors in the effectiveness of the process for creating diphone databases described in Section 4.3. Naturalness was not weighted higher as it is not within the scope of this thesis. Furthermore, the work having to do with naturalness and flow/smoothness of speech, namely the TD-PSOLA algorithm, will be evaluated independently.

5.1.2 Evaluating the Diphone Concatenation Algorithm

One major component implemented for this Diphone-Based Concatenative Speech Synthesis system was the TD-PSOLA algorithm. TD-PSOLA was used to smoothly string

¹²<https://www.google.com/forms/about/>

together consecutive diphones, so that the transition between them is seamless.

In order to evaluate this, at the end of the survey mentioned in Section 5.1.1, the question shown in Appendix A - The Evaluation Survey - Figure 13 was asked. In this question, Phrase 4 is synthesised first using simple concatenation of audio waves, and then using TD-PSOLA, using 2 of the diphone databases. These pairs of speech samples are presented to the respondent, who is then asked whether they noticed any differences between them, and whether one sounded better than the other. After both pairs of synthesised speech were presented and evaluated by the respondent, they are asked for any further comments on the differences between the audio files.

The speech generated by simple concatenation serves as a benchmark for the speech generated by TD-PSOLA. If survey respondents score the speech produced by TD-PSOLA higher than the speech generated by simple concatenation, that would confirm that the implemented TD-PSOLA algorithm does, in fact, produce higher quality speech.

5.1.3 Evaluating the Front End

This section will focus on the evaluation of the Front End of the Text to Speech (TTS) system, namely the Text Normalisation component for Non-Standard Word (NSW) Handling and the Grapheme to Phoneme (G2P) component.

The Text Normalisation component can be evaluated by testing it with a number of examples of NSWs that the system should be able to handle, and others which it shouldn't be able to, and determining whether the text output is as expected. The implemented Text Normalisation component should be capable of handling all integers between 0 and 9,999. Handling for other NSWs has not been implemented. Out of Vocabulary (OOV), on the other hand, should be handled by the G2P component.

Similarly, the G2P component can be evaluated by testing it with a variety of Maltese text, trying edge cases to break it.

5.2 Evaluation Results and Discussion

5.2.1 Text Normalisation and G2P

The 'Text Normalisation' component was tested with the following inputs:

- '0', returned '*żero*', as expected
- '9', returned '*disgħa*', as expected
- '33', returned '*thieta u tletin*', as expected

- ‘30’, returned ‘*tletin*’, as expected
- ‘123’, returned ‘*mija tlieta u ghoxrin*’, as expected
- ‘4567’, returned ‘*erbat elef hames mija sebgħa u sittin*’, as expected
- ‘9999’, returned ‘*disat elef disa mija disgħa u disgħin*’, as expected
- ‘10000’, returned ‘’(nothing), as the component can currently only handle integers between 0 and 9,999.
- ‘9,999’, returned ‘9,999’, as the component does not currently recognise this as an integer.
- ‘-1’, returned ‘-1’, as negative numbers are not currently handled
- ‘9.0’, returned ‘9.0’, as decimals are not currently handled
- ‘25!’, returned ‘25!’, as this is not considered an integer, and thus not handled
- ‘50cm’, returned ‘50cm’, as the component does not currently handle units

In short, the system can successfully recognise and handle integers between 0 and 9,999. The integers must be separated from other words and punctuation (by a whitespace). Dates, time, abbreviations, symbols and measurements are currently not handled.

The ‘G2P’ program developed by Crimsonwing was also tested and evaluated. Some notable test cases were the following:

- Inputting ‘*sur*’ always returns “S UH R”, which translates to ‘mister’, but never “S UW: R”, which reads as ‘fortified wall’. This is because the G2P program is unable to extract the sense of a word, and thus its correct pronunciation.
- Inputting any word starting with the letter ‘gh’, such as ‘*ghandek*’, alone will return “K HH AH: N D IH”, incorrectly adding a ‘K HH’ sound to the beginning of the word. This error does not occur when the word is used in a sentence and “AH: N D IH” is correctly returned.

5.2.2 Diphone Databases and Overall Synthesis

The results of the survey which pertain to the evaluation of the diphone databases and overall synthesis quality are shown in Table 1. This table was built by first assigning a numerical score to each result - 1 to ‘Not at all’, 2 to ‘Not very’, 3 to ‘Somewhat’, 4 to ‘Quite’ and 5 to ‘Very’. Each cell of Table 1 was then filled by finding the average score assigned to the criteria of that cell’s row for the diphone database of that cell’s column.

From Table 1:

- The FestVox English ‘CMU US KAL’ Diphone DB [8] (DB1) obtained a rather poor score for naturalness, however a decent score for clarity and intelligibility. The score

Table 1: The results of the evaluation survey

	Voice 1	Voice 2	Voice 3
Naturalness	1.81	1.13	2.57
Clarity	2.69	1.26	2.72
Intelligibility	2.44	1.04	3.06
Overall Score	13.18	6.12	15.82

for clarity can be explained due to this database being professionally recorded in a studio. The score for Intelligibility is also quite respectable, given that this is an English diphone database and not a Maltese one.

- The Diphone DB built by using DTW on Maltese carrier phrases (DB2) performed terribly, as not one respondent was able to understand what was being said. This shows that the implemented DTW algorithm was not effective.
- The Diphone DB built by manually extracting diphones from the Maltese carrier phrases (DB3) scored the highest for each criteria. It was scored the most natural, significantly beating the English DB. Surprisingly, it also beat the English DB in clarity, albeit barely, despite not being recorded in a studio with professional equipment.

The fact that diphone DB 3 outperformed the English DB (DB1) in, not only intelligibility, but every other criteria, alone proves that the defined procedure for creating a diphone DB for a new language is, in fact, effective.

Furthermore, when asked which diphone database respondents thought sounds the best overall, more than 70% of respondents voted for diphone DB 3. The most common reasons given by respondents for their choices were (replies are paraphrased):

- I could understand every word - It had the best pronunciation
- Much clearer than the others, resembles a natural speaker of the language
- Easy to comprehend, sounded the most similar to human speech

When asked to transcribe Phrase 3, all respondents answered correctly. There were, however, a couple of remarks about the word “*jisimni*”. This type of feedback is very constructive as it signals that, perhaps, the diphones used to make up the word need their boundaries fine-tuned, or for the diphone to be re-recorded in a new carrier phrase.

5.2.3 Diphone Concatenation Algorithm

When asked to compare the speech generated using simple concatenation versus concatenation using TD-PSOLA, 53.7% of respondents said both audio files sounded exactly the

same, 35.2% said they preferred the speech generated by simple concatenation, while the remaining 11.1% said they preferred the speech generated when TD-PSOLA was used. The most common reasons given by respondents for their choices were (replies are paraphrased):

- They sound almost identical, however there is a slight difference; the words in v1 [the speech synthesised using simple concatenation] are more intelligible.
- The speech in v2 [the speech synthesised using TD-PSOLA] sounds slightly grainier.
- The speech in v2 [the speech synthesised using TD-PSOLA] has less background noise.
- The speech in v2 [the speech synthesised using TD-PSOLA] has more noise.

It is hard to draw a conclusion from the results obtained. Not only did more than half the respondents not manage to find any difference between the recordings, but there were quite a few conflicting opinions on which sounds better. Just by looking at the numbers and the majority of comments however, simple concatenation seems to have outperformed TD-PSOLA.

This section first provides a description of how the Front-End, the Diphone Concatenation algorithm, the quality of the Diphone Databases and the overall quality of the speech synthesised by the developed system were evaluated, that is, primarily by distributing a survey. The results from the described processes are then presented, interpreted and discussed. The Diphone Database built entirely as part of dissertation by manually extracting diphones from the Maltese carrier phrases achieved the best score for intelligibility, clarity and naturalness.

6 Conclusions and Future Work

A Diphone-Based Concatenative Speech Synthesiser for Maltese was successfully developed in this dissertation, receiving an average rating of 3.06 (meaning ‘somewhat’) for intelligibility.

The developed implementation can receive Maltese text from the user and deal with some Non-Standard Words (NSWs), such as integers between 0 and 9,999. The Grapheme to Phoneme (G2P) component, using Crimsonwing’s G2P program, converts any body of text into a phonemic transcription, without needing a pronunciation dictionary.

Three diphone databases were made available to the speech synthesiser. One being a diphone databases extracted from English carrier words. This database has complete coverage of all the diphones present in Maltese, and achieved an intelligibility score of 2.44 (meaning ‘Not Very’ - ‘Somewhat’). Another being a diphone database with diphones manually extracted from carrier phrases in Maltese recorded for this work. This database achieved an intelligibility score of 3.06 (meaning ‘Somewhat’). While 58 diphones of the 1,350 required for complete coverage of Maltese were extracted, the rest can be simply recorded and extracted with the outlined processes and added to the database for complete coverage. The last diphone database was extracted automatically from the recorded Maltese carrier phrases, using the Dynamic Time Warping (DTW) algorithm and the aforementioned English diphone database for alignment. This diphone database, however, achieved an unsatisfactory result, producing completely unintelligible speech, receiving an intelligibility rating of only 1.04 (meaning ‘Not At All’).

To facilitate the creation of new Diphone Databases for Maltese, a program was written which can automatically generate Carrier Phrases for any given list of diphones. A process was then defined for recording these carrier phrases at the best quality possible, with no specialised equipment. For the manually extracted diphones, a process was also defined which mentions tools that greatly optimise and increase the precision of this task.

The Time Domain - Pitch Synchronous OverLap Add (TD-PSOLA) algorithm was implemented as the Diphone Concatenation algorithm. This was however scored by evaluators as adding more noise to the synthesised speech than a simple ‘appending’ concatenation algorithm. Due to the modular design of the speech synthesiser, the simple concatenation algorithm can be set as the default until a more refined algorithm is implemented.

A prosody modification component was not developed in this implementation, however, even without it, the synthesiser still obtains a rather respectable naturalness score of 2.57 (meaning ‘Not Very’ - ‘Somewhat’).

Not only is this work a significant step forward in building a state of the art speech synthesiser for Maltese, but it can also be used by other low resourced languages. The ‘Text Normalisation’ and ‘G2P’ components would of course need to be adapted, however the rest of the developed programs and processes can be applied to any other language with minimal additional effort.

6.1 Future Work

While the work done satisfies many of the initial aims and objectives of this thesis, there still remains a considerable amount of work to be done for the speech synthesiser to be considered state of the art, or even usable in extent.

6.1.1 Core Functionality

Some work that can be done in each component to greatly improve the quality of the synthesised speech is the following:

Text Normalisation:

- Implementing NSW handling for a higher range of numbers, negative and floating point numbers, dates, abbreviations and symbols.
- Implementing smarter NSW detection. For example, the number 21123456 should not be read as “*wieħed u għoxrin miljun ...*”, but rather as “*tnejn wieħed, wieħed ...*”, as it is probably a telephone number, and not an integer.

G2P:

- Addressing some issues detected in the G2P program, where some instances of the grapheme ‘gh’ are converted to the incorrect phoneme/s.

Diphone Database:

- Following through with the process explained in Section 4.3 so as to create a diphone database with complete coverage of all the diphones present in Maltese, thus being able to synthesise any body of text using a Maltese voice and not having to fall back on an English diphone database.
- Alternatively, the implemented DTW-based algorithm for automatic diphone extraction can be revisited, given that it produced unusable results. This would still work better when using a Maltese diphone database for alignment, so manually extracting one would still be a worthwhile effort.

Diphone Concatenation:

- Revisiting the TD-PSOLA algorithm, or exploring alternative concatenation algorithms, given that the implemented TD-PSOLA algorithm wasn't considered effective by evaluators.

Prosody Modification:

- Implementing prosody modification for sentences with punctuation, such that the intonation and expression of speech is different, given that Diphone database (DB) 3 produced speech which was scored 2.57 for naturalness, i.e. sounds not very - somewhat natural.
- Making it possible to adjust the speed of the synthesised speech. This would make it possible for Maltese listeners who are less fluent to slow down the synthesised speech.

6.1.2 Usability and Accessibility

Apart from the suggested improvements to the core functionality of the synthesiser, there are some accessibility improvements which can be implemented that would, by far, improve the usability of this synthesiser. Keeping in mind that a large portion of the users of speech synthesisers are vision impaired individuals, making the synthesiser highly accessible and effortless to use is paramount. While it was not within the scope of this thesis, building the synthesiser as a web browser plug-in or a smartphone application would improve its usability. For example, a user reading an article written in Maltese on a website can simply press a button within their browser, or perhaps issue a voice command, for the text to be read to them.

This should not be complex to implement. The synthesiser implementation is very lightweight, with all algorithms chosen with efficiency in mind and with the size of the entire diphone database at 10 MB, allowing it to run on virtually any hardware. Alternatively, the core script can be run on a web server with Python installed and the web plug-ins/android apps can send raw text to the server, which will then return the synthesised text as an audio file.

One additional minor fix that can be done is synthesising a very large body of text paragraph by paragraph, instead of synthesising the entirety of it as a whole. This would make playback instantaneous for the user, as they can listen to the first paragraph, while subsequent paragraphs are synthesised in the background. This is, again, a very simple fix as all that needs to be changed is instead of sending all of the phonemic transcription from the front-end to the back-end, it is sent batch by batch, and played one after the other.

6.2 Discussion

All in all, a basic, yet modular and expandable, Text Normalisation component as well as a very robust and versatile G2P component were built. Three separate ‘Diphone Databases’ were made available to the speech synthesiser, and the TD-PSOLA concatenation algorithm was implemented. The diphone database of manually extracted diphones from Maltese carrier phrases concatenated by the TD-PSOLA algorithm, both created as part of this dissertation, outperformed the speech synthesised when using the professionally recorded English diphone set.

References

- [1] Census of population and housing. Technical report, National Statistics Office, 2011.
- [2] Shunsuke Aihara. fastdtw, Aug 2015.
- [3] Areej Al-Wabil, Hend Al-Khalifa, and Wafa Al-Saleh. Arabic text-to-speech synthesis: A preliminary evaluation. In *EdMedia: World Conference on Educational Media and Technology*, pages 4423–4430. Association for the Advancement of Computing in Education (AACE), 2007.
- [4] Najwa K. Bakhsh, Saleh Alshomrani, and Imtiaz Khan. A comparative study of arabic text-to-speech synthesis systems. *International Journal of Information Engineering and Electronic Business*, 6(4):27–31, 08 2014.
- [5] Slobodan Beliga and Sanda Martincic-Ipsic. Text normalization for croatian speech synthesis. pages 1664–1669. IEEE Publishing, 2011.
- [6] A. Bellur, K. B. Narayan, K. Raghava Krishnan, and H. A. Murthy. Prosody modeling for syllable-based concatenative speech synthesis of hindi and tamil. In *2011 National Conference on Communications (NCC)*, pages 1–5, Jan 2011.
- [7] A. W. Black and K. A. Lenzo. Multilingual text-to-speech synthesis. In *2004 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 3, pages iii–761, May 2004.
- [8] Alan Black and Kevin Lenzo. Building voices in the festival speech synthesis system, 2000.
- [9] Paul Boersma and David Weenink. Praat: doing phonetics by computer.
- [10] Mark Borg, Keith Bugeja, Gordon Mangion, and Carmel Gafa. Preparation of a free-running text corpus for maltese concatenative speech synthesis. Apr 2011.
- [11] Joseph M Brincat. Maltese—an unusual formula. *MED Magazine*, 27, 2005.
- [12] Paul Brossier, Tintamar, Eduard Mller, Nils Philippsen, Tres Seaver, Hannes Fritz, Cyclopsian, Sam Alexander, Jon Williams, James Cowgill, and et al. aubio/aubio: 0.4.9. Feb 2019.

- [13] H. T. Bunnell, Debra Yarrington, Steve Hoskins, and Keith Taylor. Speech synthesis program. *Journal of Rehabilitation Research and Development*, 33:154, 06 1996.
- [14] Steve Cassidy and Jonathan Harrington. Multi-level annotation in the emu speech database management system. *Speech communication*, 33(1-2):61–77, 2001.
- [15] Alain de Cheveign and Hideki Kawahara. Yin, a fundamental frequency estimator for speech and music. *Journal of the Acoustical Society of America*, 111(4):1917–30, 2002.
- [16] Robert A.J. Clark, Korin Richmond, and Simon King. Multisyn: Open-domain unit selection for the festival speech synthesis system. *Speech Communication*, 49(4):317 – 330, 2007.
- [17] M. O. Co and R. C. L. Guevara. Prosody modification in filipino speech synthesis using dynamic time warping. In *TENCON 2003. Conference on Convergent Technologies for Asia-Pacific Region*, volume 1, pages 397–401 Vol.1, Oct 2003.
- [18] Armin W Doerry. Catalog of window taper functions for sidelobe control. *Sandia National Laboratories, Albuquerque, New Mexico, USA*, 2017.
- [19] J. Dubnowski, R. Schafer, and L. Rabiner. Real-time digital hardware pitch detector. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 24(1):2–8, February 1976.
- [20] Thierry Dutoit. *An Introduction to Text-to-speech Synthesis*. Kluwer Academic Publishers, Norwell, MA, USA, 1997.
- [21] Paulseph-John Farrugia. Text to speech technologies for mobile telephony services. 2005.
- [22] Hamza Frihia and Halima Bahi. Hmm/svm segmentation and labelling of arabic speech for speech recognition applications. *International Journal of Speech Technology*, 20(3):563–573, Sep 2017.
- [23] Wendy Holmes. *Speech synthesis and recognition*. CRC press, 2001.
- [24] A. J. Hunt and A. W. Black. Unit selection in a concatenative speech synthesis system using a large speech database. In *1996 IEEE International Conference on Acoustics*,

- Speech, and Signal Processing Conference Proceedings*, volume 1, pages 373–376 vol. 1, May 1996.
- [25] R. Janakiraman, J. C. Kumar, and H. A. Murthy. Robust syllable segmentation and its application to syllable-centric continuous speech recognition. In *2010 National Conference On Communications (NCC)*, pages 1–5, Jan 2010.
 - [26] Pijus Kasparaitis and Kipras Kancys. Phoneme vs. diphone in unit selection tts of lithuanian. *Baltic Journal of Modern Computing*, 6(2):162–172, 2018.
 - [27] S P Kishore and Alan W Black. Unit size in unit selection speech synthesis. In *IN PROC. EUROSPEECH 2003*, pages 1317–1320, 2003.
 - [28] Thomas Kisler, Uwe Reichel, and Florian Schiel. Multilingual processing of speech via web services. *Computer Speech & Language*, 45:326–347, 2017.
 - [29] A Klautau. Arpabet and the timit alphabet, 2001.
 - [30] Ki-Seung Lee. Mlp-based phone boundary refining for a tts database. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(3):981–989, May 2006.
 - [31] Kevin A Lenzo and Alan W Black. Diphone collection and synthesis. In *Sixth International Conference on Spoken Language Processing*, 2000.
 - [32] Estella P.-M. Ma and Amanda L. Love. Electroglossographic evaluation of age and gender effects during sustained phonation and connected speech. *Journal of Voice*, 24(2):146 – 152, 2010.
 - [33] Fabrice Malfrere and Thierry Dutoit. High-quality speech synthesis for phonetic speech segmentation. In *Fifth European Conference on Speech Communication and Technology*, 1997.
 - [34] FITA Malta. Erdf 114 maltese text to speech synthesis.
 - [35] Philip McLeod and Geoff Wyvill. A smarter way to find pitch. In *ICMC*, 2005.
 - [36] Paul Micallef. *A text to speech synthesis system for Maltese*. PhD thesis, University of Surrey (United Kingdom), 1997.


- [37] K. Prahallad and A. W. Black. Segmentation of monologues in audio books for building synthetic voices. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(5):1444–1449, July 2011.
- [38] Ernst Pulgram. Phoneme and grapheme: A parallel. *Word*, 7(1):15–20, 1951.
- [39] E. Veera Raghavendra, B. Yegnanarayana, and K. Prahallad. Speech synthesis using approximate matching of syllables. In *2008 IEEE Spoken Language Technology Workshop*, pages 37–40, Dec 2008.
- [40] MZ Rashad, Hazem M El-Bakry, and Islam R Isma’il. Diphone speech synthesis system for arabic using mary tts.
- [41] Stan Salvador and Philip Chan. Fastdtw: Toward accurate dynamic time warping in linear time and space. *Intelligent Data Analysis*, 11(5):561–580, 2007.
- [42] Florian Schiel. Automatic phonetic transcription of non-prompted speech. 1999.
- [43] Tanja Schultz. Globalphone: a multilingual speech and text database developed at karlsruhe university. In *Seventh International Conference on Spoken Language Processing*, 2002.
- [44] Richard Sproat, Alan W. Black, Stanley Chen, Shankar Kumar, Mari Ostendorf, and Christopher Richards. Normalization of non-standard words. *Comput. Speech Lang.*, 15(3):287–333, July 2001.
- [45] Adriana Stan, Junichi Yamagishi, Simon King, and Matthew Aylett. The romanian speech synthesis (rss) corpus: Building a high quality hmm-based speech synthesis system using a high sampling rate. *Speech Communication*, 53(3):442 – 450, 2011.
- [46] Y. Stylianou. Applying the harmonic plus noise model in concatenative speech synthesis. *IEEE Transactions on Speech and Audio Processing*, 9(1):21–29, Jan 2001.
- [47] S. Toma, G. Tarsa, E. Oancea, D. Munteanu, F. Totir, and L. Anton. A td-psola based method for speech synthesis and compression. In *2010 8th International Conference on Communications*, pages 123–126, June 2010.
- [48] Aäron Van Den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. In *SSW*, page 125, 2016.

- [49] M. V. Vinodh, A. Bellur, K. B. Narayan, D. M. Thakare, A. Susan, N. M. Suthakar, and H. A. Murthy. Using polysyllabic units for text to speech synthesis in indian languages. In *2010 National Conference On Communications (NCC)*, pages 1–5, Jan 2010.
- [50] M. Zeki, O. O. Khalifa, and A. W. Naji. Development of an arabic text-to-speech system. In *International Conference on Computer and Communication Engineering (ICCCE'10)*, pages 1–5, May 2010.
- [51] Heiga Zen. Generative model-based text-to-speech synthesis. In *2018 IEEE 7th Global Conference on Consumer Electronics (GCCE)*, pages 327–328. IEEE, 2018.
- [52] Heiga Zen, Takashi Nose, Junichi Yamagishi, Shinji Sako, Takashi Masuko, Alan W Black, and Keiichi Tokuda. The hmm-based speech synthesis system (hts) version 2.0. *Proc. of ISCA SSW6*, 09 2007.

Appendix A - The Evaluation Survey

Voice 1

Voice 1 - Phrase 1



How would you rate the speech sample above? *

Natural refers to how human-like the speech sounds. Clear refers to any background noise in the generated speech. Intelligible refers to how well you can understand what is being said.

	Not at all	Not very	Somewhat	Quite	Very
Natural	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Clear	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Intelligible	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Figure 10: Question Type 1 of the Evaluation Survey, asked for Phrases 1 and 2 of each Diphone Database

What do you think was said in the speech sample above? Were there any words which were particularly unintelligible? Do you have any further comments?

Your answer

Figure 11: Question Type 2 of the Evaluation Survey, asked for Phrase 3 of each Diphone Database

Comparison

Which Voice do you think sounds best overall? *

- ☐ 1
- ☐ 2
- ☐ 3

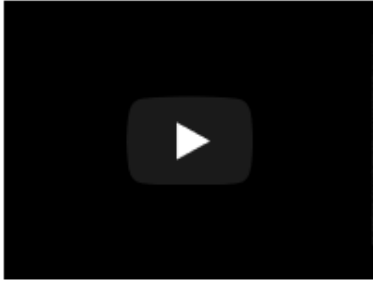
Why?

Your answer

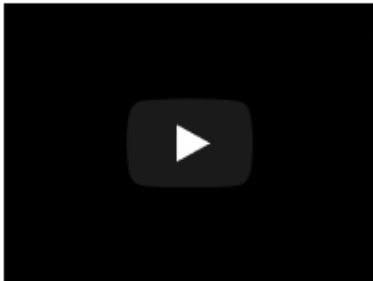
Figure 12: Question Type 3 of the Evaluation Survey, asked after 3 Phrases are presented from each Diphone Database

Concatenation Algorithm Comparison

Voice 1 - Phrase 4 - v1



Voice 1 - Phrase 4 - v2



After listening to the two speech samples above, which do you think sounded better?

- ☐ v1
- ☐ v2
- ☐ They sound the same

Figure 13: Question Type 4 of the Evaluation Survey, asked after presenting two versions of the same Phrase, one using simple concatenation, the other with TD-PSOLA