April 14, 2021

University of St Andrews
School of Computer Science

CS4098 Minor Software Project

Professional Pronunciation Tool

Final report

Lim Jin-Han

Supervisor : Dr Ishbel Duncan

Abstract & Declaration

0.1 Abstract

There are many pronunciation tools in the current day and age which help users improve pronunciation but there are fewer pronunciation tools that help with scientific words and in British English. As a foreign student studying in the UK, there are noticeable pronunciation differences when conversing on a day to day basis let alone in the professional field. Words such as paracetamol and ibuprofen are pronounced differently depending on where the speaker is from and as such it would be beneficial for international students in the STEM fields to be able to communicate fluently their intent. Scientific terminology are often difficult and complex to pronounce due to the proficiency of language it commend. The pronunciation tool create aims to assist non-native speakers improve their British English pronunciations for scientific terms. The tool stores audio recordings of professional terms and allows for playback. Following this, users are then allowed to record their own attempts at pronouncing the word and are then provided feedback after being compared with the professionals.

0.2 Declaration

Contents

Chapter 1

Introduction

In each STEM field, there is a multitude of scientific words and phrases used on frequent basis. These words and phrases often have complex pronunciation and may written similar to one another. For example, the word Ileum and Ilium may be have different pronunciation and mean different things but to a non-native speaker may be a point of confusion. Clarity is of major importance in fields which have less room for error such as the medicine or bio-chemistry. Furthermore, the differences between American spoken and British spoken English makes it difficult to streamline these terms with respect to pronunciation. As such, it may prove to be a barrier for academics who speak English as a second language to come up to par with locals when attempting to pronounce such words.

The focal point for this project was to alleviate some difficulties that non-native speakers may encounter when attempting to learn such scientific words by providing a tool to guide them. The tool would be of use not just to students but also professionals struggling in certain pronunciations. The tool will be used to store professional pronunciation of words and users will be able to record their own pronunciation to be compared and provided with feedback.

This report will provide some insight towards the problems with pronunciation alongside an overview of the tool that was created and the decision process behind the development. This will be followed with an evaluation and some ideas for future projects with similar goals.

The report will first begin by provided some background research on pronunciation and scientific words along with some currently available pronunciation tools. It was then discuss the tool's specifications when envisioned along with explanations for design choices. Following this, the development process will be explained along with the external tools used in the project. The experimentation and implementation will be described next which will show a step-by-step in the development of the tool. After, a testing and evaluation section will be used to gauge the functionality of the tool. Lastly, a discussion and conclusion section to discuss and motivate future projects. Below are the objectives set during the planning stages of the pronunciation tool.

1.1 Objectives

1.1.1 Primary Objectives

- Storage and playback of professional pronunciations for the scientific terms
- Recording and storage of user attempts at the word pronunciation
- Comparing both audio and scoring the attempt by some metric
- Providing feedback to the user to help improve

1.1.2 Secondary Objectives

- User-friendly User Interface (Web based)
- A search function with an auto-fill feature
- Grouping terms by pronunciation respelling
- Grouping terms by how often used in tandem
- Recommending other words based on the words attempt
- Normalizing professional pronunciations
- Keep a statistical record for comparison

Chapter 2

Background Research

Pronunciation of words have been researched throughout history and there are various factors that come into play when pronouncing words. Ideas such as pitch, intonation and accents can drastically alter word pronunciation. This section aims to cover background knowledge and provides an idea for what should be implemented into the tool.

2.1 Literature Review

In the context of communication, pronunciation is crucial as accurate delivery of ideas may be incomprehensible when poor pronunciation is involved. According to Duncan (1983), communication is insufficient if the listener is unable to decrypt the connotations expressed by a speaker. Wei and Zhou (2002) noticed that for Thai speakers, pronunciation problems can occur with consonants, vowels, intonation and stress and as such these factors may affect the learning capabilities of any second language (L2) speaker. A majority of the issues faced here in the acquisition of L2 is due to the different language styles of English and Thai pronunciation. This is due to the difference is each languages phonics and they proceed to state that Japanese find it difficult to differentiate between ”r” and ”l” also due to phonics. Here they also bring up a fair comparison in which an English speaker would have troubles pronouncing the Chinese word for ”mother” and ”horse”. This issue arises when looking at the romanization of Chinese characters called ”Pin Yin” in which horse is written as Mǎ and mother is written as Māmā. Although the difference is subtle, the intonation is different for both of the ”Ma” sound in which the tones used are displayed above the ”a” in which horse being the 3rd tone while mother uses the 1st tone,

Instruction in pronunciation is the main source of understanding new words as it provides insight on native speakers and provides goals to improve speech in order for a much more refined pronunciation (Gilakjani, 2016). If the speaker is unable to pronounce words correctly, communication will be limited not only due to the fact that they might not be fluent with the language but it also ties back to Duncan's point in which the listener might not be able to understand the the speaker.

Thomson (2011) states that computer assisted pronunciation training (CAPT) is a necessary tool going forward in the future of language teaching. An example of a CAPT application is a program ”that provides a pronunciation meter that purportedly tells learners how native-like their pronunciation is”. It was basically a program that scores a user's pronunciation based on how proper their pronunciation was in the context of the programs origin. This application shares similar traits to the pronunciation tool that this project revolves around. Neri et al. (2008) also carried out research on the effectiveness of CAPT to teach children English as a foreign language. The CAPT system introduced was PARLING which included an automated speech recognition (ASR) module which was based on Hidden Markov Models (HMMs), which are defined by Rabiner and Juang (1986) as "a doubly stochastic process with an underlying stochastic process that is not observable, but can only be observed through another set of stochastic processes that produce the sequence of observed symbols", trained on British English. In a more basic explaination, HMMs contain hidden states and observable states in which one has to assume the hidden states based off the states observable. Results showed that the CAPT training provided improve the childrens pronunciation skills and Neri et al., similar to Thomson, believed that there is room for the integration of CAPT into traditional teaching in either a case of a large student teacher disparity or a case of time constraints. The teaching tool would be a way to alleviate some teaching duty from teachers to allow for more time to focus on specific speech problems in individual students, which was not solvable by CAPTs. Current day speech recognition systems still rely on HMMs.

The results from Lee's (2015) experiments show that oral corrective feedback should be used to construct a positive environment for L2 students. In the study, a majority of the participants were from Asian countries and as such English was not the first language. There was an improvement in English proficiency when a positive environment was established due to a reduction in anxiety and emotional distress. It was also noted in the study that clarification request such as "What?" or "Sorry?" could provoke additional anxiety in L2 students.

Koren (1995) proposes pronunciation tests for adults and children due as the current pronunciation proficiency tests does not portray an accurate representation of a foreign language learner's pronunciation. The tests are based upon pronunciation, stress and intonation and the research was concluded to have produced two useful tools for testing pronunciation. Two reasons for its success is due to the tests ability to differentiate between the three articulation aspects and an objective rating scale.

Yang (2005) discusses in her paper that nursing pre-professionals are troubled when having to learn medical terminology in Taiwan. Her research displayed that although higher-level learners were able to learn and remember more words, both groups were not willing to use social strategies to improve their vocabulary. These includes working in groups and requesting for help. It was discovered that visual and verbal repetition were main features in the participants learning methods. This was seen to be a cultural issue which stunts the populous.

2.2 Similar available tools

A tool developed by Google and built into their search engine includes the ability to search up words which provide verbal and textual help with pronouncing words. By typing a word followed by "pronounce" in Google, a tab will show up in the web page which provides the word, a selection of American or British pronunciation, the word broken down into sounds, word playback including a slow down function and a visual aid which syncs with the playback. In the example below, the word entrepreneur is broken down into different sound depending on British or American English. It is noticed here that it would be desirable to have some kind of word splitting to visually aid the user.

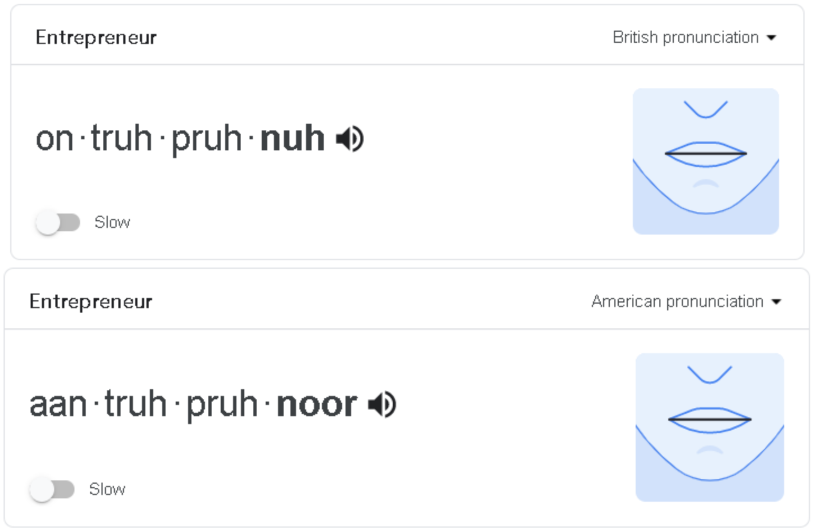
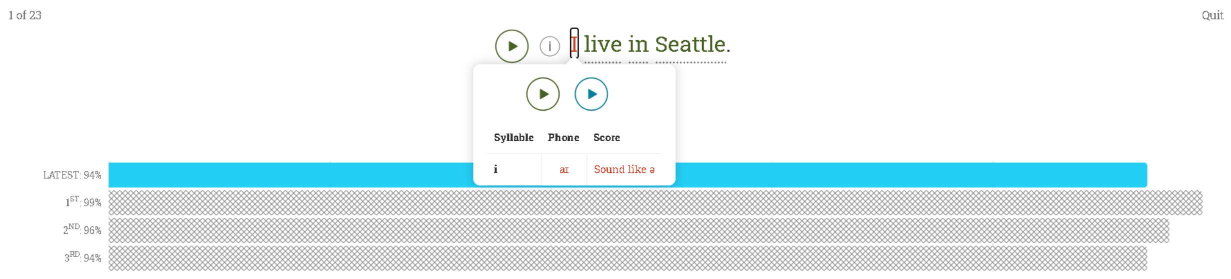


Figure 2.1: *Comparison between British pronunciation (Top) and American pronunciation (Bottom) on Google*

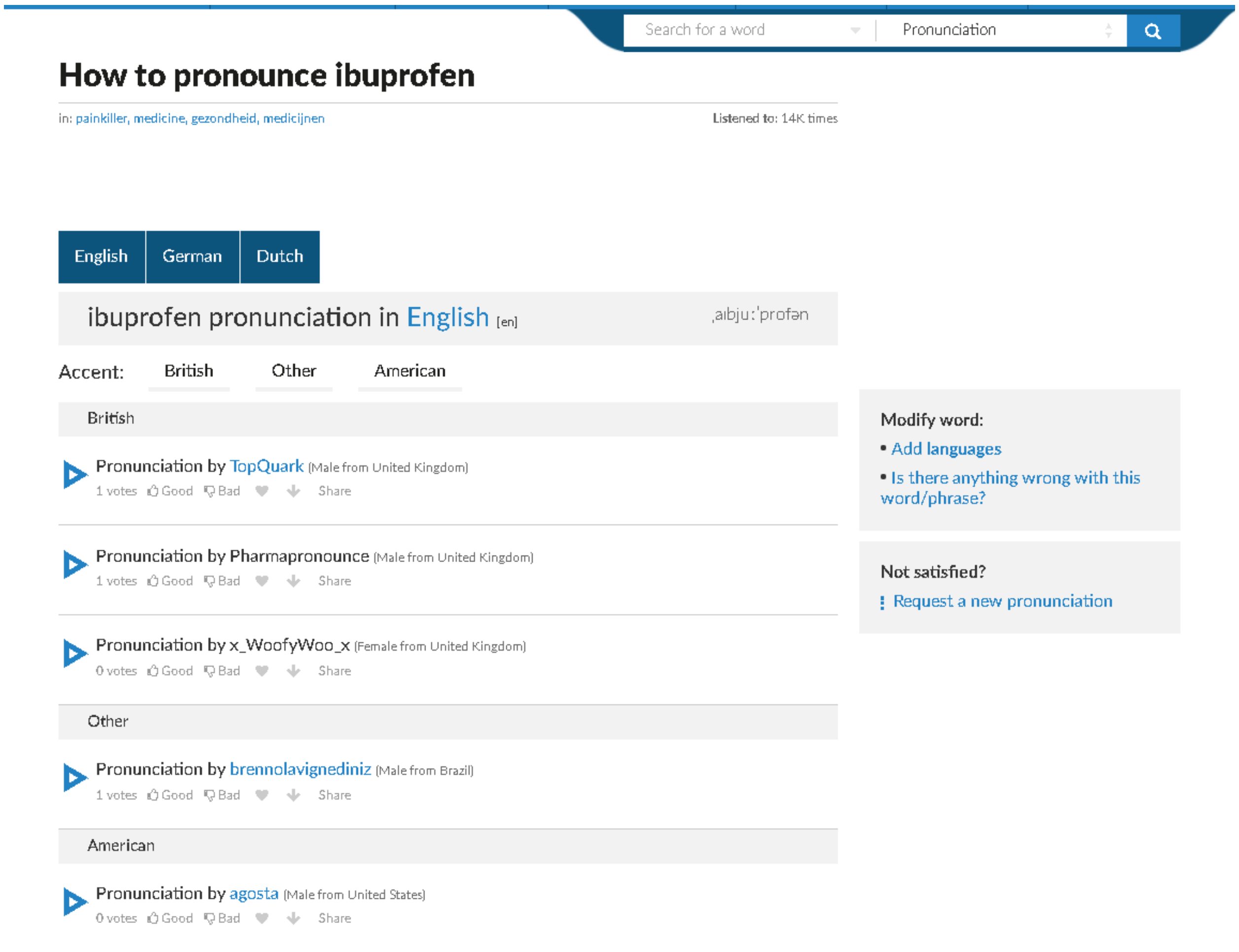
Another tool discovered was the Speechace which uses speech recognition to assess pronunciation and fluency of language. The application returns a score depending on the timing and correctness of phonemes depending on the word the user attempts. When recording an attempt, the latest attempt score will be displayed and the previous ones are displayed below it. The word that the user pronounced wrongly is highlighted and when clicked shows each phoneme in the word and the specific phoneme that the user mispronounced. In the case of the example below, the syllable I was spoken more like 'a'. The highlighting system locates the point of error for the user which notifies where they are struggling and as such is a useful feedback mechanism.



Syllable	Phone	Score
i	aɪ	Sound like a

Figure 2.2: Speechace *pronunciation tool*

Forvo is another pronunciation tool publicly available. The key difference between Forvo and the previously mentioned tools is that Forvo contains a database of user submitted pronunciations for different words. When searching up words, a list of different audio files are returned along with the submitter's username, gender and country of origin. This is then further sorted into accents to allow the user to freely choose a specific accent to learn. When clicking on a submitter's username, a page is brought up which contains every word or phrase they have added into the database. Assuming the submitter has pronounced some scientific terms, there is a higher chance they might have submitted other scientific terms in the same field to the website. These features allow for users to explore a multitude of pronunciations for various words and may encourage them to record their own words and further expand an evergrowing database. An example for the word ibuprofen can be seen in figure 2.3. The site also allows the user to learn words from various other languages. One of the main caveats for Forvo is that it strongly encourages user participation and would not be functional without its active userbase.



How to pronounce ibuprofen

in: painkiller, medicine, gezondheid, medicijnen

Listened to: 14K times

ibuprofen pronunciation in English [en] ˌaɪbjuːˈprofən

Accent: British Other American

British

Pronunciation by TopQuark (Male from United Kingdom)
1 votes Good Bad Share

Pronunciation by Pharmapronounce (Male from United Kingdom)
1 votes Good Bad Share

Pronunciation by x_WoofyWoo_x (Female from United Kingdom)
0 votes Good Bad Share

Other

Pronunciation by brennolavignediniz (Male from Brazil)
1 votes Good Bad Share

American

Pronunciation by agosta (Male from United States)
0 votes Good Bad Share

Modify word:
- Add languages
- Is there anything wrong with this word/phrase?

Not satisfied?
Request a new pronunciation

Figure 2.3: *Forvo user submitted database*

Finally, although not a tool for pronunciation, Shazam is an interesting tool used for audio analysis for music. It is able to listen and identify music based on audio fingerprinting. This is done by Shazam first capturing the audio source and proceeding to pattern match similarities of the audio using a spectrogram with a database of music which have already been fingerprinted. The application runs on multiple devices and has been further improved since its introduction with features to integrate it with various music playing applications and lyric syncing. Using the same principles as Shazam, an application could be created using audio fingerprinting to identify proper pronunciation and score attempts based off the initial audio fingerprinting.

Chapter 3

Requirements Specification

At the beginning of the project, a DOER document was submitted which contains the required specifications for the pronunciation tool and the following is the same requirements which will be explained in detail why each requirement is necessary for the tool. Higher level priorities are focused on primarily to provide a minimum viable product initially and when sufficiently implemented, the lower level priorities were worked on, which mainly consists of functions to improve the overall applications quality of life. Some functionalities were altered during the development process and will be discussed further on.

3.1 High Priority

3.1.1 Requirement 1

Storage and playback for the professional pronunciation of scientific terms.

- An essential function for the tool would be to able to store each of the audio files and playback through the application.

3.1.2 Requirement 2

Record and store user's attempts at the word pronunciation

- For the user to be able to test their pronunciations, their own recordings should record with similar specifications for frequency, sample rate and if the audio is mono or stereo.

3.1.3 Requirement 3

Compare user's audio file against the professionals and score the attempt.

- The main purpose for the application is to enable users to improve pronunciation by providing them a platform to learn.
- This is where the audio files are compared and feedback will be provided in a metric that should be suitable for any users to identify key problems in their own pronunciation

3.1.4 Requirement 4

Provide feedback on how to improve pronunciation

- This function was initially supposed to provide moderate feedback in areas where the user might be weak in.
- It was conceptualized to analyse each syllables and find the area in which the user is weakest in and provide other word with the same syllables.

3.2 Low Priority

3.2.1 Requirement 5

User-friendly UI (Web based front end).

- A web based application was the intended final product but as to be seen in the further sections is was later changed into an executable.

3.2.2 Requirement 6

Search function which can autofill.

- This was intended as a quality of life function for users in the event they do not have specific terms they are looking for.
- Autofill is a function found is a majority of applications which a search function as it alleviates the problems users might face when searching up words with difficult spelling.

3.2.3 Requirement 7

Group pronunciation by respelling.

- The respelling function was meant to group terms by a respelling to indicate the word syllables and stress.
- This was similar to the previously mentioned tool by Google.
- Searching by respelling would be useful when users are practicing word which contain a certain sound.

3.2.4 Requirement 8

Group terms based on how often they are used in tandem.

- This was envisioned to provide other terms for the user to attempt to pronounce which are used in similar context to the main term that the user is practicing.
- Later, the idea was deemed to be too abstract to be able to implement and was altered to a categorical listing.
- The idea would work much better on a dictionary type pronunciation tool or a sentence pronunciation tool.

3.2.5 Requirement 9

Recommend other words based on the words attempted.

- This would be using the syllables of each word and finding syllables of other words in the database to provide to the user

3.2.6 Requirement 10

Stored professional pronunciation should be normalized.

- This would be to provide for fair testing with the user spoken terms
- It was attempted to normalize both the professional pronunciation and the user pronunciation to similar peaks.

3.2.7 Requirement 11

Statistical records for comparison.

- It was envisioned to gather up every output and store it for the user to come back and check on their progress.

Chapter 4

Software Engineering Process

4.1 Agile approach

An Agile approach was used when planning the development of the tool through weekly meetings with my supervisor and while coming up with requirements for the end product. Early development for the basic features were priorities and were further expanded upon later. Meetings were carried out either weekly or biweekly through Microsoft Teams to provide updates on ideas for project implementation. The Agile approach was chosen due to its familiarity and previous usage in the Junior Honors project. The further use of Agile in the application will only benefit and reinforce the concepts that were previously learnt.

Sprints were initially decided upon by divide the main implementations of the requirements into portions that will be worked on in a few weeks. This was later altered to enable more flexibility due to some requirements being more difficult to implement than others. After essential requirements were developed into a functional state, the additional lower priority requirements were worked on. While working throughout the weeks, additional functions and features were conceptualized and implemented depending on its practicality and worth to the tool as a whole. A cycle involved research and planning, followed by designing and testing which was repeated and evaluated at every step during production.

Chapter 5

Ethics

Voice recording are a main issue in ethics due to being data which is identifiable. All professional pronunciations which were recorded to be used in the application was stored in a password protected device and the files were renamed as to not associate with the participants. Data was meant to be gathered from two groups of participants. The first being participants who provided professional pronunciations and the second being participants who were tested using the pronunciation tool. A questionnaire was initially envisioned for testers after the participant testing phase but was not provided in the end. The workaround included keeping the audio recordings on password protected device, decide on a naming convention for files to make the data unidentifiable and converting the files which would wipe out any initial metadata which could identify any of the participants. Ethical approval was attained and will be available in the appendix.

Chapter 6

Design

6.1 Development language - Java vs Python

Initially, Java was decided upon as I had more experience with the language than Python, in which my only forms of Python knowledge comes from having used it in previous mathematical modules for calculations. The initial development on Java was while using Processing which was an approach initially used due to its graphical library which would allow for a simple graphical user interface (GUI) to be implemented. Further down the line, it was discovered that the conversions of audio files from M4A to WAV would prove to be challenging and was essential for the program to play the audio files. As such, Python was used instead of Java due to the problems in the conversion of audio. Also, Google's Speech-To-Text API was used in the project which was easier to implement in Python than in Java.

6.2 GUI interface

The main GUI when opening application is as show in Figure 6.1. The functionalities implemented are: playing words, recording words, adding sound files to the database, category selection and category editing.

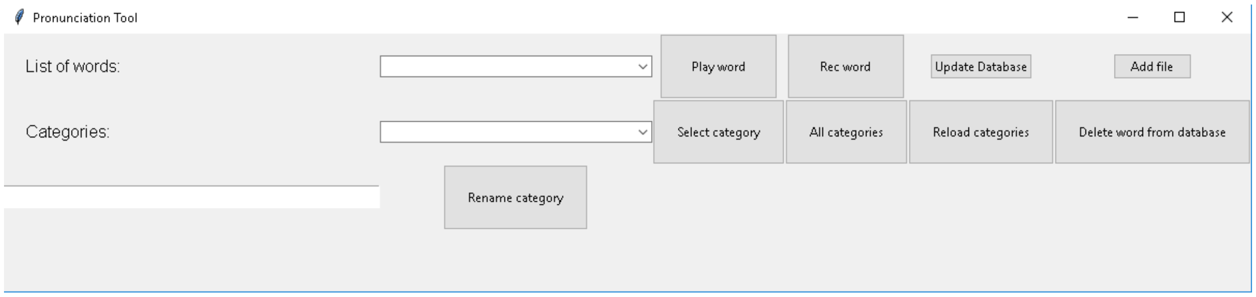


Figure 6.1: *Pronunciation Tool GUI*

6.2.1 Choosing a word

When choosing a word, the user can either click the list box and pick one of the words, type out the whole word or type a portion of a word and click on the list to display words with the portion. The words are displayed in the order they were added into the SQLite database. As show in Figure

6.3, when typing the letters ”in” in the textbox, the list returns only words with ”in” in them. This is a useful function when users may have forgotten the whole spelling of a word but knows it contains certain letters.

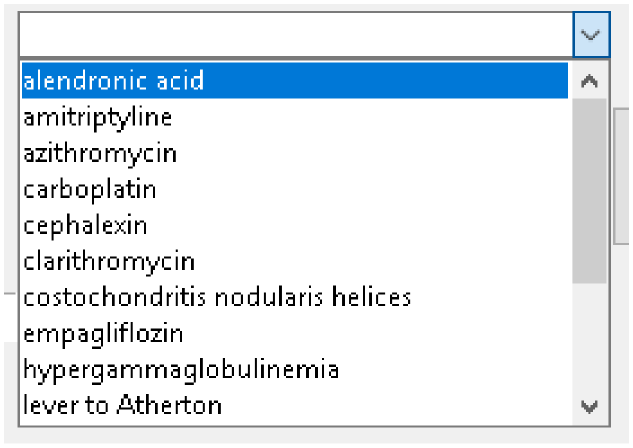

alendronic acid
amitriptyline
azithromycin
carboplatin
cephalexin
clarithromycin
costochondritis nodularis helices
empagliflozin
hypergammaglobulinemia
lever to Atherton

Figure 6.2: *Searching list*

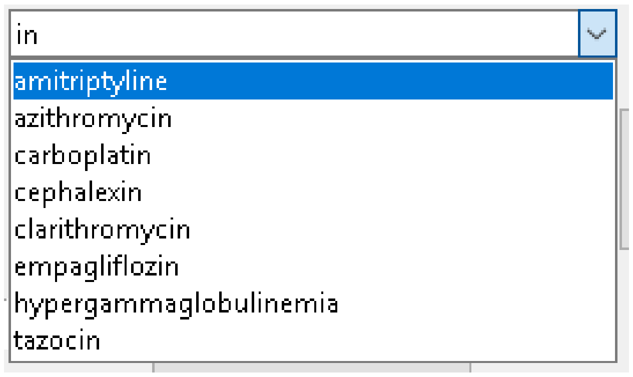

in
amitriptyline
azithromycin
carboplatin
cephalexin
clarithromycin
empagliflozin
hypergammaglobulinemia
tazocin

Figure 6.3: *Searching list with specific word ”in”*

6.2.2 Searching for a word using categories

Words can also be searched up using categories. The category drop down list, as seen in Figure 6.4, is used to select the category and after clicking the select category button, the words in the list will be reduced to only words in the the specific category. Categories are not set in stone and if the user wants to change them they are able to. This can be done by first selecting a word in the list, typing the new category that they want the word associated to in the text box next to ”Rename category” and then pressing the ”Rename category” button. This will replace the category that the word is associated to. If needed, the user can click on the ”All categories” button to categorize the words and there is a button to reload categories after a word’s category is changed to include the new category in the event it did not exist before.

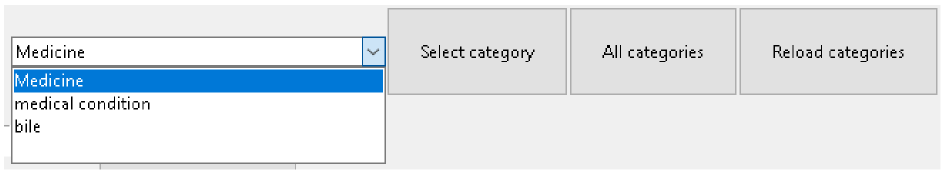
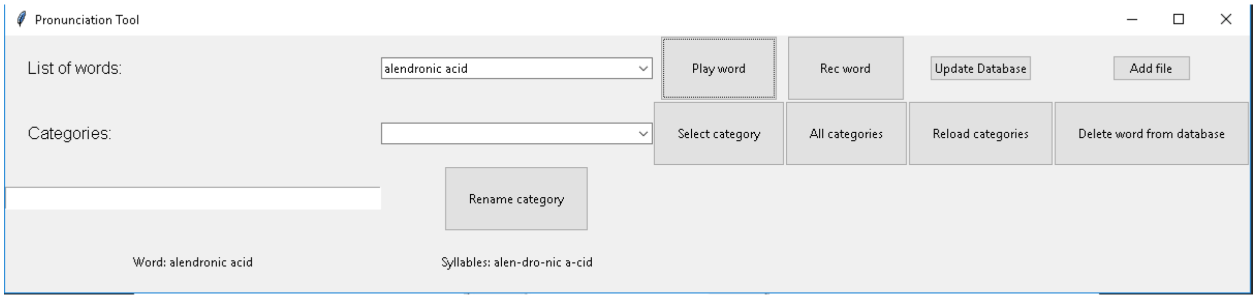


Figure 6.4: *Category for word searching*

6.2.3 Playing a word

To play a word, the word has to be selected in the word list first and then the "Play word" button can be pressed to play the audio file for the user to listen to. In addition to this, the word and its syllable form will be displayed. Figure 6.5 shows an attempt after playing the word "alendronic acid". The word is broken down to its syllables.



Word: alendronic acid

Syllables: alen-dro-nic a-cid

Figure 6.5: *Attempt after playing alendronic acid*

6.2.4 Adding own "professional recordings"

It was devised later that it would be beneficial to implement a feature which would allow users to add their own recordings. This is done with an additional "Add file" button. For the add file to show up in the word list, one has to click the "Update Database" button. The reason these are separate is because it takes a while to update the database due to factors that will be discussed later on how the database was set up. Also only audio files allowed to be added are M4A and WAV. Audio files added by the user will have no category assigned to them as such the user will have to set the category for the word if they would want to. Also, it should be noted that it is not necessary to name the file with the name of the word spoken as the words are ran through the Google Speech-to-text API before adding it to the database and setting the name of the word to the correct word.

6.2.5 Recording audio for testing

When the user is ready to record their attempt, the user can click "Rec word" to start the recording feature and after it finishes, a new window will open to show if the user has pronounced the word correctly or incorrectly. Two graphs of user's audio file is then displayed along with two graphs of the professional's audio, the first being the original and the second an attempt to normalize the audio. The user can then also replay their own pronunciation along with the professionals pronunciation. Figure 6.6 shows a correct attempt at pronouncing alendronic acid wherelse Figure 6.7 displays an incorrect pronunciation of azithromycin. For incorrect pronunciations, the app will let the user know what it thinks the user is saying.

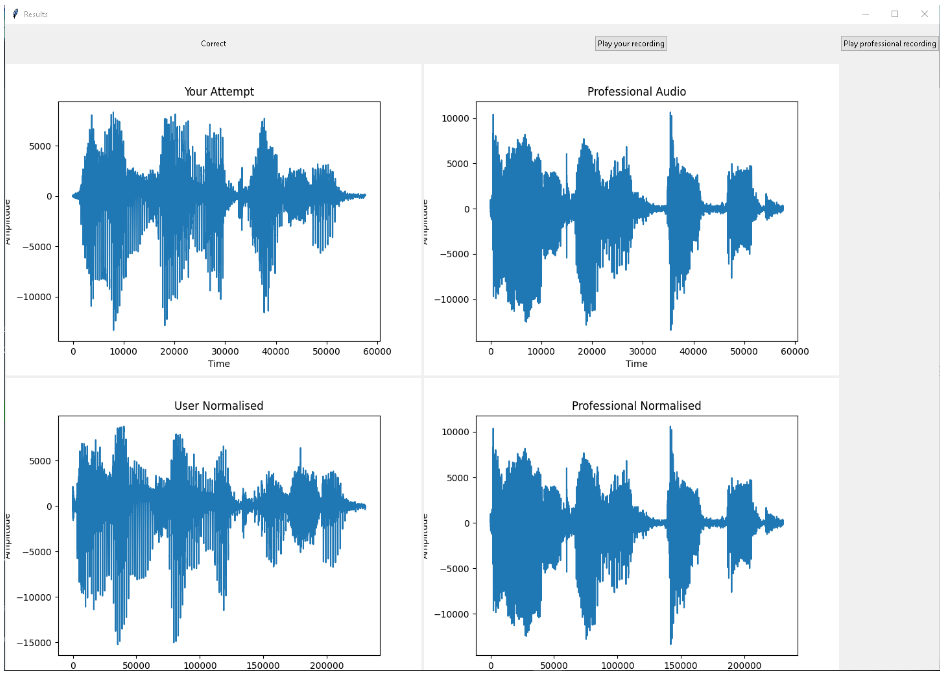


Figure 6.6: *Recording a correct pronunciation of alendronic acid*

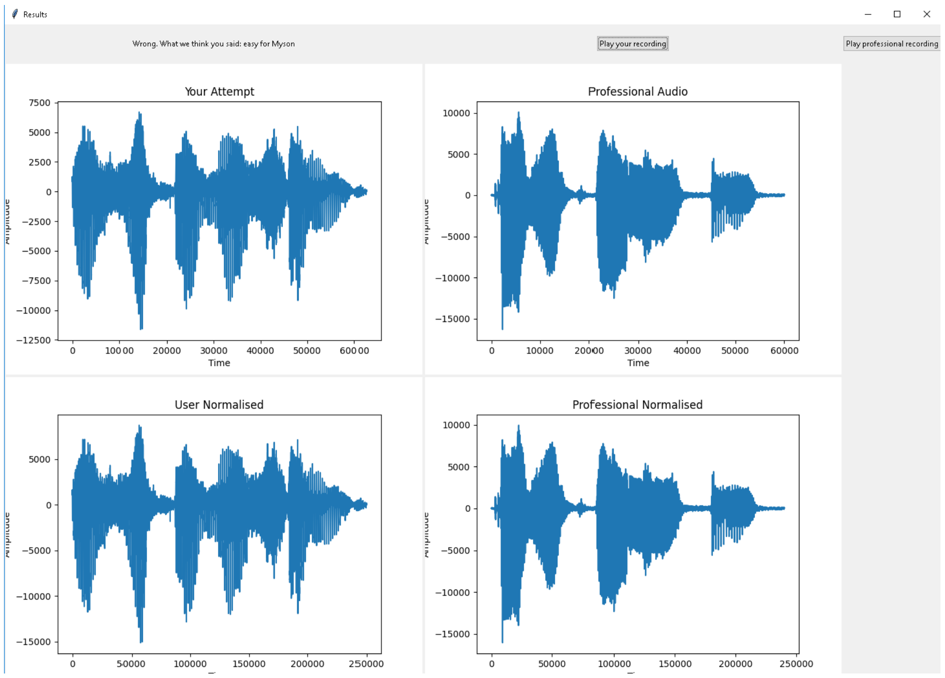


Figure 6.7: *Recording a wrong pronunciation of azithromycin*

6.3 File storage

All audio files which were gathered from professionals were in the M4A format. An ”M4A” folder was created to store all the audio files and after the program is ran, all M4A files are converted into WAV files. After recording the audio, both the users recording and the professional playback are spliced to remove trailing background noise and outputted into the ”out” file. The normalised audio files are also outputted into the ”out” folder. Whenever the user records a new word, the files in the ”out” folder will be updated accordingly. The waveform graphs however are just stored in the main folder.

Chapter 7

Implementation

7.1 Stage 1 - Research

Before implementing the program, it was first necessary to carry out background research in factors which affects a persons ability to pronounce words and any caveats that may come with pronunciation and learning words which were discussed in Section 2.1. Java was decided upon as the language of choice due to familiarity and planning on the functions was carried out. While planning, research into the various functionality and libraries that could provide benefit and use to the project was carried out to provide a baseline on what the program should look like. It was then decided upon, along with the DOER write up, what functions the program required. The main functionality included audio storage, playback, recording and analysis.

7.2 Stage 2 - Playback

Once implementation begun, the initial files were stored in the main folder of the application. This will change later along the stages. Ways to playback audio was researched on and it was initially decide upon that the Java Sound Library will be used to playback audio. Audio formats that were compatible with the library were limited to AIFF, AU and WAV so a conversion of M4A files to WAV was necessary to proceed with audio playback implementation. After some investigation, it was noticed that conversions from M4A to WAV was difficult without the use of FFmpeg which had cross platform libraries for audio recording. At this point, python alternatives were also considered for audio conversion. During one of the research sessions, when taking into account the speech-to-text (STT) which will be used to compare audio files, two Java STT API's were discovered. These were Java Speech API (JSAPI) and Voce, both implementations used similar internals such as CMUSphinx4 and FreeTTS. Both CMUSphinx4 and FreeTTS were also looked into along with Google's Cloud Speech-To-Text API. A notable wrapper that was discovered was SpeechRecognition which supported many of the speech recognition API's including CMU Sphinx and both Google Cloud Speech API and Google Speech Recognition. It was decided upon to use the SpeechRecognition library and thus the project was shifted to Python from Java. Both PyDub and FFmpeg were both installed into the system and a script by Sharma (2019) was modified to convert the M4A files to WAV.

As for the actual playback, some libraries that were considered in python were playsound, sounddevice and PyDub. Since PyDub was already being used, it was natural to continue its usage for playing the audio files as it was able to playback WAV files. SimpleAudio was also installed as recommended by PyDub to play audio.

7.3 Stage 3 - Recording Audio

Moving on to audio recording, this will be used for the user when attempting the pronunce the word, the function should save the pronunciation attempt into a WAV file. PyDub does not contain any functions for recording audio and so a different library will be required for recording. The sounddevice library mentioned in the Section 7.2 contains a function to record the audio. Sounddevice rec function defaults to 2 channel audio and was edited since all professional voice recordings were done in 1 channel. It is also noted that each of the professional audio recordings were of bit rate 768 Kilobit per second (kbps). To ensure the recordings taken using the program contain the same bit rate as the professional recordings, we look at the recording variables. The formula to calculate bit rate is:

$$samplingfrequency * bitdepth * numberofchannels$$

Noting that the professional audio is in single channel this can be reduced to:

$$samplingfrequency * bitdepth$$

The unit for bit rate calculation is bits per second (bps), so with a sampling frequency of 48000 and a bit depth of of 16, this brings the bps to 768000 or, when converted, to 768 kbps, the same as the professional audio bit rate. The audio is also recorded for 5 seconds to accommodate for the longer professional terms.

7.4 Stage 4 - Storage/Database

Currently there are around 15 audio files with unique terms. A database would allow for an easier storage and retrieval functionality along with being able to categorize the terms. The data that will be stored are mostly strings which are used to represent the category, name of file and word name. Not many data types are used and as such SQLite can be used in place of MySQL. It was also decided against building a website and the application will want to be a runnable executable and as such MySQL was unnecessary and since the project is small in scale at the moment, the SQLite library is large enough. Currently the database is set up where each file name is the primary key and the word name is written into the database after checking the audio file with Google's Speech Recognition. The categories for some of the main terms are hard coded into the database but these are just placeholders and it is planned here that the user should be able to modify the categories of the words in case the categories are either wrong, missing or more words are added in. Other column headers that were initially conceptualise but were not implemented were "Best Score" and "Fluent?". These additional headers were meant to be used when after users had recorded their pronunciations and the application and compared. When a score is provided, it would check the current best score for the word from the database and replace it if the new score is higher and after a score threshold is reached, the database would update the "Fluent?" header for the word to "yes". These were never implemented due to the difficulty of the scoring mechanism for words which will be discussed further.

7.5 Stage 5 - Speech-To-Text

After deciding upon using Google's Speech Recognition in the SpeechRecognition library. The speech2text function was used to recognize the word spoken into the audio file and returning the word. This was used for a majority of the project. The first being to fill in the SQLite database, which was done by first running every file from the database using the STT API followed by setting the name for each word in the database to the the word from its corresponding STT output. The one flaw of this is that some words, an example being levetiracetam, are not recognized or misheard by the STT API and as such a combination of what the API thinks the audio is saying is returned instead of the proper term. In the case of levetiracetem, the word returned was "lever to Atherton" and "level 2 Atherton" when two different cases of the word was added into the database, both being pronunciations by the same professional. Another word that suffered from a minor misspelling was "costochondritis nodularis helicis" in which the helicis portion was misspelled as "helices". All other word behaved as intended. This is a major flaw in the implementation which may affect more words unbeknownst in the future.

On the opposite end, another error was discovered when speaking a term in a wrong pronunciation. The API will attempt to match the word the user is saying to the correct word although the pronunciation was either not clear or, at rare occassions, completely wrong. An example was when extending certain vowels, a test on "alendronic acid" was carried out and it is noticeable that the word was pronounced incorrectly but the API judged it as correct as it assumed that was what the user was saying. This assumption by the API may encourage users to continue with a wrong pronunciation and as such should be taken into consideration when using the application.

An offline API that the SpeechRecognition library offered was also considered. This was using the CMUSphinx but was decided against as it requires a dictionary to function which contain little scientific words and as such the idea was scrapped. Other than that, the other API that are in SpeechRecognition were examined and not used due to the fact that they required API keys and user authentication to use.

7.6 Stage 6 - Plotting The Waveform Graph

A few variations of plotting the wave form graph was considered. These included attempting to apply filters on the audio before plotting. Initially a simple plot was formed which reads the audio data and plots it. It was then extended later to plot audio samples equal to the length of the audio file as to remove trailing graphs. Plotting with offsets was also attempted with the intent to ensure the plots are always plotted in the same way. The plots was made using the MatPlotLib and SciPy libraries. The initial plotting attempts are as shown in Figure 7.1.

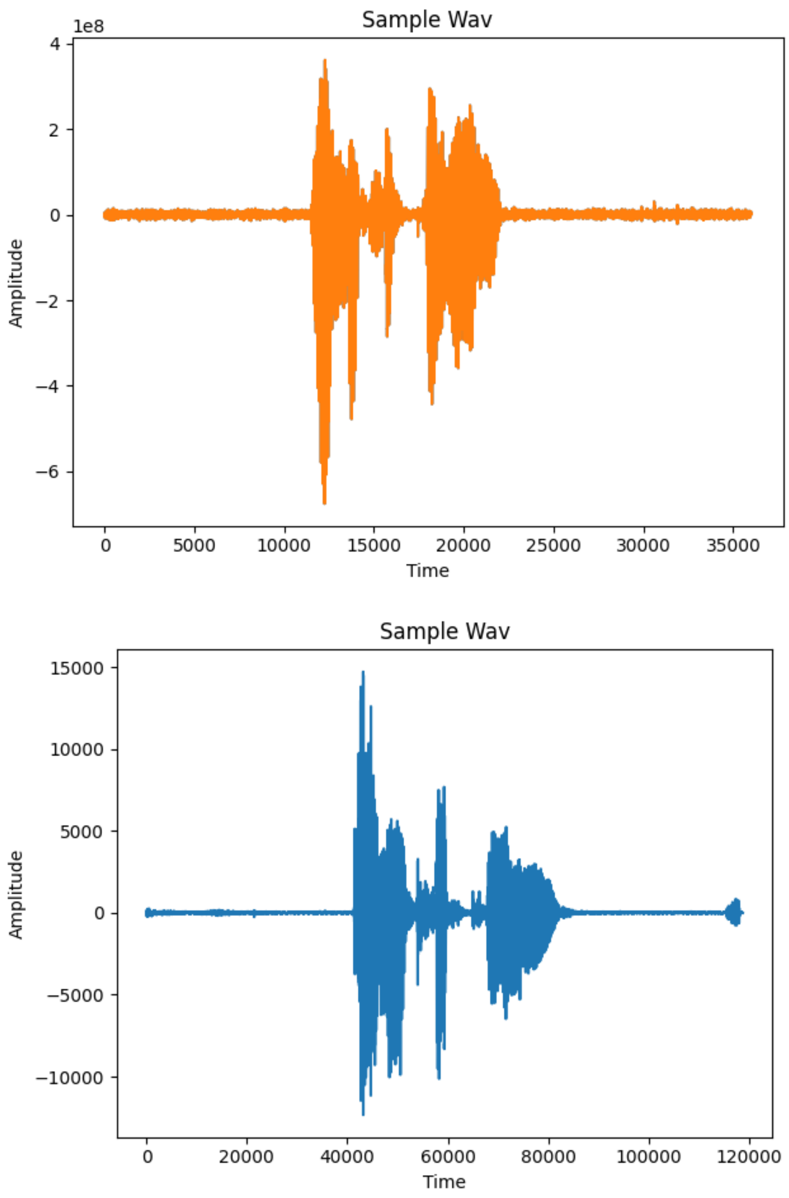


Figure 7.1: *Initial comparison between profession pronunciation graph (Bottom) and my pronunciation graph (Top) for the word Amitriptyline*

Some voice activity detectors were also looked into, these included WebRTC Voice Activity Detector (VAD) which was used to classify audio data if there is voice activity or not. This was deemed unnecessary as it looks into the whole audio file and returns a boolean statement based on if there is actual voice activity in the whole file and does not specifically find areas which have no voice activity. Auditok, an audio tokenization tool was also explored which is a tool used for processing audio files. Auditok was also uses matplotlib to generate graphs along with highlighted audio activity detection. It uses a audio detection threshold to cut out background noise and keep only the human audible sounds. An example of a graph plotted using Auditok can be seen in Figure 7.2

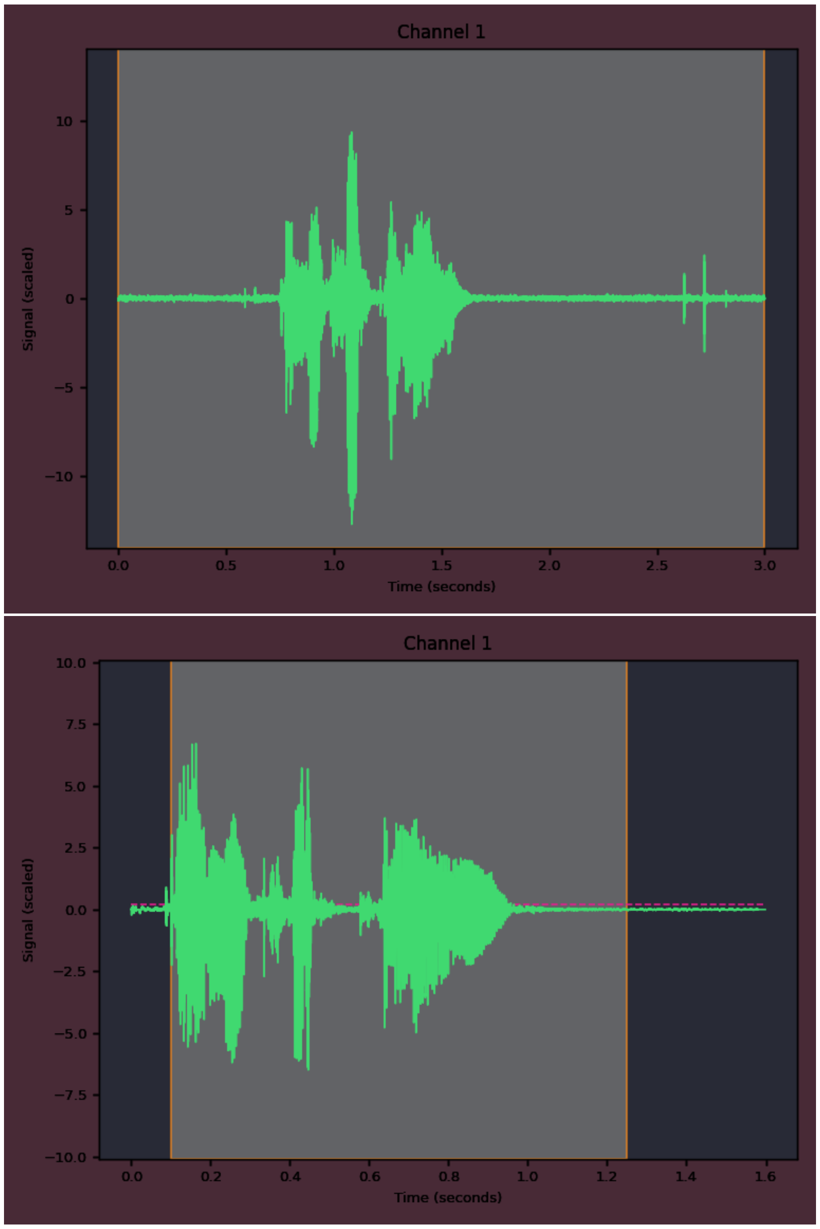


Figure 7.2: *Auditok comparison between profession pronunciation graph (Bottom) and my pronunciation graph (Top) for the word Amitriptyline*

By examining both Figure 7.1 and 7.2, it is noticed that the audio volume is significantly higher for the recordings from my microphone in comparison to the recordings from the professionals. Also the energy threshold for the sound detection is unable to distinguish the background noise from my audio recording when using Auditok. This will be explored later when attempting to splice the audio files to remove unnecessary portions of the audio, those including empty noise trailing after the word has been said but the recording has not terminated.

7.7 Stage 7 - Splicing Audio

Using Auditok, the split function was used to splice audio files based on a the minimum duration of audio, maximum duration of audio, a max amount of continues silence and an energy threshold level. The energy threshold level was left unchanged as from testing, it was a suitable threshold for removing trailing noise from an average of tests. The max silence was set to 0.1 seconds to remove noise in between words in special circumstances and also the a max duration of 5 seconds was set due to every recording only lasting 5 seconds long at most. This was done by following Auditoks documentation. The splicing of audio provides a better looking graph which allows for a more fair comparison as shown in Figure 7.3.

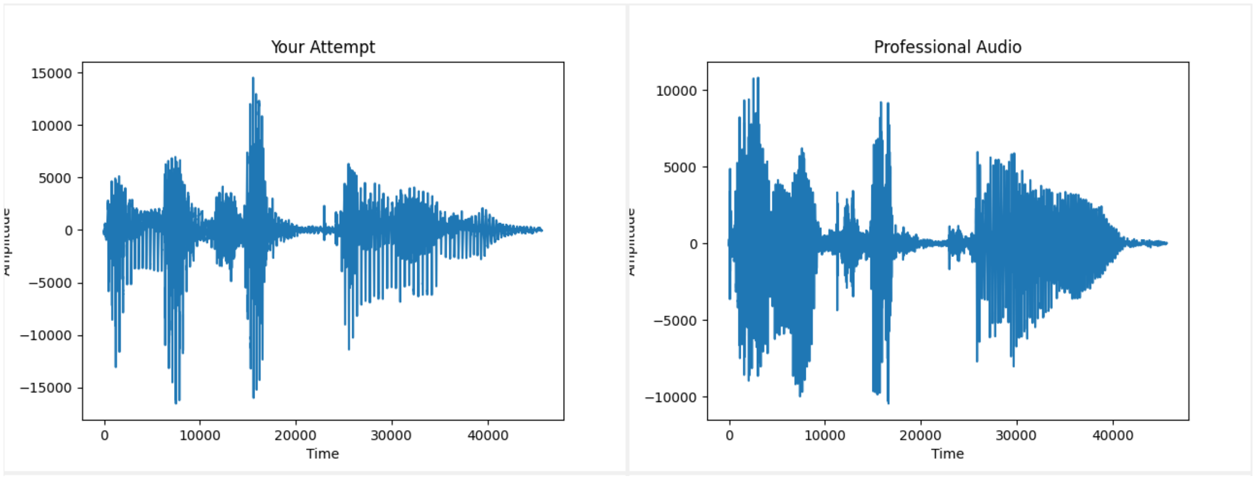


Figure 7.3: *Graph of spliced recordings of Amitriptyline. User attempt (Left) and Professional (Right)*

Comparing Figure 7.3 with Figure 7.1, the spliced graphs provide a better picture on the pronunciation of the user. The spliced graphs allow the user to more effectively locate the pitching or vocal errors solely due to the fact that both graphs are now graphed out in an identical manner.

Other than splicing audio, normalization of audio was also looked into. Using FFmpeg's normalizer, a commandline argument was ran to attempt to normalize both the spiced user pronunciation and the spliced professional pronunciation. The issue that arose here was that in some cases, the normalization may not function due to the audio stream being shorter than 3 seconds. The output of the normalization were also graphed to produce two sets of graphs for users to compare.

7.8 Stage 8 - Spectrogram Analysis

Two different functions for spectrograms were made, the first one plotted using MatPlotLibs specgram function and the second one using pcolormesh to plot. Using pcolormesh was an attempt to manually create the spectrogram by filling colors in a rectangular grid based on the initial function provided. This was experimented on before but was replaced with the specgram function. Spectrograms for both the regular audio data and normalized audio data was tested and results as follows in Figures 7.4 and 7.5.

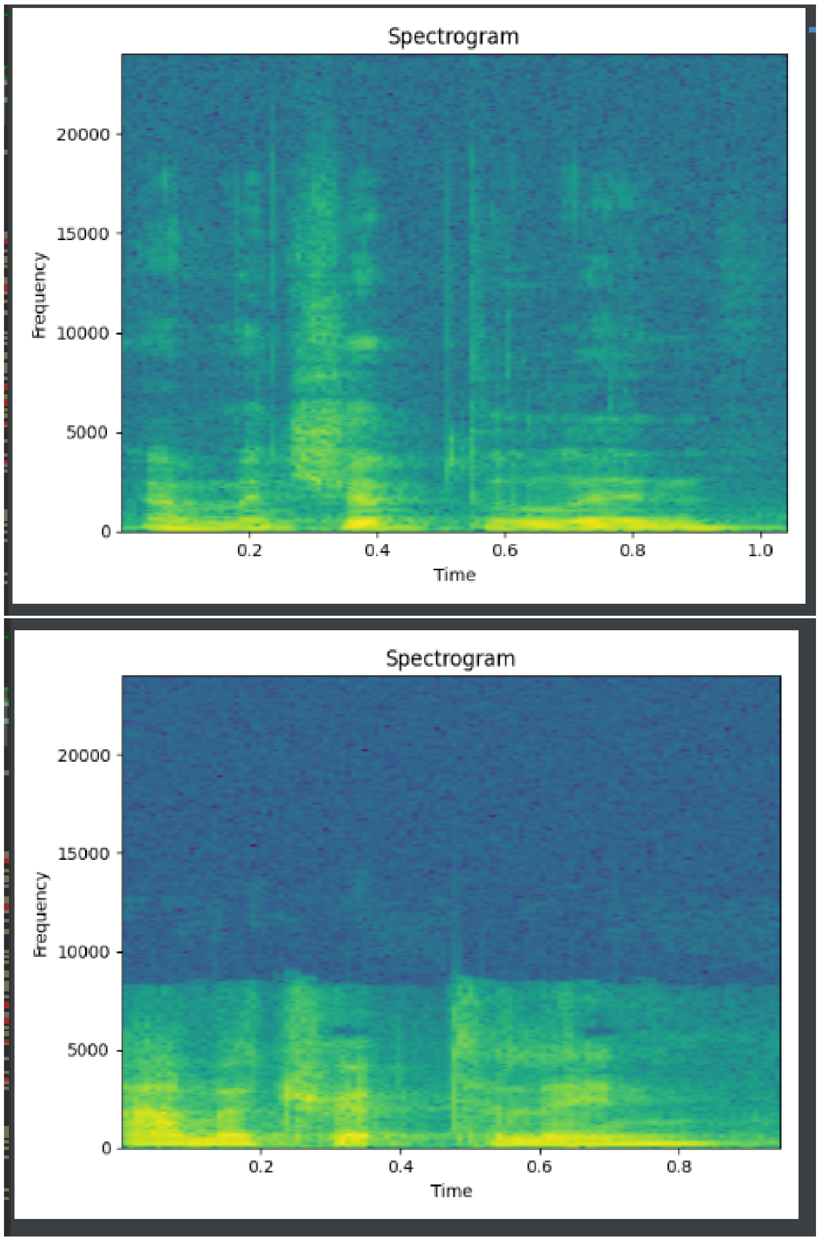


Figure 7.4: *Spectrogram for Amitriptyline audio. User (Top) and Professional (Bottom)*

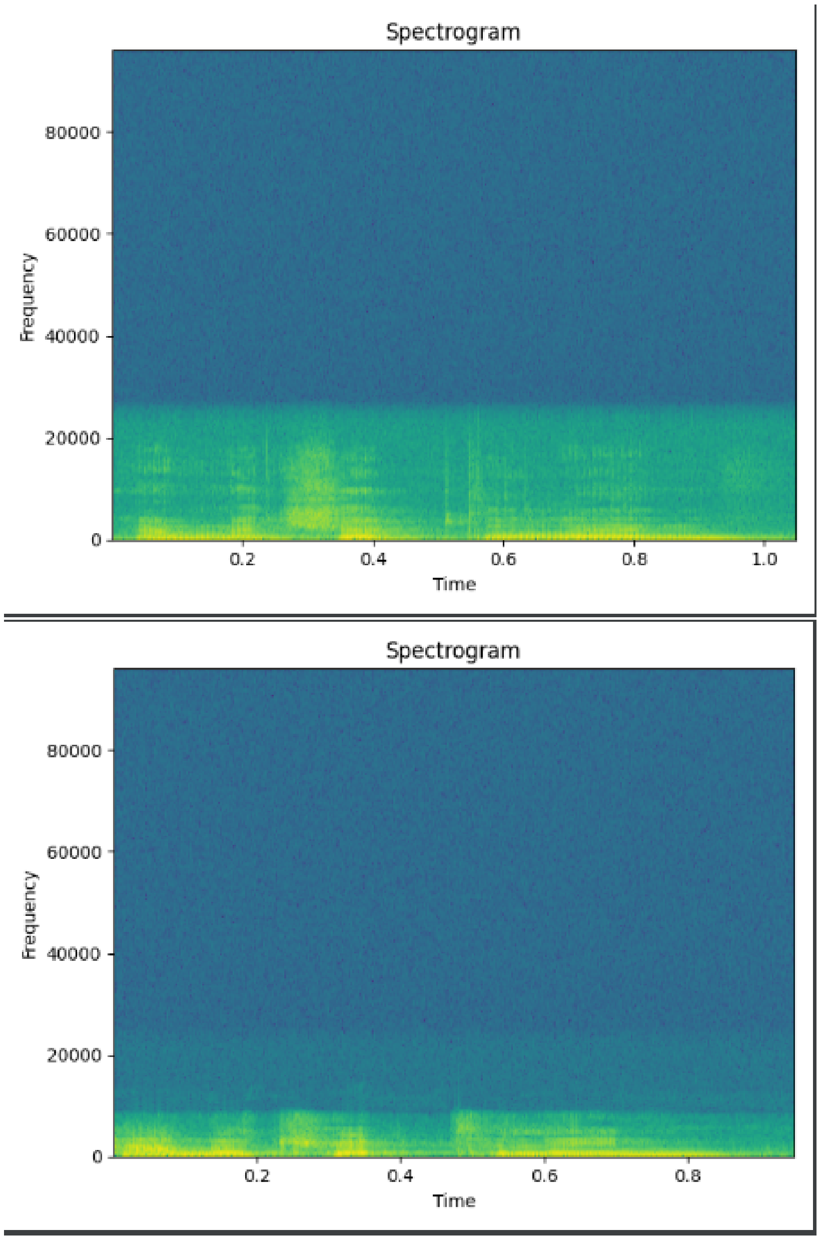


Figure 7.5: *Spectrogram for normalized Amitriptyline audio. User (Top) and Professional (Bottom)*

As seen in the figures above, the user spectrograms have higher frequencies compared the professional spectrograms which is caused due to the microphone that I was using. Currently, the spectrograms serve no purpose for the user as it is not possible to estimate the difference by examination. At the time of investigating spectrograms, it was discovered that there was a use for spectrograms, audio fingerprinting. Baluja and Covell (1888) presents a system called waveprint which reads spectrograms and identifies patterns which are used to categorize audio with fingerprints and allows for comparison with other audio files. Although spectrogram analysis was not included into the final application, future research on this section can be done in order to attempt to fingerprint segments of audio and pattern match them when recording user audio. Audio fingerprinting is a method used in applications that detect music from snippets of audio, like

Shazam, this is harder to implement here due to the relatively small audio files. The shortest audio file being 1 to 2 seconds long, it would be unreasonable to audio fingerprint the length of audio and the peaks and troughs may not be very distinct for a large vocabulary of words. PyAcoustid was a library that was researched on which uses a fingerprinting library called Chromaprint. An attempt to intergrate the audio fingerprinting was done but was unsuccessful due to, as mentioned previously, audio files being too short to provide a unique fingerprinting. A manual attempt at peak detection was investigated upon which used blob detection and averaging to find peaks in the spectrogram. In the non-normalized case (Figure 7.4), using blob detection and then normalizing the blobs could make for a fair comparison between both blobs. Blob detections tools can be found in OpenCV's Blob Detection.

7.9 Stage 9 - GUI Implementation

Setting up the GUI required the use of tkinter for adding the GUI and Python Imaging Library (PIL) for adding the graph images. The GUI can be seen in Section 6.2. The first interface added was the list box for all words in the database. The names for the words which are displayed is taken from the name column from the database. The play button was implemented next which will allow for the user to play the professional recording for the currently selected word in the list box. Following this, the feature which allows the user to type into the list box and when clicking on the arrow to list the words, it returns all words from the database which contains the current words in the list box. The category list box was implemented next along with a select category button which would the only display words which was in the currently selected list box. A category renaming function was implemented to allow users to modify the category of individual words. This can be done by first selecting the word, typing in the name of the category you want it to have in the text box next to the "Rename category" button and then clicking the button. Since the categories are not automatically updated, a reload category button was implemented. A feature that was deemed to be useful in the coming future when more terms are to be added is the "Add file" button. This accepts WAV and M4A files which will then copy the file into the folder containing all the professional terms. An "Update Database" button was included which should only be used when adding new files to the application as it will reprocess the database along with adding the word name for the newly added file. For example, when adding a WAV file called "kraken.wav" and it contains the pronunciation of "Tuesday", when updating the database after the file is added, the primary key for the entry will be "kraken.wav", it's name that will show up in the list of words will be "Tuesday" and the category will be empty and the user can manually update its category. The "Rec word" button was implemented last which will record 5 seconds of audio from the user, create a new "Results" window. It will then compare both user's pronunciation with the currently selected professional pronunciation. It will then splice the both audio files to remove trailing noise and plot the newly spliced wave graphs for both files. It then uses FFmpeg's normalize function to normalize both words and plot the wave graphs for the normalized audios. All 4 graphs will then be displayed in the "Results" window along with affirmation if the pronunciation is correct or wrong (if it is wrong, it will return what the STT thinks you said) and two extra buttons, one to replay the professional's pronunciation and one to replay the user's own. A button was later added to delete a word from the database

7.10 Stage 10 - Splitting Word By Syllable

Blevins (1995) wrote in a book on phonological theory that native speakers of languages have clearer pronunciations due to their ability to identify where the syllables break up. This syllables breaking might not be second nature to a non-native and therefore would be a welcoming implementation for the pronunciation tool. Initially, each word was broken down manually into syllables but this approach will be ineffective when proceeding on with the implementation of the audio file adder. Thus, syllable splitting libraries were investigated to find an suitable implementation. Two hyphanators were considered; hyphenate and pyphen. The first, hyphenate was based off Liang's (1983) algorithm which can be found in his paper on word hyphenation. This was not used due to some issues with the more medical terms. One example in testing the hyphenation using "ursodeoxycholic acid" provided the output "ur-sodeoxy-cholic acid". Other examples include "em-pagliflozin" and "cephalex-in". Pyphen was used instead which contained multiple dictionaries from different languages and, from testing, had a decent hyphenation track record with the scientific terms. Using the same three examples produce "ur-so-deoxy-cho-lic-a-cid", "em-paglif-lo-zin" and "cep-ha-lex-in" which match the pronunciation with decent success. After the user click the "Play word" button, the word will be passed into the hyphenator and will be displayed for the user to see.

7.11 Stage 11 - Executable

An executable was create using PyInstaller which creates an executable based off a python script. This includes all modules and libraries required to execute the script. An issue was encountered here with missing pyphen files but was corrected by editing the spec file to collect the data files generated by pyphen. The executable runs fine but a issue it faces is that FFmpeg's normalization is in command line mode and as such is not bundled when PyInstaller is used and as such the executable created contains slightly less functionality. This can be seen in Figure 7.6 where new normalised graphs are not generated and only shows the most recent graph.

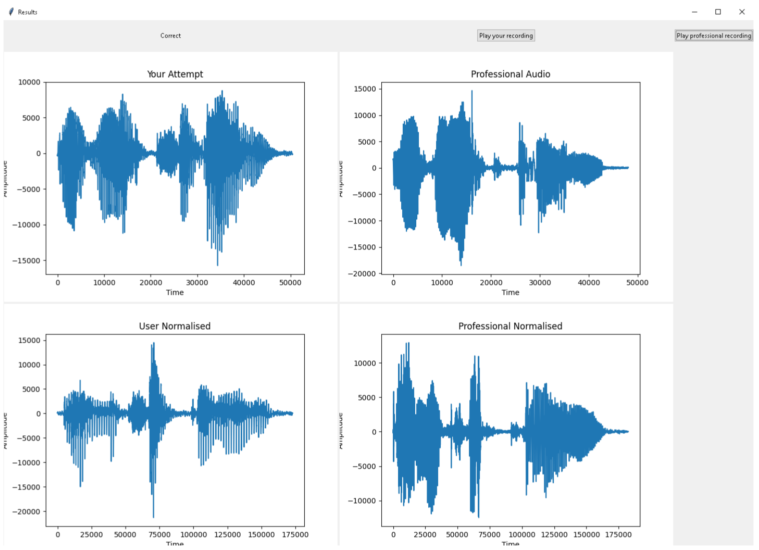


Figure 7.6: *Top graphs are working as intended but the bottom normalized graphs do not output properly*

7.12 Overview

The pronunciation tool stores professional pronunciation recordings in a database and users are able to playback these files. Word splitting is included to allow for users to visually understand the pronunciation. User's are able to record their own pronunciation and a limited functionality comparison is done with the professional term and provides 4 graphs of the pronunciation along with the result of the comparison and two buttons for extra playback of the user and professional pronunciation. User's are also able to modify the categories of individual words and user's can add additional WAV or M4A files to the database.

Chapter 8

Application Testing

8.1 Online Testing

Initially, the application was intended to be a web-based application to allow for participant testing despite being in a remote location. This has regrettably been altered and was not until the final few weeks that the executable was implemented.

The initial plan was to provide the tool, by sending the participants the web-based application along with a set instructions of how to proceed with testing. Tester's would be provided with a word to find using the tool, listen to its playback and proceed to record their own pronunciation. This was to be attempted 5 times and the outputs would be collected. Alongside this, a questionnaire would be provided to the tester which each question contained an ordinal rating scale from 1 to 7, 1 being mostly disagreed and 7 being mostly agreed followed by two open ended questions which was optional to complete. The questions were as followed along with the purpose of the question (in brackets):

- How would you rate your British pronunciation?
  (Allows for a rough estimate if the user has a British English speaking background)
- Are you familiar with the word you were provided?
  (Helps gauge if the tester has had familiarity with word in the specific field)
- Did you struggle with pronouncing the word before using the tool?
  (Identify if participants find word difficult)
- Would you download this application if it was available?
  (Checks the desirability for the tools usage)
- Did the waveform graph help you locate areas in pronunciation to improve?
  (Identify if the graphs were useful for participants and if it is worth being kept or scrapped)
- Did the hyphenation of word into syllables help you with the pronunciation?
  (Measure the effectiveness of the hyphenation)
- Did you feel that you pronounced the word better after using the tool?
  (Determine if the tool benefitted the user)
- Overall experience with application
  (Evaluate user satisfaction)

- What did you like or dislike about the application?
  (Open ended - provides feedback on what the application did well and what should be changed)
- What would you like seen implemented as an addition to the application?
  (Open ended - provides more ideas for future features)

I am fortunate to still have two flatmates that were able to assist me in the testing of the application. Both being non-UK native and one having been trained in the medical field. They were provided with the application in person for testing the ease of use and providing feedback. No formal testing was conducted upon them.

The aim for user testing was to introduce the application to a variety of participant of different backgrounds, different voice types, genders and audio input device. The diversity of conditions would allow for the full limits of the application to be tested and the feedback would allow for the further tweaking of the application.

8.2 Personal Testing

Personal testing using the application was conducted for all 15 words in the pronunciation tool to try to analyse errors in the program that may occur. Being a Chinese born Malaysian, I have had experiences with the pronunciation of words in other languages such as Malay and Chinese although English is the language I use on as my first language. In my initial testing of the words, the words that I found hardest to pronounce were clarithromycin, empagliflozin and hypergammaglobulinemia. When searching for the words in the application, it was noted that hypergammaglobulinemia would be difficult to remember the correct spelling and the additional feature in which the search box shortens based on text inputted helps alleviate the spelling issue. When searching for hypergammaglobulinemia, I would type gamma into the search bar along with clicking the drop down menu which will provide word as seen in Figure 8.1.

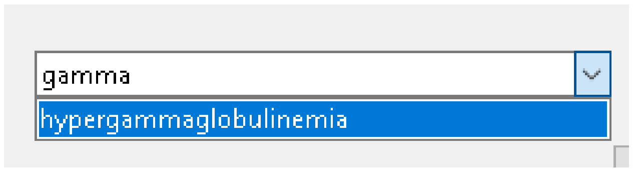
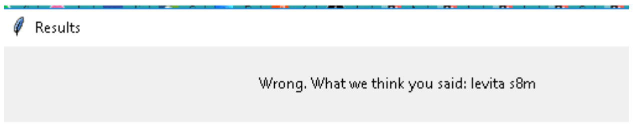


Figure 8.1: *Results for typing gamma in search bar*

The initial testing of most terms produced desirable results. As mentioned before, during testing it was noted that the word levetiracetam had the most issues in the tool. When the professional recordings for levetiracetam were tested with Google's Speech Recognition, it was discovered that both recordings outputted ”lever to Atherton” and ”level 2 Atherton”. From my pronunciation testing, some of the outputs include ”North Reston”, ”lavatory cistern” and ”levita s8m”, which can be seen in Figure 8.2. These results were concerning due to the sheer variation of outputs when trying to pronounce the same word. One of the reasons this could be occuring is due to the STT API not having the word in its online dictionary and is trying to guess what was being said. In the last case though, it was very surprising to see ”levita s8m” as that could not possibly be a possible word combination. When there are issues with the STT for a word, as a collateral, the hyphenation will also produce poor results.



Wrong. What we think you said: levita s8m

Figure 8.2: *Errors when pronuncing levetiracetam*

Although clarithromycin and empagliflozin were words that initially were difficult to pronounce, after interacting with the words during the project, I am able to now pronounce the words into the application with a better rate of success than before. The graphs help with intonation of the syllables and the hypenation makes it easier to break up the phrases before pronouncing. As for hypergammaglobulinemia, my own testing with the word produces varying results. The difficulty comes in pronouncing the ”globulinemia” portion of the word. Although the hypenation does help which splits the term into ”glo-bu-li-ne-mia”, the problem may come from my innate inability to pronounce ”globu” properly and the word comes out as ”globlu”. More testing on the word proved this to be the case and by slowly using the hypenation provided, the word was able to be pronounced at a higher positive rate than usual.

8.3 Takeaway From Testing

Google's Speech Recognition API produce some promising results, and at times the API was produced results too positive for the good of the application. The aim of the application was to produce a system to allow for accurate and fair comparison for pronounciation improvement. With Google's Speech Recognition API, false positive (when the wrong word was spoken but the STT API deemed it correct) and true negatives (vice versa) have been observed occasionally which was pointed out in the previous section and in Section 7.5. Other than that, syllable splitting provides an additional layer in assisting user's when pronouncing words due to the visual aid provided. The ability to autofill the search bar based of small portions of the word allowed for faster searching since words could be searched by using it's syllables.

The lack of a larger testing group reduced the ability to view the full capabilities of the application and as such would need to be worked on in the future. For future testing, the target audience should be examined and recognized for the application to ensure a majority of the testers are from a group that will benefit from the application usage.

8.4 Demo

A demonstration for the application was included here [1] [2] [3]

Chapter 9

Evaluation & Critical Appraisal

9.1 High Priority Requirements Review

9.1.1 Requirement 1 - Storage and playback for the professional pronunciation of scientific terms

All 15 of the professional pronunciation recordings were stored in a folder in the program, can be manipulated using SQLite, all M4A are also converted into WAV files and playback works on each of the WAV file.

9.1.2 Requirement 2 - Record and store user's attempts at the word pronunciation

User's recordings have also been achieved. These recordings are in the same format as the professional audio to make for fair comparison. The users attempts are stored in a separate folder.

9.1.3 Requirement 3 - Compare user's audio file against the professionals and score the attempt

The users audio file is compared with the professional audio using Google's Speech Recognition API and the user is prompt if they have pronounced correctly or incorrectly.

9.1.4 Requirement 4 - Provide feedback on how to improve pronunciation

Feedback was provided via the use of graphs to allow users to examine the audio files graphically.

9.2 Low Priority Requirements Review

9.2.1 Requirement 5 - User-friendly UI (Web based front end)

A web-based application was not implemented but a GUI was set up for the application along with an executable which contains all libraries required to run it.

9.2.2 Requirement 6 - Search function which can autofill

An autofill function was successfully implemented.

9.2.3 Requirement 7 - Group pronunciation by respelling

Grouping by syllables and pronunciation respelling was not implemented.

9.2.4 Requirement 8 - Group terms based on how often they are used in tandem

This requirement was also not implemented due to ambiguity, The scale of being able to relate how often different words are used together made this requirement unreasonable. To replace this, the a category section was implemented for the word searching.

9.2.5 Requirement 9 - Recommend other words based on the words attempted

This was initially envisioned to recommend other words that may help with pronouncing a certain word. An example of this would be pronouncing hypergammaglobulinemia and if the user was having trouble, the program might offer the user to attempt the word gamma, or if the user struggles in the ”nemia” portion, the program may suggest ”anemia”. This was not implemented as a larger portion of words would be needed.

9.2.6 Requirement 10 - Stored professional pronunciation should be normalized

Professional pronunciation and user pronunciation were normalised using FFmpeg.

9.2.7 Requirement 11 - Statistical records for comparison

No statistical record was kept but the most recent recording of audio files are kept and overwritten when a new attempt is made.

9.3 Critical Evaluation and Improvements

9.3.1 Using Python over Java

As discussed previously, the choice of programming language was swapped from Java to Python due to the libraries present in Python along with the ease of implementation of the STT API in Python. If the project was to be done again, more time would have been spent in researching on better API's for Java. Another approach that could have been taken was the additional research into JavaScript which would allow for an easier implementation of a web-based application which would allow for cross-platform usage of the application. This would further allow more testing to be carried out on the application due to the ease of use of the application for participants.

9.3.2 Scoring System

Currently the user is only notified if either they have pronounced the word correctly or wrongly. An area which could be investigated on a small scale would be the incorporation of the Levenshtein Distance which measures the distance between two words. A study in 2014 used the Levenshtein distance to measure the difference between native American English from non-native (Wieling et al., 2014). It would be possible to extend the Levenschtein distance to incorporate syllables splitting which can be followed up by calculating the distance for individual syllables and provide a scoring system. Other than that, as mentioned previously, if audio fingerprinting can be extended to handle smaller audio files, it will enable for the word comparison to use audio fingerprinting as a measure to compare accuracy of audio files. Also on a side tangent, as mentioned by Lee's (2015), words of discouragement may lead to a decrease in performance by L2 students and as such it would be of best interest not to flat out state that the user pronounced the word wrongly, rather the wording could be changed to "Good attempt but we thought you said: ".

9.3.3 STT API

The current STT API have been working with an impressive degree of accuracy in estimating the word from an audio file. Although this is usually a positive feature, its use in the application leaves room for desire in the case where a slightly mispronounced word is provided but the API corrects the word and instead of providing the word that was said, the API provides the word that it thinks was said. A method for future projects could include the use of hyphenation on the STT to split the word pronounced before changing it into text and so the comparison will be done tuple-wise, in which the tuples are the syllables. instead of as whole words.

9.3.4 Clarity of requirements

Some requirements were initially considered but were later dismissed due the the lack of clarity or due to the scale of the idea. An example being requirement 8 which was design to allow the user to group words in tandem. What does this mean? Would an example for this mean "paracetamol" will be grouped with terms like "medicine", "purchase", "from" and "hospital"? The other unclear requirement being requirement 9 which was used to recommend other words based on the words attempted. How would other words be recommended? Would it be based of the category of the word? Would the words recommended be based of the syllables in the word originally attempted? These requirements should have been flushed out and clarified in more detail when the functionality of the application was still in its initial stages.

9.3.5 GUI improvements

The current GUI is basic implementation due to it being the first attempt at creating a GUI in Python. Not much was known beforehand on the process that would be involved when creating it. More time would be spent into improving the looks of the GUI to make it more attractive for participants to use. Other than that, the graphs that were produced in the GUI were just png files and they could be extended to allow the user to manipulate the graphs. It would also be possible to use the spectrograms created to further provide additional analysis for users.

9.3.6 More participants for user testing

Due to the time constraints, the final product was not able to be pushed out into the hands of participants in time thus additional user feedback was unable to be obtained. Along with the creation of a web-based application as mentioned in Section 9.3.1, the availability of the program would allow for more participants to test the application.

9.4 Reflections of the development process

The development of the application provided additional insight to the factors that are considered when creating an application for the use of assisting a targeted group of people. Background research provided a baseline to begin investigating the topic at hand, that being the creation of an application to assist non-native British speakers in pronouncing scientific terms. In addition to the research done on the background of pronunciation learning, investigations on the functions and features that similar applications and programs contain is an essential skill to understand the needs of the users. A large portion of investigation midway went into the research on audio fingerprinting and blob analysis in an attempt to create a suitable scoring system for the application which was unsuccessful. Another issue came in the fact that it was impossible to create proper streamlined categories for every word provided and a workaround this was to provide the user with the tools to change the categories themselves. Additional columns for the database were never realised, again due to the lack of a scoring system.

While there were drawbacks, it was attempted to make the program user-friendly and it can be said that the program does this well providing, to an extent, freedom for the users to edit categories, add words or remove words from the database. Although the application is not web-based, a executable was able to be created and ran on windows platforms which allowed for some users to be able to use the application without having to download all the dependencies and libraries.

The Agile approach to software development was also a good step towards a healthy development process. This was initially built up in the Junior Honors project and now further refined to accommodate the solo project. The constant communication and flow of ideas with the supervisor also allowed for some additional discoveries which improved upon the application development.

Chapter 10

Conclusions

10.1 Project summary

In this project, a pronunciation tool has been deployed that allows for the user to store and playback professional audio recordings. Alongside this, users are able to record their own pronunciations to compare with the professional audio files. The users are provided with graphs that show waveform analysis for users to identify errors in their own pronunciation. Other functionality included: an SQLite database, GUI and STT recognition. The employment of Agile development allowed for proper resource and time management thought the project, although this became blurry at the end due to some stages taking too long. This report intends to provide an overview of the project as a whole, along with some discussion behind the design and to demonstrate the implementation of the requirements.

10.2 Final conclusion

Communication is essential in STEM fields due to the complexity of the terminology and the degree of accuracy required by those in the field. The application at hand currently contains a majority of medical terms but can be extended further for the other fields. The application hopes to allow non-native speakers the opportunity to develop skills that are essential and are required in such lines of research. Along with the implementation of the majority of the requirements. Additional features can still be in improvised to produce a clearer, higher functioning application. Although the application has a long way to go before being able to be deployed to professional who might find the additional aid beneficial, this application hopes to provide enough context in the correct path to proceed upon for those who aim to further improve and innovate the pronunciation teaching in the STEM field.

Chapter 11

Bibliography

Auditok. Retrieved 11 April 2021, from https://auditok.readthedocs.io/en/latest/

Baluja, S., Covell, M. (2007). Audio Fingerprinting: Combining Computer Vision Data Stream Processing. 2007 IEEE International Conference On Acoustics, Speech And Signal Processing - ICASSP '07. doi: 10.1109/icassp.2007.366210

Blevins, J. (1995). The syllable in phonological theory. In Goldsmith (ed.) The Handbook of Phonological Theory, 206-244

Chromaprint. Retrieved 11 April 2021, from https://acoustid.org/chromaprint

CMUSphinx. Retrieved 11 April 2021, from https://cmusphinx.github.io/wiki/tutorialsphinx4/

Duncan, S. (1983). Cheap Ship Trips: A Preliminary Study of Some English Phonological Difficulties of Language-Minority Children and Their Relationship to Reading Achievement. Bilingual Education Paper Series, Vol. 7, No. 4. (ED255048). ERIC. https://files.eric.ed.gov/fulltext/ED255048.pdf

FFmpeg. Retrieved 11 April 2021, from https://www.ffmpeg.org/

Forvo. Retrieved 10 April 2021, from https://forvo.com/

FreeTTS. Retrieved 11 April 2021, from https://freetts.sourceforge.io/

Gilakjani, A. (2012). The Significance of Pronunciation in English Language Teaching. English Language Teaching, 5(4). doi: 10.5539/elt.v5n4p96

Google Speech-to-Text. Retrieved 11 April 2021, from https://cloud.google.com/speech-to-text/docs/sync-recognizespeech-sync-recognize-python

hyphenate. Retrieved 11 April 2021, from https://pypi.org/project/hyphenate/description

Java Sound Libraray. Retrieved 11 April 2021, from https://docs.oracle.com/javase/8/docs/technotes/guides/soun

JavaSpeechAPI. Retrieved 11 April 2021, from http://jsapi.sourceforge.net/

Koren, S. (1995). Foreign language pronunciation testing: A new approach. System, 23(3), 387-400. doi: 10.1016/0346-251x(95)00033-g

Liang, F. (2008). Word Hy-phen-a-tion by Com-put-er. url: http://www.tug.org/docs/liang/liang-thesis.pdf

Neri, A., Mich, O., Gerosa, M., Giuliani, D. (2008). The effectiveness of computer assisted pronunciation training for foreign language learning by children. Computer Assisted Language Learning, 21(5), 393-408. doi: 10.1080/09588220802447651

OpenCV Simple Blob Detection. Retrieved 11 April 2021, from https://docs.opencv.org/3.4/d0/d7a/classcv_1_1Sin

playsound. Retrieved 11 April 2021, from https://pypi.org/project/playsound/

Processing. Retrieved 11 April 2021, from https://processing.org/

PyAcoustid. Retrieved 11 April 2021, from https://pypi.org/project/pyacoustid/

pydub. Retrieved 11 April 2021, from https://github.com/jiaaro/pydub

PyInstaller. Retrieved 11 April 2021, from https://pyinstaller.readthedocs.io/en/stable/index.html

pyphen. Retrieved 11 April 2021, from https://pyphen.org/

python-sounddevice. Retrieved 11 April 2021, from https://python-sounddevice.readthedocs.io/en/latest/

Rabiner, L., Juang, B. (1986). An introduction to hidden Markov models. IEEE ASSP Magazine, 3(1), 4-16. doi: 10.1109/massp.1986.1165342

Sharma, A. (2019). Python script to convert m4a files to wav files. url: https://gist.github.com/arjunsharma97/0ec

Shazam. Retrieved 10 April 2021, from https://www.shazam.com/

simpleaudio. Retrieved 11 April 2021, from https://pypi.org/project/simpleaudio/

Speechace. Retrieved 10 April 2021, from https://www.speechace.com/

SpeechRecognition. Retrieved 11 April 2021, from https://pypi.org/project/SpeechRecognition/

Thomson, R. (2011). Computer Assisted Pronunciation Training: Targeting Second Language Vowel Perception Improves Pronunciation. CALICO Journal, 28(3), 744-765. doi: 10.11139/cj.28.3.744-765

tkinter. Retrieved 11 April 2021, from https://docs.python.org/3/library/tkinter.html

Voce. Retrieved 11 April 2021, from http://tyl.st/projects/voce/

WebRTCVAD. Retrieved 11 April 2021, from https://github.com/wiseman/py-webrtcvad

Wei, Y., Zhou, Y. (2002). Insights into English Pronunciation Problems of Thai Students (ED476746). ERIC. https://files.eric.ed.gov/fulltext/ED476746.pdf

Wieling, M., Bloem, J., Mignella, K., Timmermeister, M., Nerbonne, J. (2014). Measuring Foreign Accent Strength in English. Language Dynamics And Change, 4(2), 253-269. doi: 10.1163/22105832-00402001

Yang, M. (2005). Nursing Pre-professionals' Medical Terminology Learning Strategies. Asian EFL Journal, 7(1), 137-154. url: http://asian-efl-journal.com/March_05_mny.pdf

Appendix A

Setting up the Pronunciation Tool

Step 1
Download the Pronunciation Tool from MMS. Proceed to Step 2 if you want to manually install the application. If not, proceed to Step 4.

Step 2
Download Python 3.8 along with the following libraries and go to Step 3:

- os
- Sounddevice
- Pydub
- SciPy
- SpeechRecognition
- Matplotlib
- auditok
- contexlib
- wave
- sqlite3
- tkinter
- PIL
- pyphen
- shutil

Step 3
In the PronunciationTool folder, run main.py

Step 4
Instead of downloading Python 3.8 and the libraries, just double click the main.exe file (Only on windows)

Appendix B

Ethics Approval

University of St Andrews | FOUNDED 1413

School of Computer Science Ethics Committee

29 January 2021

Dear Lim,

Thank you for submitting your ethical application which was considered at the School Ethics Committee meeting on Monday 11th January.

The School of Computer Science Ethics Committee, acting on behalf of the University Teaching and Research Ethics Committee (UTREC), has approved this application:

Approval Code:	CS15232	**Approved on:**	29.01.21	**Approval Expiry:**	29.01.26
Project Title:	Scientific Terms Pronunciation Tool				
Researcher(s):	Lim Jin-Han				
Supervisor(s):	Dr Ishbel Duncan				

The following supporting documents are also acknowledged and approved:

1. Ethical Application
2. Participant Information Sheets
3. Participant Consent Form
4. Advertisement

Approval is awarded for 5 years, see the approval expiry data above.

If your project has not commenced within 2 years of approval, you must submit a new and updated ethical application to your School Ethics Committee.

If you are unable to complete your research by the approval expiry date you must request an extension to the approval period. You can write to your School Ethics Committee who may grant a discretionary extension of up to 6 months. For longer extensions, or for any other changes, you must submit an ethical amendment application.

You must report any serious adverse events, or significant changes not covered by this approval, related to this study immediately to the School Ethics Committee.

Approval is given on the following conditions:

- that you conduct your research in line with:
  - the details provided in your ethical application
  - the University's Principles of Good Research Conduct
  - the conditions of any funding associated with your work
- that you obtain all applicable additional documents (see the 'additional documents' webpage for guidance) before research commences.

You should retain this approval letter with your study paperwork.

Yours sincerely,

Wendy Boyter
Ethics Administrator