

SOFTENG 787 – Project Report

Visualizing Pronunciation Feedback

Li Li (lli380)

Supervisor: Dr. Catherine Watson

Faculty of Engineering

University of Auckland

Abstract

The project created a Maori Pronunciation Aid by integrating two early modules and adding some extra functionalities. Descriptions of the two early modules are provided and then the design and development, as well as the evaluations of the new language aid is presented. A graphical user interface was developed using C# to wrap around the speech recognizer process with the command line, a volume bar was added along with features to allow multiple users to store their progress. Then, the software development process of MPai was addressed, and the design aspects were also covered. Finally, the known defects were addressed and advice for future work was made.

1. Introduction

Nowadays, with the requirement of learning a second language. There are varieties of language aids. In the past, the most common form of language aid was audio tapes and textbooks. By listening to the sample recordings of prerecorded by native speakers. The student is able to understand the dialogues as well as imitate how the native speaker speaks. But the drawback of this approach was, the student cannot evaluate their performance without the help of their teachers.

In recent years, with the fast development of information technologies. A new form of language aid rises, this new approach is based on computers and it is known as Computer based language aid. Computer-based language aid, in its sense, can be perceived as a computer software which provide language learning support for the language learners. With the help of the computers, the design of the new forms of aid can be various. Since the computer can provide multimedia support for its applications. The new computer based language aid can have multiple features such as sample audio, sample video, as well as some coaching services

under which the student can get how much right he has performed. For instance, an ordinary language aid can play sample pronunciations for certain words or vowels. Therefore, by learning and imitating the sample pronunciation, the user could grasp the correct pronunciations. Moreover, a language aid can also help check the correctness of the users' pronunciations, and give a feedback to the users' input. The feedback could be in various forms. It could be a score, a visual chart, or a data table. Aimed at informing user of their performance. There are several examples of language aids. For example, there is an English-learning tutoring system, named as "EyeSpeak" developed by Visual Pronunciation Software Ltd. This application is able to present audio and visual feedback to the users and uses speech recognition system to evaluate the user's performance [5], also the application is able to interact with the user and assist the user with their learning process.

MPai which full name is Maori Pronunciation Aid, is a computer-based application used to assist the Maori language learners with their language learning process. This project is led by Dr. Catherine Watson, and there are two sub projects done in the past at the University of Auckland.

There are several MPai projects. One of the first versions of MPai was developed by Gutla. J. J. [4] and Daniel T Rivers [5]. This application was written using Java. By taking the advantage of multimedia technologies provided by the computer. This aid is able to perform speech recognition and provide aural feedbacks as well as visual feedbacks to the users. This application used Hidden Markov Model Toolkit as the speech recognizer.

Later, another project is the Formant Plot [1], this application is able to provide real-time feedback to the user on their pronunciation of Maori vowels. Just as its name indicates, the application first draws several ovals on a canvas, represents the correct range of the vowel pronunciations. When the user speaks a vowel to the application, the application

will be able to plot the user's pronunciation to the canvas. Then, if the pronunciation is correct, the users plot will be inside the ovals. In this way, the user will be able to adjust their pronunciations and make precise pronunciations.

Then, in another project, a new module for MPAi was developed [2], this application is able to respond to the pronunciation of Maori words. When the user speaks a Maori word to the application, it will be able to analyze the pronunciation and quickly give the user feedback regarding to the correctness of their pronunciation. However, this was only accessible via the command line in MS-DOS or Linux operating system. It had no user friendly graphical interface. Due to limited time and resources, this project has not yet been linked with the formant plot.

This report will focus on the new MPAi – an extension to the old MPAi where the new one combined the previous work of both Hui [1] and Lu [2]. The new MPAi is constructed in C# and brings the MPAi with a brand new user interface as well as several new features. At the same time, the new MPAi remained consistent with the old MPAi – it integrated both the formant plot [1] and HTK speech recognizer [2] together, and efforts are made to make sure the new MPAi can do exactly as the old MPAi did, and can do better.

The report is in two parts, the first part of the report will provide some background information of MPAi as well as the previous work, and the new features being added to MPAi. The first part is for non-technical audiences to get a general idea of this research project, and efforts has been made to make sure the technical details are conveyed in a proper way so that everyone could understand. Then, the second part is designed for the technical audiences to prepare them well to take over this research project in the future. In the second part (in the appendix), the report will provide a detailed technical documentation of the new MPAi for future students or developers who is going to take

over the project, and make sure all the technical details are covered.

2. Goal

The overall goal of this research project is to continue the development of MPAi. Let's first look at the context for this project.

As mentioned before, this application consists of two parts, the formant plot and the pronunciation aid. Before the onset of this research project, the formant plot was fully functional and the accuracy is good, therefore no more work to do for the formant plot. However, the pronunciation aid, was in its conceptual stage and had not been implemented. The previous student who worked on the MPAi – Hui has recommended that more research should be carried out towards the HTK (the engine of the speech recognizer) to get the Pronunciation Aid fully functional. In addition [2], in his report, he also mentioned that the MPAi has got some bugs that needed to be fixed.

A year later, another student of Dr. Watson – Lu carried out the research towards the HTK. She used the HTK toolkit to create Hidden Markov Models (HMMs) and made a word recognizer [2]. To train the HMMs, Lu used Maori word recordings from MAONZE project. The MAONZE project (Māori and New Zealand English) uses recordings from three sets of male and female speakers to track changes in the pronunciation of Māori and evaluate influence from English [6]. Although Lu made the word recognizer, she did not link the recognizer with the MPAi. In her report, she advised that the accuracy of the word recognizer could be improved, and then links to the MPAi in the future [2].

Therefore, in terms of these, after several meetings with my supervisor Dr. Watson and her team. We agreed on the following goals for this research project.

- 1) Resolve the issues existed in the MPai and improve the user interface
- 2) Integrate the HTK word recognizer into MPai, and produce an accessible none technical graphical user interface
- 3) Design and implement a new report page as a feedback to the users detailing the correctness of each of their pronunciations
- 4) Add a user management module to the MPai, that is to say, the MPai will be able to provide services to different users and create separate account to store the users' data
- 5) Add the feedback option to the MPai, therefore the user will be able to voice their valuable feedbacks.

3. Design & Development

This section will cover the new features I added to the new MPai and the technical aspects of the features.

3.1 The design of the new MPai

The new MPai is actually a hybrid application, using both C# and Python, where C# is used to construct the user interface, therefore the new MPai will be able to enjoy native support by Windows. Then, the old Python MPai was trimmed and integrated into the new C# MPai, only the formant plot was preserved.

Here is the rationale of using the new hybrid architecture.

One major reason, it was difficult to fix the existing Python code. Since the old MPai was written using Python, and it is running in Windows. However, Python has limited functions on creating user interfaces, and in fact Windows cannot provide native support for Python applications. It was inevitable that the old MPai had some defects. One major defect is that, the sound produced by the old MPai was sometimes broken in Windows 7 and

later versions of Windows. After a close investigation into this issue. It was found that the Python code was fine, but the defect actually stems from Windows. Starting from Windows Vista, Windows has used a totally new core – the NT6. That was a brand new core and it introduced a new driver model. However, since the third party libraries used by the Python code was based on the old driver model that prevails in NT5 – the core for Windows XP. It was not supported well by the new driver model. Therefore, the sound produced is sometimes broken under the new Windows core.

See figure 3.1.1 and 3.1.2 below for the comparison of the old and new MPai.

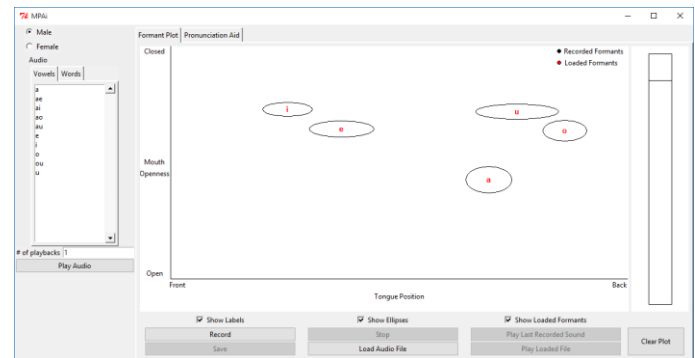


Figure 3.1.1 – the old MPai

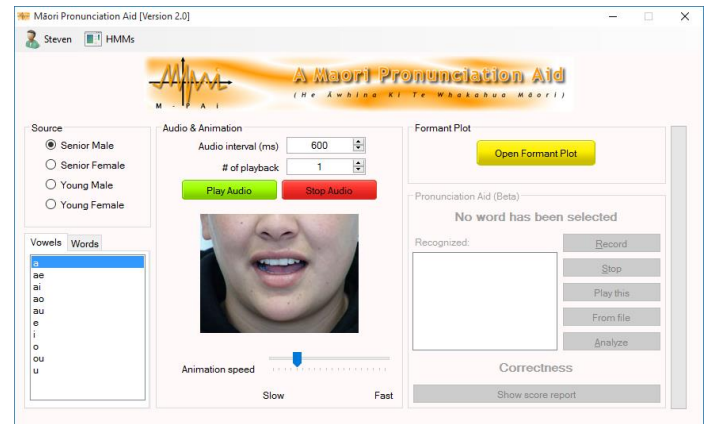


Figure 3.1.2 – the new MPai

From the screen shot of the new MPai, it can be seen that the user interface has changed drastically, but the main features have remained consistent. The formant plot has been incorporated into the group box in the top right corner, and can be invoked by clicking on the button “Open Formant Plot”.

The improvements brought by the new MPai are as follows.

- 1) Resolved the broken-sound issue, the new MPai will never produce a broken sound
- 2) More fine-grained selection of audio sources, the user will be able to define the audio source, since MPai has a database for different age group of speakers, the new MPai has enabled the options for the users
- 3) A controller for the animation, now the users will be able to control the speed of the animations
- 4) All known bugs are fixed

3.2 Linking the word recognizer with the new MPai

By studying Lu's report [2], a way was found to use the word recognizer. When the .wav recordings are ready, the first step is to transform them to .mfc files, so that the recognizer could use. Then, some necessary files should also be provided to the recognizer. Afterwards, the recognizer will be able to produce a result and save it to a text file. Then, the MPai will check that file and transform the result to meaningful presentations. The idea to the linking process was, using C# code to produce the files and the scripts that are needed for the recognizer, then run the recognizer, then analyze and present the results produced by the recognizer.

The technical details of this process are included in the Appendix of this report. Next, I will talk about the user manual of this newly-built feature.

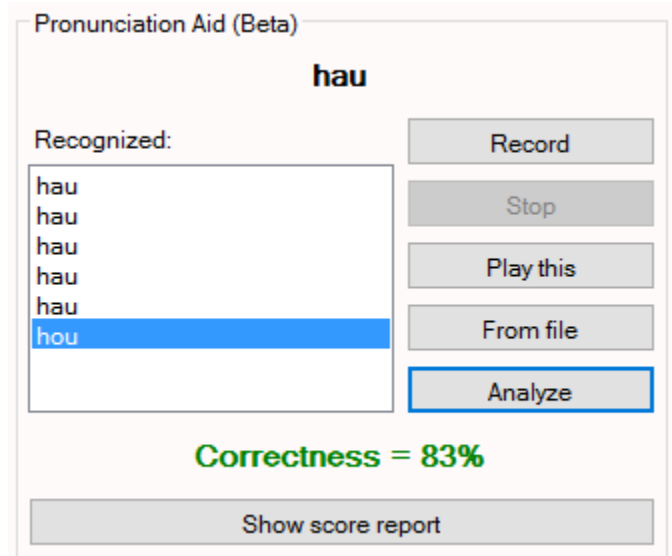


Figure 3.2.1 – the pronunciation aid with Lu's recognizer

See Figure 3.2.1 for the user interface for the speech recognizer [2]. Now the new MPai is able to pass data to the command-line speech recognizer and interpret the data produced by it, and then present it on the user interface therefore the non-technical audience will be able to use the speech recognizer.

When the user has selected a Maori word, it will be displayed on top of the recognizer module. See Figure 3.2.1 as an example, the user has selected "hau". After the user has selected a word, he could start recording by clicking on the record button, and the user is able to speak the word and his recording will be saved to the data folder of the MAPi. Later, when he has finished speaking, he could save his recording by clicking on the stop button. Then the MPai will run the word recognizer in the background and display the result immediately in the recognized list, and the user is then able to see the feedback. After the users have made enough attempts, he is able to click on the analyze button and the MPai will return the correctness of his pronunciations based on how many words the user got right and how many the user got wrong.

But unfortunately, the word recognizer is not yet ready to produce an accurate result, it seems like that more research should be carried out into Lu's

work as well as the HTK to improve the accuracy. Therefore, the pronunciation aid is now marked as “beta” software and cannot be used in formal situations.

3.3 The design of a report page for the recognizer

Despite the fact that the recognizer is not yet ready to be released. We can still make a detailed report page for the pronunciation aid. The report page will detail on each of the user’s recording and provide the user with meaningful feedbacks.

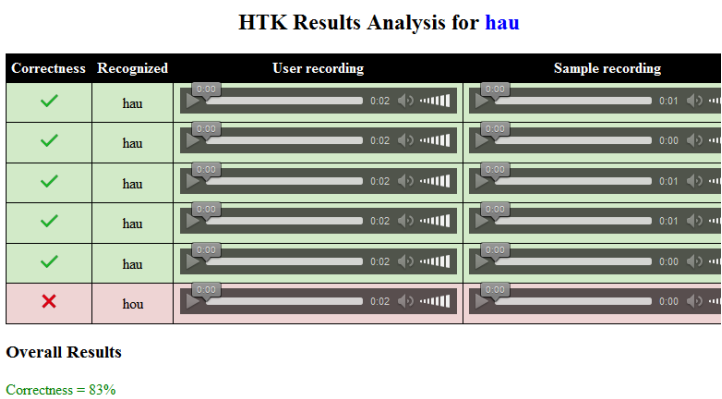


Figure 3.3.1 – the new report page based on HTML

The new report page is created by HTML using the HTML generator class written in C#. When all the result has been collected, MPai will use the HTML generator to create the report for the user. There is some reasons why using HTML for the report:

- 1) HTML is light weight, we will no longer need to create new forms for the report, we can save a lot of code
- 2) HTML is user-friendly, and it will generate the layouts and the controls automatically and make sure the contents are displayed correctly and elegantly

When the HTML report is ready, the MPai will open the default browser in the system and load the report to the user. Then the user will be able to see his/her performance and compare his/her recording with sample recordings.

From Figure 3.3.1, we can tell that the user made 6 attempts for the word “hau”, and he got the first five attempts correct. Therefore, in the report page, in the correctness column, his attempts are marked as the “correct” sign. Then in the second column, there is the recognized results produced by the recognizer. We can see that the user failed to pronounce the word correctly, it was recognized as “hou”. Then, in the third and fourth column, the user will be able to hear his own recordings and listen to sample recordings for that word from the database. That would help with the learning process for the users.

3.4 The design of the user management system for the new MPai

I have also developed a user management system for the MPai, which means, different users can create different accounts in the MPai, so the MPai will separate the data of each user and provide personalized services.

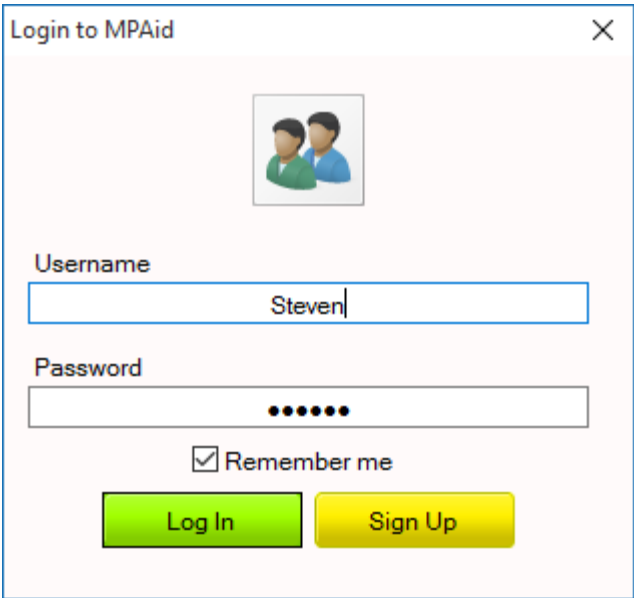


Figure 3.4.1 – user login window

Figure 3.4.1 is the initial window you’ll see when you start MPai, it would request a username and a password to login. If you don’t have them, don’t

worry. You can always create an account by clicking on the “Sign Up” button.

By default, there is also an administrator account, and the username and password is by default “admin”. You cannot change the password for the administrator, the administrator has the access right to view the user list and modify all the users.

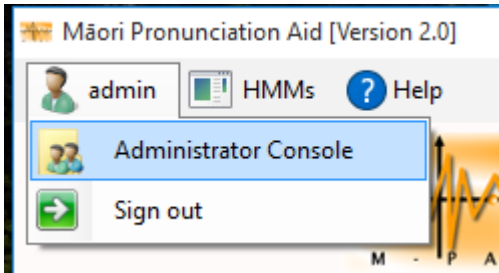


Figure 3.4.2 – the link to the admin console

See above Figure 3.4.2. When the user is logged in as the administrator, the administrator console will be enabled.

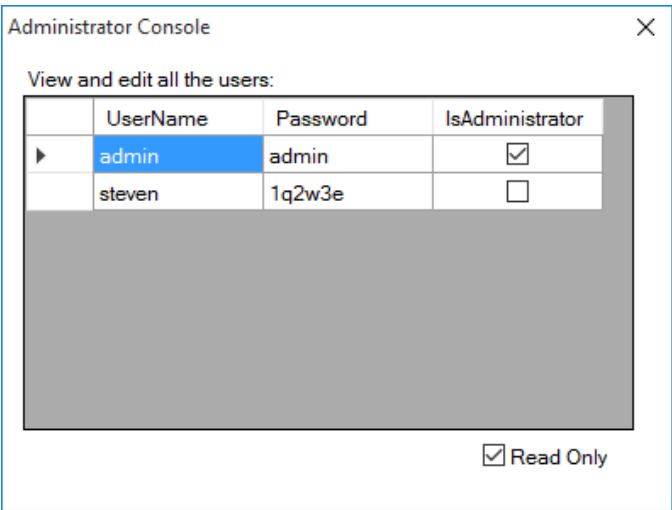


Figure 3.4.3 – the administrator console

When entering the administrator console, see Figure 3.4.3, all the users in the MPAi will be displayed. The administrator can also change the username or password for MPAi users.

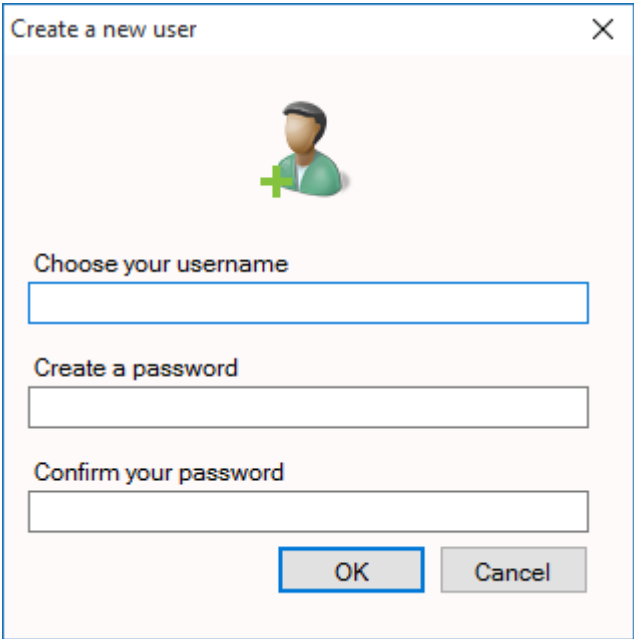


Figure 3.4.4 – user sign up window

See above figure 3.4.4. You can create a new user in the sign up window, then use the account you’ve just created to sign in. Since the user management system has just been added into MPAi and it does not require any other items rather than the username and password. I think in the future, more details about the users should be required. Such as the gender and the age of the user can be required.

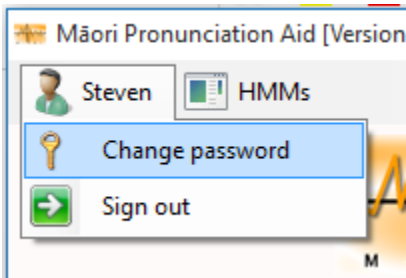


Figure 3.4.5 – user operations

When the user has logged in, he/she would be able to change the password or sign out for different users.

Therefore, in the main window, two operations are added in the menu bar.

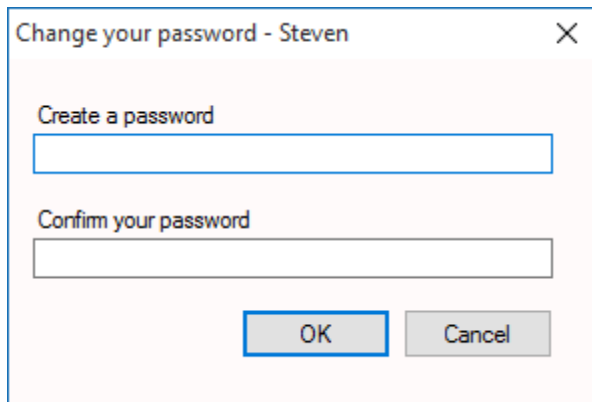


Figure 3.4.6 – change password window

See figure 3.4.6 above. That's the window pops up when the user has selected the "change password" command from the menu bar. The user is able to create a new password for his/her account.

With the help of the user management system, the different users can store their progresses in separate folders. For example, the reports generated for different users will be stored in different folders, and can be retrieved later.

In sum, the first step of the user management feature is implemented and fully functional. There are yet more to go, since the features provided by current MPai do not distinguish between users. In the future, more customized features can be added to MPai with the support of the user management system.

3.5 Miscellanies

In this last section, some small miscellaneous new features will be displayed.

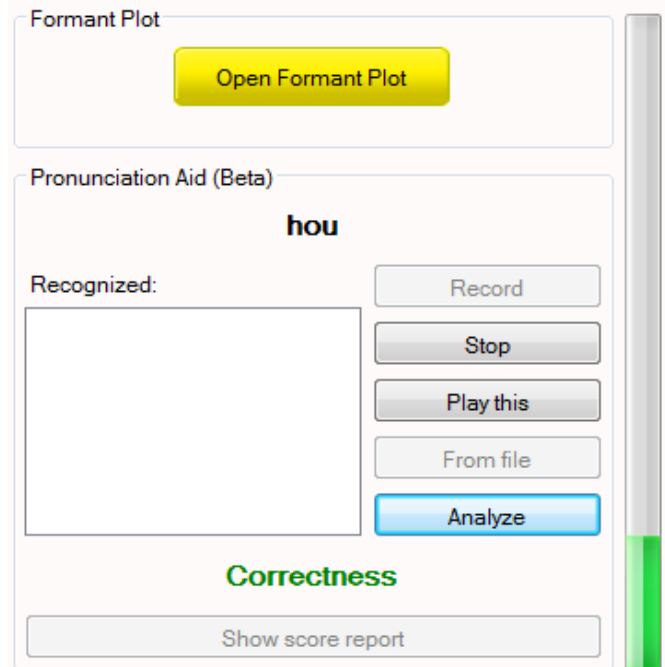


Figure 3.5.1 the volume meter

See Figure 3.5.1. In the right part, there is a vertical progress bar used as a volume meter. When the user is recording their speech. The volume meter will be activated and measure the loudness of the user's voice. This will help the user to adjust his/her volume to adapt to the speech recognizer, so the recognizer will be able to get better accuracy on the results.

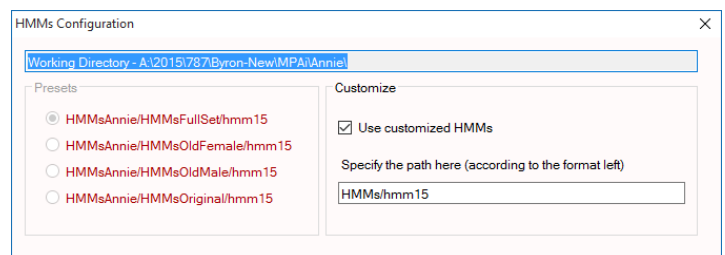


Figure 3.5.2 – the HMMs configuration window

See Figure 3.5.2. The second small design for the new MPai is the HMMs configuration window for the word recognizer. Since the recognizer is using HMMs, and the models are actually replaceable. Therefore, I made this feature for the recognizer to deal with the situations when new models are trained and produced. The MPai will be able to utilize the new models without being modified.

However, in the presets, only the first set of HMMs work. The reasons are not yet known. More future work should be required to investigate deep into this issue.

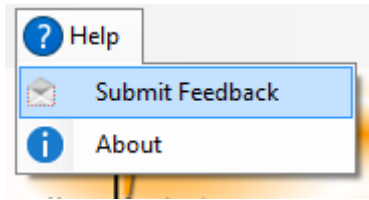


Figure 3.5.3 – feedback

See above figure 3.5.3, there is another design which being added to MPAi – user feedback option. By clicking on that menu item, the system email client will be started and a new email will be created. Then the user will be able to type in any feedback they like and then send the feedback to Dr. Catherine.

4. Evaluations

In this section, the software development method and process will be discussed. Also, the evaluation and feedback methods will be addressed.

The software development method we practiced was similar to Agile. I work feature after feature in sprints of two weeks. In the starting of the sprint, Dr. Watson and I would have a meeting to decide on the next feature that I would be working on. Then, the sprint would start, I would then work on the next feature, and at the end of the sprint, another meeting will be held again for the demo of the new features completed in the last sprint. In addition to this, a meeting of evaluation with Dr. Peter Keegan and Mr. Ary Noviyanto is held for every six weeks, and they would also provide good feedback to the project, and some of them has been implemented and included in the MPAi. Finally, in the end of the project, another meeting with the MAONZE group was held, and a demo on all the features from the previous sprints were given. MPAi received great feedbacks from the MAONZE group.

5. Future work

For the future development of MPAi. I think there are three parts.

First and foremost, the accuracy of Lu's word recognizer should be improved, as Lu has mentioned before [2], a completely different HMM training toolkit could also be used, instead of HTK, if it is deemed that HTK is unable to provide desirable results, or a completely different method of speech recognition such as using artificial neural networks could be tried [2]. When the accuracy has been enhanced to a higher level, I think the pronunciation aid will be released for formal use.

Second, still for the pronunciation aid. Since the students might use the pronunciation aid for multiple times for different words at different timelines. Multiple records could be collected and analyzed by the MPAi. By analyzing these data, the MPAi will be able to know the weakness of each student and provide personalize support for the students.

Third, since the MPAi support only windows platform and it is a desktop application. I think it is possible that a web-based MPAi can be made using ASP.NET – that is to say, a web version of MPAi can be made. If that become real, the students with a Macintosh or Linux will be able to use the MPAi. Since the new MPAi is written in C#, it can be migrated to ASP.NET and transform to a Web application without too much difficulty.

6. Conclusions

For this research project, I continued the development of the MPAi project under the supervision of Dr. Watson, and made several achievements for this project. Now, several new features are added to the new MPAi, including the connection to the word recognizer, the new user

management system, the new report page for the pronunciation aid, as well as the new user interface with minimum number of bugs – no bugs were found up to now.

However, there are yet more work to do for the MPAi project in the future. The most important one I believe is to improve the accuracy of the word recognizer, since the word recognizer cannot produce meaningful results up to now, due to its limited accuracy. Also, when the accuracy issues is fixed, it could be better if the MPAi can analyze the students' performances and provide personalized support for each student – theoretical that is achievable. Obviously, to do that, more time and efforts are required. Finally, I think the one of the most contributing idea is that – a web version of MPAi can be made using ASP.NET, if that can be done, the MPAi will be able to reach to more students and the time and efforts for deployment can be minimized.

Acknowledgements

Special thanks to Dr. Watson for the continuing support and guidance for this research project, thanks to Dr. Peter J Keegan for testing and guidance on the word recognizer, also thanks to Mr. Ary Noviyanto for making great advices for the MPAi project.

7. References

- [1] Hui, Byron (2011). "Visual Feedback for Language Production", unpublished report of student project, the University of Auckland.
- [2] Lu, Annie (2012). "Maori Pronunciation Aid Speech Recognizer using HTK Practical

Work Report", unpublished report of student project, The University of Auckland.

- [3] KING, J., HARLOW, R., WATSON, C. KEEGAN, P. AND MACLAGAN M (2009). "Changing Pronunciation of the Maori Language: Implications for Revitalization" in Reyhner J, and Lockhead, L (Eds) Indigenous Language Revitalization Encouragement, Guidance & Lessons Learned Northern Arizona University, p 75-86.
- [4] Gutla, J.J. (2006). "A Māori pronunciation aid". In Proceedings of the 2006 Year Four Research Projects (Vol. II, pp. 504-512). Department of Electrical and Computer Engineering, University of Auckland, New Zealand.
- [5] Rivers T Daniel. "A Maori Pronunciation Aid". In Proceedings of the 2006 Year Four Research Projects (Vol. II, pp. 513-521). Department of Electrical and Computer Engineering, University of Auckland, New Zealand.
- [6] KEEGAN, P. J., WATSON, C. I., KING, J., MACLAGAN, M., & HARLOW, R. (2012). The Role of Technology in Measuring Changes in the Pronunciation of Māori over Generations. In T. Ka'ai, M. O Laoire, N. Ostler, R. Ka'ai-Mahuta, D. Mahuta & T. Smith (Eds.), Language Endangerment in the 21st Century: Globalisation, Technology and New Media, Proceedings of Conference FEL XVI (pp. 65-71). AUT University, Auckland, New Zealand: Te Ipukarea - The National Māori Language Institute, AUT University/Foundation for Endangered Languages.

Appendix – Technical details

This part is for technical audiences, by following this technical guide will quickly get you through the preparation work for the development of MPAi.

A. Development Tools

The new MPAi we've talked about is written in C#, and it is strongly recommended that you edit the project using the latest Visual Studio. Up to now, the latest version is Visual Studio 2015. You can download Visual Studio Community Edition here, it is free to use.

<https://www.visualstudio.com/en-us/downloads/download-visual-studio-vs.aspx>

Or you can download the Visual Studio Enterprise Edition from Microsoft DreamSpark for University of Auckland.

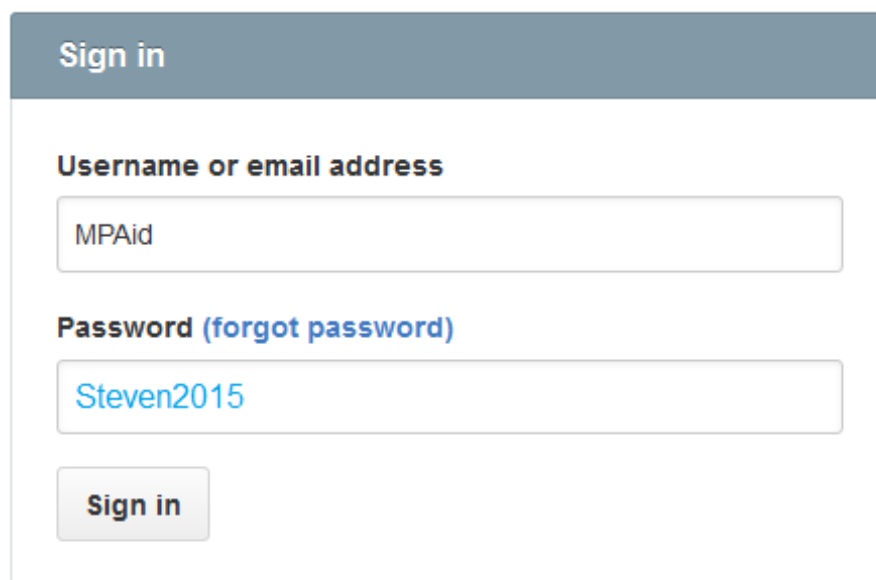
Since it is running on Windows. The new MPAi is actually a WinForm application written by C#, WPF was not chosen because the architecture (MVVM) of WPF is very complicated to grasp. Compared with WPF, WinForm is very flexible and it is easy to design. However, the disadvantage of WinForm is that, it cannot provide beautiful 3D animations, and the controls cannot be nested like WPF does. It is similar to the comparison of WebForm and MVC. But for this project, WinForm is enough.

After you have Visual Studio ready, you can go to the next part.

B. The Version Control System

All of my part is on GitHub, for Byron and Annie's part of this project, please consult Dr. Watson.

You can use this GitHub account to visit the MPAi project on GitHub.



Sign in

Username or email address

MPAid

Password (forgot password)

Steven2015

Sign in

Then, the next thing you'll need to do is, using Visual Studio to import the code from GitHub. If you are not familiar with this process, you can start a google search by "Import Github project to Visual Studio". Or you can check the link here from Stack overflow:

<http://stackoverflow.com/questions/19892232/how-to-connect-visual-studio-2012-with-git-github/19893242#19893242>

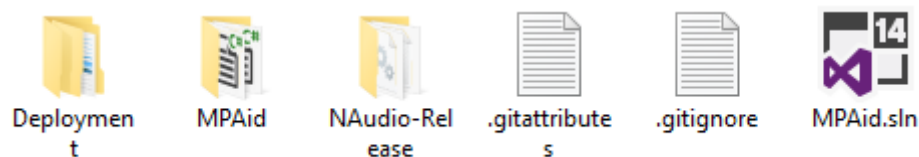
The GitHub repository location you'll need in this process:

<https://github.com/mpaid/main>

And you can use your own GitHub account or you can use the account that mentioned before. If you are using your own account, remember to add your account into the contributors of the MPAi project.

C. The hierarchy of the project

When you have finished cloning the repository from GitHub, you'll see several directories.



Now you can open the project by opening the **MPAid.sln**. Please do remember to build the project first to check that the project is complete, and since I used customized controls, by building the project, Visual Studio will be able to find all the controls needed. Otherwise, you might get an error when you try to open the forms.

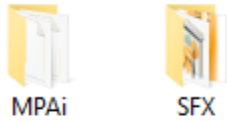
The **NAudio-Release** folder is not the focus of our topic, it just stores the one of libraries used in the MPAi, since we have already included the NAudio.dll in the Visual Studio project, this folder can be removed safely. But for legitimate purposes , I put that on GitHub to indicate that this is a third party library.

The **MPAid** folder is exactly where the C# project is stored, you'll not need to open it. Just opening the **MPAid.sln** file will be enough.

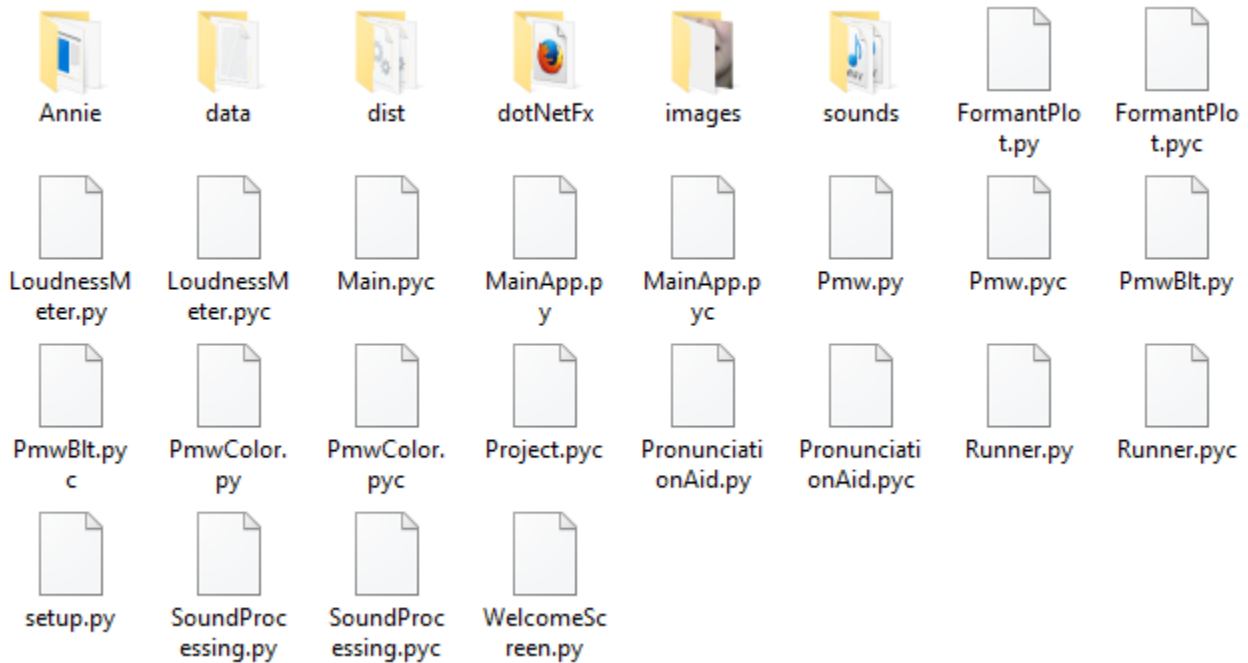
Then, in the next section, we'll discuss about the Deployment folder.

D. Deployment

The **Deployment** directory stores all the files that are need for the deployment process. In simple terms, the deployment folder stores all the files that are needed when you are trying to make an installation package for MPAi. When you go inside it, you'll see two folders. The **MPAi** and **SFX**.



The **MPAi** folder is exactly the files you'll need to install to the user's machine. When you go inside it, you'll see a lot of Python files and several folders – don't worry about the python codes, I will explain them later.



Note that there is a folder name **dist**, which is exactly the folder where the executable files are stored. When you have modified the C# project and generated a new exe file, please remember to put that exe file to this dist folder, so that it can be installed to the user's machine.

Now let's go to this **SFX** folder. Since we are using WinRAR to create installation packages for MPAi. This folder stores the icon for the installation package (see MPAid-Setup.ico below), and the scripts needed to create WinRAR self-extract packages.

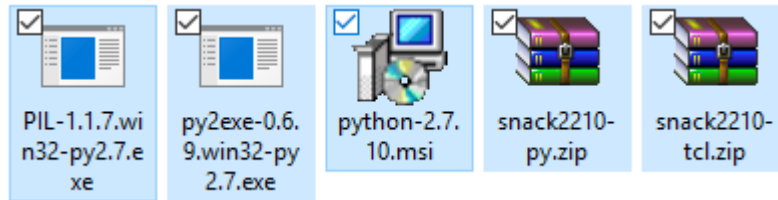


We are talking about the software deployment process, and I would recommend you using WinRAR to create the installer package for MPAi, because of its stableness and efficiency.

If you are not familiar with this concept, you can google it – “using winrar to create installation package”.

E. Byron's part (Python)

To edit and compile Byron's Python code, below is a list of the tools that you must have.



The first package is [Python Imaging Library \(PIL\)](#).

The second package is [py2exe](#), it is used to convert Python source code into executable Windows programs, able to run without requiring a Python installation.

The third package is [Python 2.7.10](#), please do not use Python 3, it seems like the Python 3 cannot be used for Byron's python code.

The fourth and fifth package is the [Snack Sound Toolkit](#).

For the rest of Byron's technical part, please refer to Byron's report – Visual Feedback for Language Production. You can consult Dr. Watson for it.

F. The word recognizer engine

As has been said before, the new MPai has already integrated Annie's word recognizer in. But there are yet more things to do for the recognizer to improve its accuracy.

Let's start from the most important classes in the project: the **PaEngine class** and **PaConfig class**, then we will go to the less important classes.

These two classes are the key to associate Annie's word recognizer with the MPai.

As the name indicates, "Pa" stands for pronunciation aid, PaConfig is the classes that used to control the physical path of the files used by the word recognizer.

Then, let's go to the most important class – the PaEngine. Before you continue reading, I hope you have already scanned through Annie's report. If you don't have Annie's report, you can go to Dr. Watson for it.

Annie's word recognizer is based on HTK toolkit. HTK toolkit is actually a set of command-line applications developed by Cambridge University (Wow!), you can visit its [official page](#) to learn more. When running, the HTK would take the HMMs and the recording as its input and produce a result.

The working directory should be **deployment\MPai\Annie**.

Here is the batch script used to run Annie's word recognizer taken from Annie's work:

```
HCOPY -T 1 -C user/config0 -S user/mpaidScript1.scp
```

```
HVite -H -C user/config1 HMMs/hmm15/macros -H HMMs/hmm15/hmmdefs -S user/mpaidScript2.scp -l * -T 4 -i  
MPAidOutput/mpaid0A.mlf -w user/wordNetwork -p 0.0 -s 5.0 user/dictionary user/tiedList >  
MPAidOutput/mpaidHViteOut
```

```
HVite -H -C user/config1 HMMs/hmm15/macros -H HMMs/hmm15/hmmdefs -S user/mpaidScript2.scp -l * -a -f -i  
MPAidOutput/mpaid0B.mlf -w user/wordNetwork -p 0.0 -s 5.0 user/dictionary user/tiedList
```

```
HResults -t -l user/wordTranscript.mlf user/tiedList MPAidOutput/mpaid0A.mlf > MPAidOutput/FinalResult.txt
```

Please note that this is only **one of the many ways** to run her word recognizer, since the accuracy of the MPAi is not satisfying now. You might want to explore other ways to run her recognizer or modify the script to make her recognizer work better and produce more accurate result. I would recommend that you can investigate deeply into Annie's report and her work to reveal better ways of using her recognizer.

Here I will focus on how the word recognizer is integrated into the MPAi.

The idea is simple, when you have learned how to run Annie's recognizer using batch scripts, you can also do it in C#, and even do it more quickly, because when you write those scripts manually, you will be extremely careful. But if you tell C# to write it, it will be very easy – your computer will take care of it. That's exactly the idea of the **PaEngine** class and **PaConfig** class.

Where by using the PaConfig class, you can specify all the paths used in the scripts, and the PaEngine class will then take a PaConfig-typed variable as an input and generate the batch script automatically as above.

Here I will explain more on how the script work.

There are several files that the PaEngine needs to prepare before running the recognizer. They are the mpaidScript1.scp and the mpaidScript2.scp.

In the first step, when you have a list of .wav files that holds the recording that are waiting to be recognized, you first need to transform them to .mfc files. That's where the mpaidScript1.scp comes in. This file stores the path to the .wav files and the path to the target .mfc files which will be generated later.

By running the first line in the batch script "HCOPY", the .wav files will be transformed into the .mfc files.

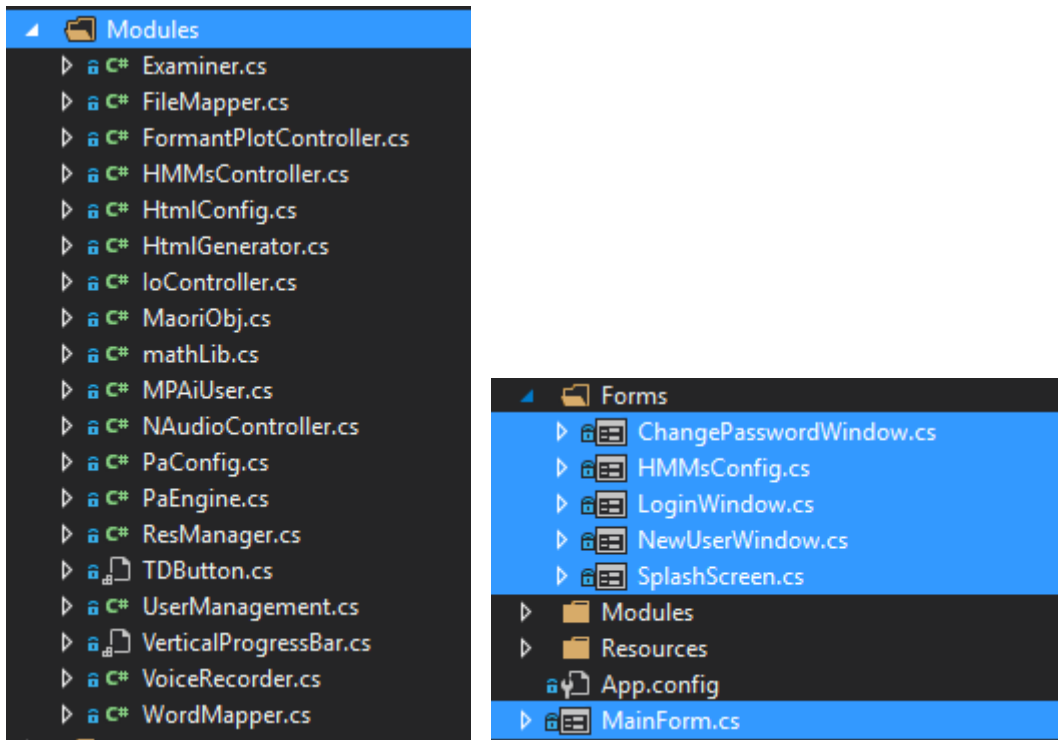
Then, the mpaidScript2.scp only stores the path to the .mfc files which generated in the first step, and the mpaidScript2.scp will be used in the later commands as an input of the recordings to be recognized.

In the later steps, the **HMMs/hmm15/macros** and **HMMs/hmm15/hmmdefs** are exactly the HMMs trained by Annie, you can replace them with new your HMMs if you have them.

In the last step, "HResults ..." is when the HTK analyze the result and produce an output. Note the sign ">" in the script, that means the script will output the result to the file. In the sample script, the script finally output the result to MPAidOutput/FinalResult.txt, and this file will be analyzed by the C# code, to get the result from the recognizer.

To conclude, the PaEngine automated this process, first, it took the input recording from the PaConfig class and generate the mpaScript1.scp and the mpaScript2.scp for the recognizer. Then it generate the script used to run the recognizer. Then it runs the script and analyze the output made by the recognizer. Then it sends the result to the user interface therefore the user will be able to see their recognized result.

G. Classes



Here I will provide a technical documentation of the project.

Let's start with the forms, see the screen shot above on the right.

| Form name | Functions |
|----------------------|---|
| MainForm | The MainForm is the main window of this application, you can see all the functions provided in the application from MainForm. |
| SplashScreen | The SplashScreen is used when the main window is loading data. |
| HMMsConfig | The HMMsConfig is the window used to configure Annie's HMMs. |
| LoginWindow | The LoginWindow is the starting window requesting your username and password. |
| NewUserWindow | The NewUserWindow is the window when you are signing up for this application, you can create a new user in that window. |
| ChangePasswordWindow | The ChangePasswordWindow, as the name indicates, is used to change password for the user. |

For the modules, there are several classes and controllers, see the screen shot above on the left.

| Class name | Functions |
|------------------------------|---|
| FormantPlotController | The controller to manage the startup and shutdown of the formant plot – the key to the hybrid architecture |
| HMMsController | The controller to manage the path to the HMMs |
| NAudioController | The controller for NAudio.dll, get the input volume from the recording device |
| UserManagement | Provide backend support for the user management system, the class maintain a list of MPaiUser objects |
| MPaiUser | The basic type of MPaiUser, used for the user management system as an user element, consists of a username and a password |
| mathLib | A static class stored with some math functions |
| Examiner | The class used to compare recognized word and laboratory word |
| IoController | The controller to manage some certain working directories, go inside for details |
| HtmlConfig | The class used to control the physical path and file names of the HTML report |
| HtmlGenerator | The class used to create a detailed HTML-typed report for pronunciation aid, used an HTMLTextWriter class to produce the HTML script |
| MaoriObj | Each vowel and word is defined as a MaoriObj, this class stores the label for the vowel/word and the path to the pronunciation file of the vowel/word |
| ResManager | The class used to visit and initialize the resources – such as the pronunciation data and images associated with the word |
| VoiceRecorder | As the name indicates, this class encapsulates a system API used for recording sounds and make it a voice recorder |
| FileMapper | Map the sound file name to the word, you can use this class to get the sound file for a word |
| WordMapper | Map the word with an ID, used to link the word with its correct id in the sound database |
| TDButton | This is a customized button used to show a 3D effect |
| VerticalProgressBar | The vertical progress bar used to show the volume of the input |