Phoneme Recognition User Manual

T. Buckingham

May 23, 2022

Contents

1 Introduction

This user manual is for the Python GUI application, if you wish to use the command line program with arguments please see the section at the bottom of this manual to find descriptions of the arguments along with examples of run scripts.

2 Prerequisites

The GUI application has been tested using Python version 3.8 but it should work with any version of Python3 however, this has not been tested so it is recommended to update to the latest version of Python3. Please check your operating systems methods for installing or updating Python.

Using the Python GUI requires some Python packages to be installed as they may not be included in your installation of Python. It is recommended to install and use Pip for installing these packages. All packages can be installed with Pip using

pip install {package name}

The main program has been compiled using GCC with C11 and pthreads and as such has only been tested and run natively using a Linux operating system. As pthreads is not available on Windows another version has been provided for Windows, this version omitts using threads for faster MFCC generation however, in all other functionalities they are indentical.

2.1 Python Packages To Install

numpy A maths and matrices library used to handle the confusion matrix.

seaborn A stastistics and visualisation library used to generate the heatmap.

matplotlib A graphing and visualisation library used to display the heatmap.

kivy A graphical user interface library used to create the main application.

2.2 Installing GCC

Most Linux distrobutions will already have GCC installed to test this open a terminal and enter

gcc –version

the version used during development is 7.5.0 and so it is recommended to upgrade your current version if it is behind using

sudo apt-get upgrade build-essential

If GCC is not installed on your Linux operating system you can do this by entering

sudo apt-get install build-essential

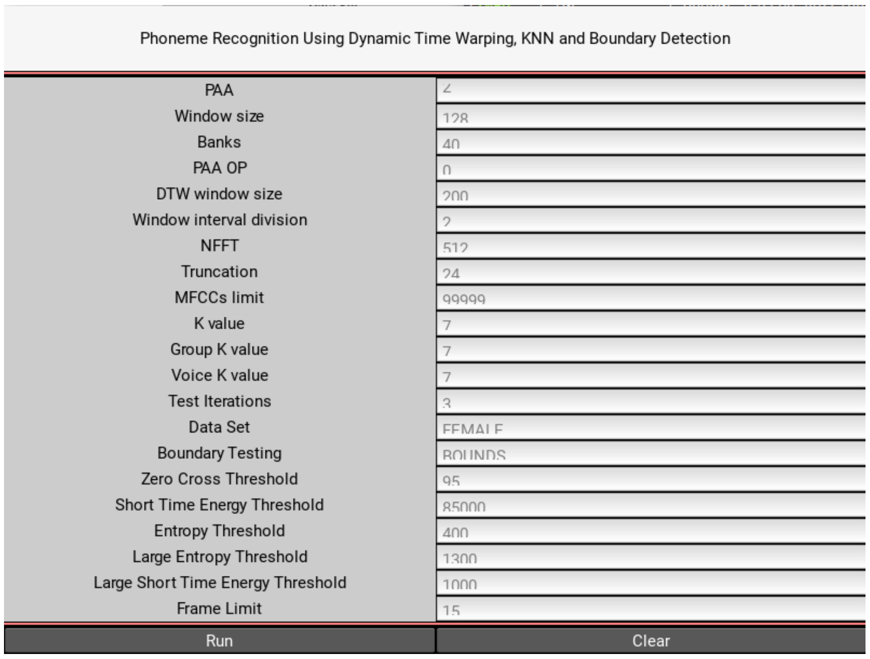
If using a Windows operating system another compiler may be used however, the results cannot be guranteed and so in order to install GCC please go to either Cygwin (`https://sourceware.org/cygwin/`) or MinGW (`https://osdn.net/projects/mingw/`) to download the associated installers and follow the step-by-step guides.

3 Using the GUI App

To run the application open a terminal in the directory containing the file 'main.py' and run the command

```
python3 main.py
```

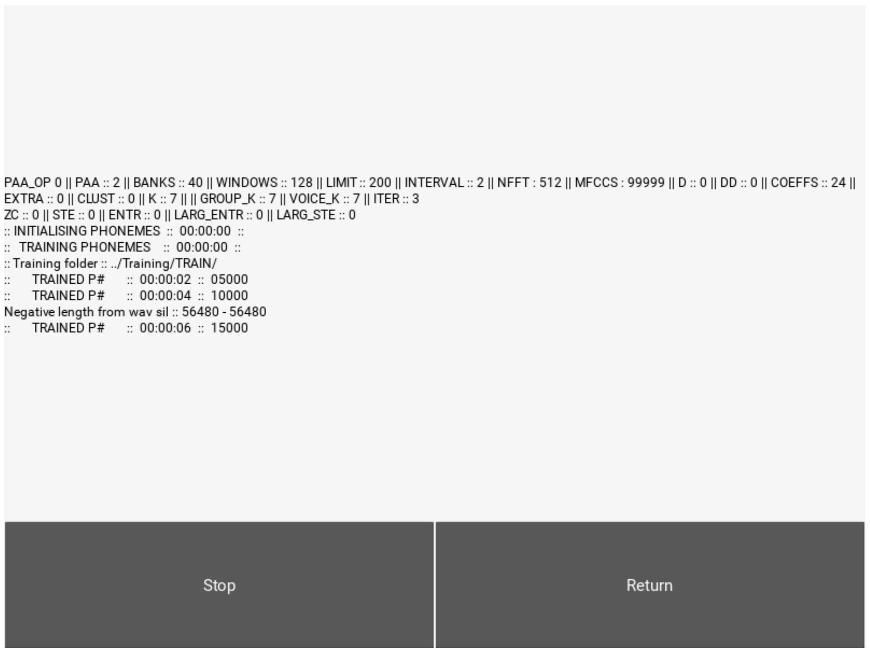
this will open an application as shown below.



Phoneme Recognition Using Dynamic Time Warping, KNN and Boundary Detection

PAA	2
Window size	128
Banks	40
PAA OP	0
DTW window size	200
Window interval division	2
NFFT	512
Truncation	24
MFCCs limit	99999
K value	7
Group K value	7
Voice K value	7
Test Iterations	3
Data Set	FEMALE
Boundary Testing	BOUNDS
Zero Cross Threshold	95
Short Time Energy Threshold	85000
Entropy Threshold	400
Large Entropy Threshold	1300
Large Short Time Energy Threshold	1000
Frame Limit	15

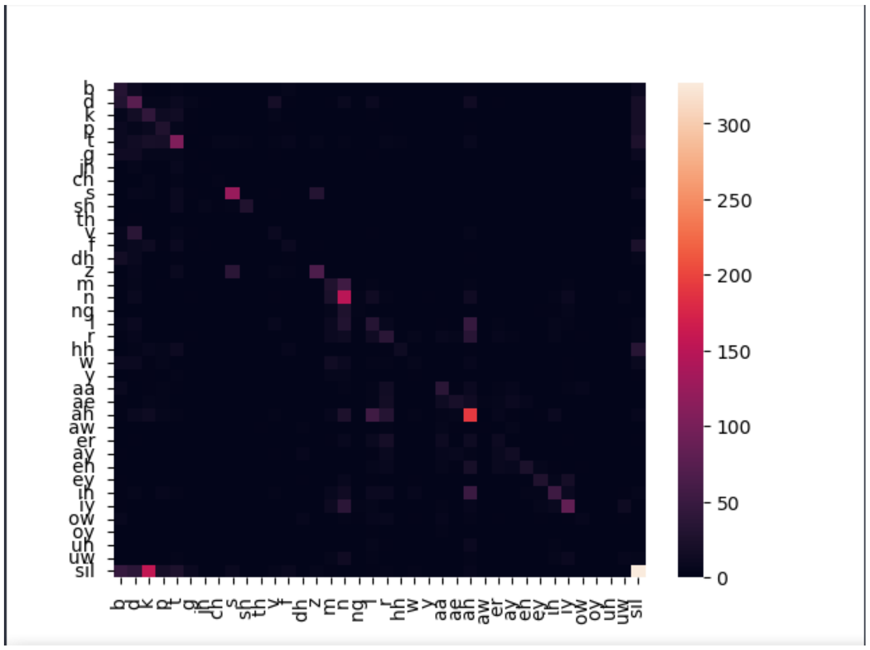
On the left side of the app is the parameter and on the right is a text input allowing you to alter the parameter's value. Once the parameters have been changed to the user's preferences click the 'Run' button in the bottom left, doing so will display a new window as shown below.



```
PAA_OP 0 || PAA :: 2 || BANKS :: 40 || WINDOWS :: 128 || LIMIT :: 200 || INTERVAL :: 2 || NFFT : 512 || MFCCS : 99999 || D :: 0 || DD :: 0 || COEFFS :: 24 ||
EXTRA :: 0 || CLUST :: 0 || K :: 7 || || GROUP_K :: 7 || VOICE_K :: 7 || ITER :: 3
ZC :: 0 || STE :: 0 || ENTR :: 0 || LARG_ENTR :: 0 || LARG_STE :: 0
:: INITIALISING PHONEMES  ::  00:00:00  ::
::   TRAINING PHONEMES    ::  00:00:00  ::
:: Training folder :: ../Training/TRAIN/
::      TRAINED P#      ::  00:00:02  ::  05000
::      TRAINED P#      ::  00:00:04  ::  10000
Negative length from wav sil :: 56480 - 56480
::      TRAINED P#      ::  00:00:06  ::  15000
```

This window will show the current progress and output of the program along with a print out of the parameters set by the user. To stop the running of the program press the ’Stop’ button and the process will be stopped then pressing the ’Return’ button will return you to the previous window allowing you to change the values again.

Once the program has finished a matrix will be outputted by the C program then read and display by the Python program. The matrix will be displayed as a heatmap as shown below.



4 Compiling and Running the C Program

Located in the *Dynamic_Time_Warping* folder is a *build.sh* script which can be run from a terminal by entering

./build.sh

this will compile the program from source using the following command

"*gcc -std=c11 ../Training/train.c dtw.c ../Misc/realloc.c ../Testing/test.c ../Seperation/cross_rate.c ../Seperation/ste.c ../Feature_Extraction/*.c ../Seperation/bounds.c ../Clustering/cluster.c ../Clustering/knn.c -D_XOPEN_SOURCE=600 -pthread -onan -o dtw.exe -lm;*"

which can then be run using the parameters provided in the report by running the *run.sh* for one-to-one comparisons or the *run_bounds.sh* script for testing with boundary detection, these are located in the same folder.

If the user wishes to input different parameters and does not wish to use the provided GUI application to do so then a list of CMD arguments is provided below. Note: Parameters in all caps denote an on/off option whereas those that are not capitalised are used by stating the parameter name followed by a space then the value: *{parameter} {value}*

5 Using Audio Files

The system has been based around the NTCD-TIMIT data-base and contains the following folders for audio and label files to be added

TRAIN/TEST These folders contain all volunteer files, no lip-speakers were added. It is recommended to add your whole data-base to this folder.

FEMALE/MALE Two seperate folders, FEMALE and MALE, were added to compare the effectiveness of different models for each gender and to test if there was any noticeable differences between genders.

SPKR1 An individual speaker from the lip speaker data. This was used to test and determine the effectiveness of the model when multiple people and accents were included in the model.

SPKR1_(NOSIL/CAFE/CAR/WHTE/STRT) A number of different noise types for SPKR1 to test the noise robustness of the model.

It is not required for the user to follow this format, or even use the NTCD-TIMIT data-set however, these are the folder names that are recognised and as the system cannot dynamically find new folder names the user should keep a note of what each folder is being used for.

To re-create place the data-set as package by the NTCD-TIMIT version, found here (called 'Clean.tar.gz), into the 'NTCD-TIMIT' folder such that the directory will be 'NTCD-TIMIT/Clean/clean/' with the lipseaker and volunteer folders, and .mlf files, are found within this 'clean' folder. Located in the 'NTCD-TIMIT' folder is a *build.sh* file, running this will generate two .exe files from the two associated .c, running one at a time they will generated the labels files and move the .wav and .PHN files to the appropriate folders. These generated files will be found in the 'labels' folder in which the user will find 'TEST' and 'TRAIN' folders with subsequent 'MALE' and 'FEMALE' folders - to generate the 'TEST' and 'TRAIN' that contain all the information simply combine the male and female folders.

The only thing that is required is that all audio files be in the '.wav' format and all label files have the file extension '.PHN', they must have the same filename. As the NTCD-TIMIT was sampled at 16Khz this is the rate that the system will look for and any other given rate will be flagged as an error; to avoid this it is recommended to down/up sample your audio files first. All label files must be in the format of number of samples and must be the same frequency as the audio file.

If the user wishes to use a higher frequency sampling rate it may be possible to change the given sample rate in the audio file without changing the data sector of the file and providing an appropriate label file. As the program simply gets the data sector of the audio file and indexes the array using the label file values. Note: this has not been tested so do still recommend downsampling to be sure.

6 Output

A number of different output files are produced depending on the options provided by the user. During regular one-to-one testing each test result will be outputted to 'correct', 'fail' and 'group' folders in the 'Dynamic_Time_Warping' folder. The filename will be the name of the phoneme being tested, at the top of the file the correct phoneme's score will be displayed along with the guessed phoneme's score, following this will be a list of all tested phonemes with their scores.

During boundary testing the result files will be in the 'res' folder of the used folder i.e. if testing with 'FEMALE' the '.res' files will be in */FEMALE/res*. Each '.res' file will be the same name as the '.PHN' file being tested and will be in the same format of {start end phoneme}.

Two different matrix files are produced, one in 'Dynamic_Time_Warping' which is for use with 'matrix.py' - which can be displayed as a heatmap using *python matrix.py* - but another for detailed file is appended to each complete run of the program and is located in */Testing/TEST*. This version of the matrix file will contain the phoneme, group and voice confusion matrices along with information about the parameters used for the test and the exact number of correct, incorrect and group correct guesses. As it is appended to each time we recommend using this to keep track of any loop tests ran with a bash script.

Upon completion a folder called 'device' will be generated, this is found in the root folder of the project. This is the exported MFCCs and necessary configuration files for the microcontroller, in order to load this onto the device simply copy this folder to the root of an SD card and plug it into the device's SD card port - if using a Teensy as shown in this project then this SD card will be a micro SD.

7 Command Line Arguments

Note: Selecting no KNN option will result in used frame based Dynamic Time Warping on its own and not selecting a data will use the data in the 'TEST' and 'TRAIN' folders.

paa The divisor used for Piece-wise aggregation psuch that the remaining signal is (signal length / *paa*). In order to prevent data loss it is advised to use PAA values that are a power of 2 due to the use of an FFT, other divisors may result in data at the end of a signal being lost.

window The size of the Hanning window, this value must be a power of 2 again, due to the use of an FFT.

banks The number of Mel filter banks to be produced, these can be truncated. This value must be lower than (window / 2).

trunc The number of coefficients the user wishes to keep from the filter banks; this value must be lower than *banks*.

dtw_window The Dynamic Time Warping window limit. This will be used as a percentage of the largest length signal. The user inputted value will be divided by 1000 due to Bash's inability to handle decimals and as such a value of 200 will result in a value of 0.2, or 20%.

interval_div The overlap value of the Hanning windows, this value must be a power of 2 in order to keep the window width a power 2 for the FFT.

nfft The NFFT value used to generating the Mel filter bank.

CLUST Will select the option of using Jenks clustering for testing; this will run a Python script in the background. Selecting this option with *KNN* will perform k-means when testing.

clust_num The number of centroids used for the Jenks clustering method; CLUST must be set in order to use this option.

KNN Will select the option of using base KNN for classification; note this option must be selected to use group or voiced KNN.

knn The k value for use in the base KNN.

GROUP Will select the option of adding the group hierarchy to the classification.

group_k The k value for use in the group hierarchy KNN.

VOICED Will select the option of adding the voiced hierarchy to the classification.

voice_k The k value for use in the voiced hierarchy KNN.

test_iter The number of test iterations to be performed. To prevent overfitting of the data the *SPLIT_DATA* option can be selected which will divide the number of tests per test run by the number provided to *test_iter*

SPLIT_DATA Splits the testing data into equally sized groups.

mfccs The limit number of MFCCs per phoneme, if more examples of a phoneme are found then they will be interpolated and averaged to the nearest in length.

frame_limit The maximum number of frames per MFCC, this is counted from the start of the sequence.

zc_incr The zero cross threshold used in the boundary functions; the value is the absolute difference of the current and previous window's zero cross values.

entr_incr The entropy increment threshold used in the boundary functions; the value is the change in value from the previous to the current window's values.

ste_incr The short time energy increment threshold used in the boundary functions; the value is the change in value from the previous to the current window's values

larg_entr_incr A larger value than *entr_incr*

larg_ste_incr A larger value than *ste_incr*

DELTA Produces and tests delta coefficients of the MFCC

DELTA_DELTA Porduces and tests delta-delta coefficients of the MFCC

NORM Normalised the MFCC, DELTA and DELTA_DELTA coefficients.

MALE Selects the folders named 'MALE' in the 'Testing' and 'Training' folders to train and test.

FEMALE Selects the folders named 'FEMALE' in the 'Testing' and 'Training' folders to train and test.

SPKR1 Selects the folders named 'SPKR1' in the 'Testing' and 'Training' folders to train and test.

SPKR1_NOSIL Selects the folders named 'SPKR1_NOSIL' in the 'Testing' and 'Training' folders to train and test.

BOUNDS Will run all testing with boundary detection and will produce a WER value, among other things, rather than an accuracy percentage.

MEAN_SIZE Produces averaged MFCCs which are the same length.

PCA Exports the MFCCs into a folder which can be used by the *pca.py* function; this has not be incorporated into training or testing and has only used for producing graphs.

ZC This will select the option of producing and using frame-by-frame zero cross values for the voice hierarchy classification; this cannot be used with *STE*.

STE This will select the option of producing and using frame-by-frame short time energy values for the voice hierarchy classification; this cannot be used with *ZC*.

STRT Selects the street noise version of the SPKR1 test data to train and test.

CAFE Selects the cafe noise version of the SPKR1 test data to train and test.

WHTE Selects the white noise version of the SPKR1 test data to train and test.

CAR Selects the car noise version of the SPKR1 test data to train and test.

THREAD Uses threads for creating the MFCCs. Warning: one thread per phoneme will be created.

EXPORT Will export the phonemes for use with the device.