National College of Ireland

Configuration Manual

MSc Research Project
Data Analytics

Peter Nolan
Student ID: x22154116

School of Computing
National College of Ireland

Supervisor: Dr. Jorge Basilio

National College of Ireland
Project Submission Sheet
School of Computing

National College of Ireland

Student Name:	Peter Nolan
Student ID:	x22154116
Programme:	Data Analytics
Year:	2024
Module:	MSc Research Project
Supervisor:	Dr. Jorge Basilio
Submission Due Date:	12/08/2024
Project Title:	Configuration Manual
Word Count:	1883
Page Count:	5

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	
Date:	16th September 2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Peter Nolan
x22154116

1 Introduction

This Configuration Manual describes the code and data used in the research project. As saved and submitted as part of the project, these should allow the reader to see the results of the models as reported. Beyond that, they can also use those to run the models themselves and replicate the results.

2 Technical Architecture

Technical setup for the data analysis project involved choices over the tools to use. R or Python were obvious choices to use, given the huge choice of libraries for data analysis in both. However, a quick search at the start of the literature on speech processing showed that R was comparatively rarely used, none of the papers in the Literature Review, for example, and that Python dominated as the tool used published research in the field.

Furthermore, given the requirements for handling and processing large data volumes, the use of scripts and of specialised libraries such as **scikit-learn**, **NumPy** and **Pandas** was considered more suitable than **Excel** or a seperate database.

As yet, there seems no decisive majority in the literature using one the two main deep learning libraries over another, **Keras** with **TensorFlow** or **PyTorch**. Given past familiarity with **TensorFlow**, it was preferred for implementing the deep-learning functionality. The **librosa** library is very widely-used in speech processing research, much more so than the **TorchAudio** library within the **PyTorch** deep-learning framework or other libraries. With **TensorFlow** already preferred, the choice in favour of **librosa** followed as the logical choice.

3 Hardware

Coding and analysis was carried out mainly on two machines.

One was a Dell Inspiron with a Intel Core i5-12400 2.50 GHz and 8 GB of RAM running Windows 11 Home which had been bought at the start of November 2022 for use during the Postgraduate Diploma in Data Analytics course.

The other was a custom-built PC with an Intel Core i9-14900KF and 32 GB of RAM and a NVIDIA GeForce 4090 GPU, running both Windows 11 Pro on bootup and Linux in WSL. Without any economical cloud resources, running data analysis on this machine was expected to be the cheapest, more convenient and secure way of handling the large candidate datasets and running processor-intensive applications. Delays in sourcing the

machine until the start of July 2024 were an obstacle to the scope of work that could be done.

4 Architecture-neutral Implementation

The Python code was written and run and output saved in Jupyter notebooks, showing the code, data and output to the end users.

The code was implemented mainly in Windows 10 and 11, using the **Anaconda** distribution of Python to manage virtual environments and libraries. Code for several models was originally written in Jupyter Notebooks using the WSL Linux environment. All models should also run under Linux environments also, but care over some issues is needed, in particular the locations of files in the directory and sub-directory structure.

The models using **Tenor Flow** were run bptj with CPU alone as well as being run faster with GPU. However, with the CUDA drivers for GPUs being released only for Linux over the past several years, using GPUs will need WSL in a Windows environment[1].

5 Data and Files

Files consist of the following categories:

- Jupyter Notebooks holding the code and output.
- Recording files containing the full voice samples, in directories named by counties, so *DUB* for Dublin, *ANT* for Antrim & Belfast. These are the input files from the SAIE book.
- Sample files, 1 second samples of the recordings, in the *sample_output_directory*, produced by the preprocessing ('PRE') notebooks.
- Features files, containing statistics for each sample, including the file name, sound wave and features including the MFCC, ages, genders and others, also produced by the 'PRE' preprocessing code.

6 Code and File Depositories

The project files are available also on GitHub at the public repository at `https://github.com/dpnolan/voxpop`, with this file as the **readme.txt**.

To avoid the size constraints on Github, links are given to the Google Drive directories where the data files for the input, sample and dataframe files are also stored.

7 Directory structure

All files are expected to be placed within a simple directory structure, with a home directory on top, called *voxpop* that contains the Jupyter notebooks and with input and output data in subdirectories beneath it.

[1] `https://www.tensorflow.org/install/pip#windows-native`

7.1 voxpop

Scripts run in this, the home directory

7.2 ANT and DUB

Directly underneath the home directory, these hold the input sound files that are input to the preprocessing scripts.

The source for all of these is the 'Sound Atlas of Irish English' (SAIE) dataset published by Hickey (2004). The directories with the coded county names, as in the files published with the SAIE book. They are WAV files, 50–100 seconds of a single speaker reading out scripted sentences, the same words for all speakers.

With the sampling rate of 22,050 observations per second used in all these files, there are typically several million observations of the sound level (y) by time (x) in each recording file.

These files are imported and their data processed by the preprocessing scripts ('PRE') and not used by other files.

7.3 sample_output_files

This directory also lies directly underneath the home directory. It holds output from the preprocessing scripts of two types.

First, are the DataFrame pickle files (PKL) generated by the preprocessing of personal data - age, gender and location data, then the sound data and the features caculated from it, namely MFCC, MFCC delta and MFCC delta 2. Usually over 1GB in size, these can be slow and difficult to move or tansfer.

Also saved to this directory by the preprocessing notebooks are the sample files, WAV sound files coming from taking one second-long samples of the full speaker's sound recording.

8 Scripts

The scripts are implemented in a series of Jupyter notebooks. The kernel used is listed at the start of each notebook. Requirements files are named with the Jupyter kernel used with each script and included in the project file depositories.

Scripts can be executed by the normal Jupyter commands, all cells or one by one from the top to the bottom of each script.

8.1 Pre DUB / ANTBEL 0 73 ipynb

Jupyter notebook with Python for the preprocessing of the recordings of speech samples.

The scripts extract the length and sample rate of each recording file.

They check that each file is WAV, and convert any MP3 to WAV instead.

The sample files, windows defined in this script of 1-second in length, are created from extracting sequential samples in order from each voice recording file and each sample saved as a separate file.

From the recording file names, as specified by Hickey (2004), the scripts extract the speaker age, gender, county, town, town urban or rural flag, town size and a numerical count, if multiple speakers have those personal characteristics.

The scripts also calculate the MFCC, MFCC delta and MFCC delta 2 features from the sound record.

A Pandas DataFrame with one row per sample file is created. Each sample is recorded in one row, with the name and location of the recording and sample files. It is further labelled by a recording number and a sample number and the features extracted from the sample. The DataFrame is saved as a pickle (PKL) file.

On a slower PC with an i5 Intel CPU, the sheets took about 20 minutes to preprocess more than 21,000 samples by saving the files and calculating the features.

Some sample sound file graphs with their MFCC features are graphed at the end of each script.

Code exists to create these graphs for the full set of recordings, but these are commented out to save running-time, over 15 minutes just for the 220-odd recordings, and space on the Jupyter notebook.

8.2 clustering12.ipynb

Fed by the sample files and the pickle files, this script estimates out a series of logistic regressions models for classifying the sample files, each using a different set of input features as the independent variables in the regression.

The scripts begin with the definition of global variables such as the directory names and file locations, which can be overwritten if necessary.

Inputs from the two counties are taken from two pickle files and for the sample files, both from the same *sample_output_directory*.

The dataset is shuffled by speaker rather than by the samples so as to avoid the classification models matching a speaker that appears in both development and test datasets, as described in the report.

Each model is implemented in a separate numbered section, with the model numbers increasing as the sheet scrolls down.

The MFCC features are reshaped for input to the **scikit-learn** models by flattening. These, with or without the age and gender features from the personal characteristics of the speaker. Starting with one variable as input, more are then used in a sequence of models in a stepwise-selection to elicit

Results are shown for all the regressions using a test-set accuracy, a confusion matrix and the ROC plotted and the Area Under the Curve calculated.

k-means clustering models are estimated further down in the sheets to try to discover patterns in the data, using the MFCC input features. Variants using the MFCC and the personal data are estimated and versions of both with 2 or 3 clusters estimated.

8.3 EDA.ipynb

Exploratory Data Analysis: Fed by the sample files and the pickle files, the EDA sheet provides some summary statistics and visualisations about the voice data, such as the breakdown of observation numbers by features such as location and gender.

8.4 TF8 MLP.ipynb

TF MLP Neural network: Fed by the sample files and the pickle files, this worksheet estimates neural network models of the feedforward with hidden layers architecture known as the Multi-Layer Perceptron (MLP) type.

Tensor Flow, Keras and related libraries are used, as specified in the requirements file for this workbook.

MFCC features are used as the inputs along with the personal characteristics of age and location. The Keras Tuner using random search is run to optimise the MLP hyperparameters.

Run-times on a regular Intel i-5 machine average under 10 minutes for the optimisation.

The classification metrics as used in the clustering notebooks are also used here.

8.5 TF10TK CNN.ipynb

TF CNN Convolutional neural network: Fed by the sample files and the pickle files, this worksheet estimates neural network models with the Convolutional Neural Network (CNN) architecture.

Tensor Flow, Keras and related libraries are used, as specified in the requirements file for this workbook.

MFCC features are used as the inputs and the Keras Tuner using random search is run to optimise the CNN hyperparameters.

Run-times on a regular Intel i5 machine average over 10 minutes for the optimisation.

The classification metrics as used in the clustering notebooks are also used here.

8.6 CommonAccent3 Classifier.ipynb

This Jupyter notebook uses one of the classification models from the CommonAccent paper by Zuluaga-Gomez, Ahmed, Visockas & Subakan (2023) through the Hugging Face API. Inputs are the test files for checking available on Hugging Face and then the sample files generated here. The model, its parameters not trained on the project sample data, estimates a classification for each sample.

This runs for more than 1 hour on the slower of the PCs using an i5 CPU.

References

Hickey, R. (2004), *A Sound Atlas of Irish English*, number 48 *in* 'Topics in English Linguistics', Walter de Gruyter: Berlin.

Zuluaga-Gomez, J., Ahmed, S., Visockas, D. & Subakan, C. (2023), CommonAccent: Exploring large acoustic pretrained models for accent classification based on common voice, *in* 'Proc. INTERSPEECH 2023', pp. 5291–5295.