

Profanity Filtering in Speech Contents Using Deep Learning Algorithms

D.D.K.R.W. Dandeniya



Profanity Filtering in Speech Contents Using Deep Learning Algorithms

**A dissertation submitted for the Degree of Master of
Business Analytics**

**D.D.K.R.W. Dandeniya
University of Colombo School of Computing
2021**



Abstract

The worldwide online exposure has significantly increased as a result of the Covid-19 pandemic and remote working, online learning, e-commerce have all become the norm. This has drastically increased the use of hate speech, swear words, racial slurs and many other inappropriate contents on the online platforms. These inappropriate contents are slowly degrading the quality of the online user experiences. Consequently, automatic detection and filtering of such inappropriate contents has grown to be a significant issue for enhancing the calibre of contents. Inappropriate contents may include profanity, violence, misleading information, sexually explicit material, extremism, and it may occur in textual, audio or video forms. In this study, a methodology for profanity filtering in speech contents is proposed. The proposed methodology focuses on identifying the audio segment 'fuck' which is the most frequently used swear word in the English Language. Audio segments related to swear words and non-swear words were collected, annotated, pre-processed, and analysed for the development of a RNN configuration by using Mel Frequency Cepstral Coefficients (MFCCs) as inputs to the model.

Keywords: *Audio Profanity, Speech Recognition, Recurrent Neural Networks, Convolutional Neural Networks, MFCCs*

Declaration

The thesis is my original work and has not been submitted previously for a degree at this or any other university/institute.

To the best of my knowledge, it does not contain any material published or written by another person, except as acknowledged in the text.

Student Name: D.D.K.R.W. Dandeniya

Registration Number: 2019/BA/004

Index Number: 19880049



Signature:

Date: 25/ 02/ 2023

This is to certify that this thesis is based on the work of Mr. D.D.K.R.W. Dandeniya under my supervision. The thesis has been prepared according to the format stipulated and is of acceptable standard.

Certified by: Dr. Manjusri Wickramasinghe

Supervisor Name: Dr. Manjusri Wickramasinghe



Signature

Date: 25/ 02/ 2023

Acknowledgement

I would like to express my deepest gratitude to all those who provided support to undertake this research on “Profanity Filtering in Speech Contents Using Deep Learning Algorithms” successfully.

I would first like to thank my project supervisor Dr. Manjusri Wickramasinghe, Senior Lecturer, University of Colombo School of Computing, for his consistent support and guidance during the running of this project.

My special thanks go to Dr. T. Halloluwa for the support given to me as the course coordinator.

Finally, I want to express my gratitude to everyone who assisted me in making this project a success.

Table of Contents

Declaration.....	iii
Acknowledgement.....	iv
Chapter 1. Introduction.....	1
1.1 Project Overview.....	1
1.2 Motivation.....	1
1.3 Objectives.....	2
1.4 Background of the study.....	2
1.5 Scope of the study.....	3
1.6 Feasibility study.....	3
1.7 Structure of the dissertation.....	4
1.8 Timeline of the project.....	4
1.9 Summary.....	5
Chapter 2. Literature Review.....	6
2.1 Introduction.....	6
2.2 Related Work.....	6
2.3 Summary.....	8
Chapter 3. Methodology.....	10
3.1 Introduction.....	10
3.2 Data Gathering.....	10
3.2.1 Online available datasets.....	10
3.2.2 Recorded dataset.....	11
3.3 Data pre-processing.....	12
3.3.1 Formatting the audio.....	12
3.3.2 Cleaning the audio.....	12
3.4 Model Development and Censoring.....	13
3.4.1 RNN approach with MFCC features as inputs.....	14
3.4.2 CNN approach with spectrogram features as inputs.....	16
3.4.3 Onset detection for long audio speech files.....	16
3.5 Summary.....	16
Chapter 4. Evaluation and Results.....	17
4.1 Introduction.....	17
4.2 Systematic Approach.....	17
4.3 Exploratory Analysis Results.....	18
4.4 Performance Evaluation.....	19
4.4.1 Accuracy.....	19
4.4.2 Precision.....	19
4.4.3 Recall.....	20
4.4.4 F1-Score.....	20
4.5 Summary.....	20
Chapter 5. Results.....	21
Chapter 6. Conclusion.....	26
List of References.....	27
Appendix A.....	29

List of Figures

Figure 1.1 Refined timeline phase 1 from October 2021 to April 2022	4
Figure 1.2 Refined timeline phase 2 from May 2022 to November 2022	4
Figure 3.1 TAPAD open dataset home page view	10
Figure 3.2 Recording the F-words dataset using a custom function	11
Figure 3.3 Recording the clean words dataset using a custom function	11
Figure 3.4 A stereo audio file and a mono audio file	12
Figure 3.5 Flow diagram for dataset creation	13
Figure 3.6 Metadata file for the final dataset	13
Figure 3.7 Model development flow diagram	14
Figure 3.8 High-level process for obtaining MFCC features	14
Figure 3.9 MFCC feature extraction function	15
Figure 3.10 Identified starting and ending timestamps of the words	16
Figure 4.1 Systematics approach flow diagram	17
Figure 4.2 Waveforms for F-words and clean words	18
Figure 4.3 Mel spectrogram plots for a F-word and a clean word	18
Figure 4.4 MFCC plots for a F-word and a clean word	18
Figure 4.5 K-fold cross validation	19
Figure 5.1 Mel filter bands	21
Figure 5.2 Loss and accuracy learning curves for the best model	22
Figure 5.3 Confusion matrix for testing dataset with 26 MFCC features	23
Figure 5.4 Predicted results for pre-processed F-words	23
Figure 5.5 Predicted results for pre-processed clean words	24
Figure 5.6 Predicted results for non-pre-processed F-words	24
Figure 5.7 Predicted results for non-pre-processed clean words	25
Figure A.6.1 Pipeline for the F-Word detection	29
Figure A.6.2 Supportive custom functions used	29
Figure A.6.3 Recording the audio datasets	29
Figure A.6.4 Pre-processing of the speech audio clips	30
Figure A.6.5 Pre-processed outputs	30
Figure A.6.6 RNN Model development	31
Figure A.6.7 Onset detection using silence thresholds	31
Figure A.6.8 Saved RNN models	31
Figure A.6.9 Long audio file split to one second clips	32
Figure A.6.10 CNN model development	32
Figure A.6.11 CNN model prediction for a clean word	33

List of Tables

Table 5.1 Evaluation metrics for the RNN model	21
Table 5.2 Evaluation metrics for the CNN model	22
Table 5.3 Model generalizability results for RNN and CNN approaches	22

List of Abbreviations

ASR	Automatic Speech Recognition
CNN	Convolutional Neural Networks
DSP	Digital Signal Processing
FFT	Fast Fourier Transform
HMM	Hidden Markov Model
LSTM	Long Short-Term Memory
MFCC	Mel Frequency Cepstral Coefficients
NLP	Natural Language Processing
RNN	Recurrent Neural Networks
STFT	Short-Term Fourier Transform
SVM	Support Vector Machines

Chapter 1. Introduction

1.1 Project Overview

Human societies have always sought to report environmental hazards, circulate opinions and facts, transmit knowledge and heritage, and provide entertainment. The relationship between media and human lives has been significant in achieving these objectives. Media includes audio, video, and text, or a combination of these elements. Recent advances in deep learning algorithms, particularly in computer vision, audio analysis, and natural language processing, have attracted substantial attention due to the increased use of multimedia technologies. The field of audio analysis is comprised of various subdomains, including automatic speech recognition, digital signal processing, music classification, and speech emotion detection. These subdomains continue to evolve, highlighting the growing interest and importance of media's significance and its impact on human life.

As of the year 2020, it was estimated that the amount of data in the world to be 44 zettabytes at the dawn of 2020. It is predicted that more than 175 zettabytes of data will be generated by 2025 (SeedScientific, 2021). These data consist of text, audio, video, and many other formats. At least 500 million Tweets are being sent and 720 000 hours of videos are being uploaded to YouTube every day (Maryam Mohsin, 2021). Big data is rapidly becoming one of the main driving forces behind the global economy. A larger portion of this data consists of content. Many of these contents found online contain fake news, hate speech, swearing, inappropriate language, adult content, and racial slurs.

The Covid-19 pandemic has led to a substantial increase in online exposure worldwide, with remote working, online learning, and e-commerce becoming the prevailing norm. This phenomenon has also triggered a surge in the use of inappropriate content on various online platforms, raising concerns over the filtering and censorship of media content. Given the widespread usage of portable and immediate screen time sources among young adults, the exposure to large quantities of offensive and inappropriate content poses a significant risk to their mental well-being. Consequently, the filtering and censoring of online media content has become a pressing social concern, necessitating effective strategies to regulate online content and ensure the safety of individuals. As per research conducted in 2011, a positive association has been found between exposure to profanity in multiple forms of media among adolescents and engagement in physical and relational aggression (Coyne, et al., 2011).

Profanity filtering and content moderation are 2 approaches that can be implemented to reduce the impact created from the heavy use of inappropriate content. While the profanity filters are geared towards censoring and redacting certain words from content, content moderation filters are aimed at screening inappropriate content through implementing a set of pre-arranged rules and guidelines (Wazir, et al., 2021). The usage of offensive language, hate speech, and profanity in online media has emerged as a critical issue that requires immediate attention. While extensive research has been conducted to identify profanity in textual content, comparatively less attention has been given to the detection of profanity in audio and video content. Addressing this gap by identifying and censoring profanity in audio content could significantly enhance the safety and security of online users worldwide.

1.2 Motivation

Covid-19 pandemic has impacted the world in many aspects. The world has already moved to an online culture and people have used to access online content more than ever. The recent research show that Covid-19 lockdowns and restrictions have globally impacted the mental health of people (Simon, et al., 2021). Online platforms have already become the main media of communication. All these have paved the way for heavy use of profanity content recently.

The widespread exposure of individuals to offensive and profane language incorporated in online content is a growing concern. While some individuals may have the requisite education and experience to navigate such material, younger and less experienced users may experience difficulty in handling profane content. Examples of audio incorporated profanity include foul language, swearing, and hate speech. Content creators, media-sharing platforms, and broadcasting companies have an obligation to provide appropriate content through censorship tools. However, a large amount of inappropriate content is still readily available online.

Manual censorship of audio content is a complex, time-consuming, and costly process that requires significant human resources. Moreover, manual censorship may be prone to error due to factors such as fatigue. Therefore, the need for automated models for audio and speech content censorship has increased. This project aims to develop a Deep Learning model that can automatically detect and censor unfavourable speech content in speech audio files.

1.3 Objectives

The main objective of this project is to develop a model to identify the F-words in a speech audio file using advanced deep neural network technologies. Many of the past research on this domain uses speech to text conversion methodologies to identify unfavourable speech content. This project focuses more on using audio characteristics such as the Mel-Frequency Cepstral Coefficients (MFCCs), spectral features, spectrogram images in identifying the F-words.

Apart from the main objective, this study will further focus on censoring the identified swearing words in the speech audio file. Moreover, the model will be optimized to output a censored audio file maintaining a minimal loss of quality compared to the original file. Future advancements to this approach will be to create an application (executable version) or a browser extension for filtering out swearing words in any audio speech file. Major objectives are listed as below.

- To gather a clean dataset with no swearing words and a dataset containing F-words.
- Identifying the timestamps of the F-words.
- Censoring the identified F-words words.
- Generating the final output of the censored audio file with minimal loss of quality.

1.4 Background of the study

In recent times, there has been a surge in the use of Deep Learning for speech recognition, utilizing a blend of acoustic feature extraction methods and diverse classifiers in speech and acoustic recognition (Wazir, et al., 2020). Commonly, a speech recognition system aims to classify different forms of utterances, such as continuous speech, isolated words, connected words, and spontaneous speech. Although research has mainly focused on speech-to-text conversion, there have been limited successful endeavours to use audio characteristics to address the problem.

Recent studies have used conversational and read speech datasets that are clean of foul language utterances to differentiate between profane and non-profane contents. Some examples are LibriSpeech, Wall Street Journal (WSJ) corpus, Google's voice search traffic dataset, Google commands dataset, and speech emotions dataset of conversational speech dialogues. The main classifiers used include Hidden Markov Model (HMM), Support Vector Machine (SVM), Convolutional Neural Networks (CNNs), and Recurrent Neural Networks (RNNs) (Wazir, et al., 2021).

From the background research, 10 main swear words in English language were identified. *Fuck, Shit, Piss off, Dick head, Asshole, Bitch, Bastard, Damn, Cunt, Motherfucker* are among many swearing words being heavily used on speech contents (Matthew Speiser, 2015).

1.5 Scope of the study

- Study will only focus on speech audio files.
- Study is limited to English language audio speech files.
- *Fuck* is the mostly used swearing word and the study will focus more on identifying the F-word first.
- Censoring techniques will be muting and beeping.
- Audio characteristics will be analysed and applied to the final model depending on the contribution of a specific characteristic to the model performance.
- Output audio file will be censored maintaining a minimal loss of quality compared to the original file.

1.6 Feasibility study

For the proposed solution, data needs to be gathered and processed. Online platforms such as Facebook, YouTube, TikTok contain many helpful resources for this specific study. These platforms mainly contain video data which should be downloaded to extract the audio component. This can be simply done using the MoviePy library in python (Geeks of geeks, 2020). Besides these sources, data will be collected from some datasets which have been already used in similar projects.

The study data should be pre-processed to handle audio noise and other defects in data. Afterwards the audio files should be manually annotated. This can be done in two stages (Wazir, et al., 2020). Then the dataset should be divided into training, validation, and test datasets to proceed with the Deep Learning Methodologies. Recent research shows many viable solutions for audio profanity filtering.

The paper (Zhang, et al., 2015) proposes a CNN model-based approach. The study uses spectrogram data and extracts spectral features to train the CNN model. Error detection rate and the F1-score has been used to evaluate model performance. The paper mentions that the models performed well in both 2-class and 10-class problem. In (Wazir, et al., 2021), data augmentation techniques have been used to expand the existing dataset. This study has applied acoustic MFCCs and Mel-spectrograms on two deep-learning structures, namely, CNNs, and RNNs using LSTM cells to identify pre-segmented foul language samples. Paper (Rajalingam, et al., 2021) proposes a whole system consisting of a Digital Signal Processing (DSP) module, Speech Recognition module, Natural Language Processing (NLP) module and an Audio Replacing Module to automatically replace objectionable content in Sri Lankan context. In (Iyer, 2020), BasicMelSpectrogram based approach is used to identify whether an audio file is having a swear word or not. Transfer learning from a resnet18 model has been used to train the dataset and F1 score has been used as the evaluation metric. In the prediction phase, audio file has been split into smaller segments for easier classification.

As many of the proposed solutions are having complex computational costs, detection rate related constraints and dataset related limitations, this domain remain open for potential research in finding optimal ways of audio profanity filtering.

1.7 Structure of the dissertation

The first chapter of the dissertation discusses about the introduction, motivation, background, scope, feasibility of the research and most importantly the objectives of the project. Feasibility study discusses on the available methodologies and technologies which can be used for this specific study. Chapter two focuses on the literature and the related work done in audio profanity filtering domain. Existing similar work are further analysed in chapter two. Chapter three will more focus on the methodology and the implementation steps of the proposed solution. It further describes the technologies and algorithms used in the study. Chapter four introduces the systematic approach used in this project. Furthermore, it explains about evaluation methods and the results of the study. Chapter five consists of the results of the project. It further describes on the RNN and CNN model related results separately. Chapter six consists of the conclusions of the study. Here, a summary of the project is discussed along with the future developments on the study.

1.8 Timeline of the project

	Description	Oct-2021	Nov-2021	Dec-2021	Jan-2022	Feb-2022	Mar-2022	Apr-2022
1	Project finalization	■	■					
2	proposal submission		■	■				
3	Literature review		■	■	■	■	■	■
4	Proposal acceptance				■	■		
5	Report writing				■	■	■	■
6	Data collection							■

Figure 1.1 Refined timeline phase 1 from October 2021 to April 2022

	Description	May-2022	Jun-2022	Jul-2022	Aug-2022	Sep-2022	Oct-2022	Nov-2022
1	Data collection	■	■	■	■			
2	Data pre-processing	■	■	■				
3	Interim report submission	■	■	■				
4	Interim presentation		■	■				
5	Report writing	■	■	■	■	■	■	
6	Exploratory analysis		■	■				
7	Model development		■	■	■	■		
8	Testing and evaluation				■	■	■	
9	Final thesis submission						■	■
10	Final viva							■

Figure 1.2 Refined timeline phase 2 from May 2022 to November 2022

1.9 Summary

First chapter was provided an introduction to the profanity filtering in audio content and the proposed solution. Furthermore, it explained the background of the study and discussed on the feasibility of the study. It has provided a brief understanding on the methodologies that have been used in similar work when identifying profanity in audio contents. It further has elaborated on the scope of the study and about the techniques that can be used for the study.

It also has focused on describing clear objectives of the study and the redefined timeline to achieve those objectives. The structure of the dissertation section has provided an idea on how the chapters are ordered. The second chapter is on the literature review, and it will deeply discuss on existing similar research and related work to the study.

Chapter 2. Literature Review

2.1 Introduction

Chapter 2 describes about existing literature on audio profanity filtering using deep learning algorithms. Existing research, along with their methodologies, algorithms, and concepts are being discussed in this chapter. Furthermore, this chapter focuses on identifying research gaps and optimization methods to existing methodologies.

2.2 Related Work

In (Wazir, et al., 2021), the study has presented a unique dataset (the MMUTM foul language dataset). It has been obtained and analysed at Multimedia University, Malaysia for a film censorship research project in collaboration with Telekom Malaysia. The dataset is included with profane language recordings collected under various environmental conditions. It has been recorded by asking 117 volunteers to utter nine different offensive words, while the highest number of utterances recorded by a speaker was 10 for each of the nine foul words. The dataset is also enriched with manually retrieved and natural data samples from random videos to increase the sample variations.

This study aims to develop a deep learning model for the identification and classification of profanity in speech audio files. The dataset consists of nine categories of profanity, including "Asshole", "Balls", "Bastard", "Bitch", "Cock", "Cunt", "Dick", "Fuck", and "Pussy", as well as a normal class representing casual speech. The dataset was manually labelled in two stages, with data augmentation methods employed to increase dataset quantity. The study employed deep learning networks, specifically convolutional neural networks (CNNs) and recurrent neural networks (RNNs) with long short-term memory (LSTM) cells, along with Mel Frequency Cepstral Coefficients (MFCCs) and Mel-spectrogram images for temporal structure extraction from audio samples. Two strategies were used to distinguish offensive language: RNN with two hidden layers and CNN with five convolutional layers, with the Mel-spectrogram images being the last layer. Model evaluation metrics included accuracy, true positive rate (TPR), false positive rate (FPR), false negative rate (FNR), and F1-Score. The suggested method performed positively in both the two-class and 10-class problems, with FNRs ranging from 2.53% to 3.68% for the two-class and 3.19% to 5.92% for the 10-class problem. Furthermore, the CNN outperformed the RNN and baseline algorithms of pre-trained networks in terms of performance metrics. The study demonstrates the efficacy of CNNs in categorizing and identifying unfavourable spectral images from speech and has potential applications in automated censorship of offensive language in media.

In the paper (Wazir, et al., 2020) the same MMUTM foul language dataset has been used. The collected data has been manually labelled with 2 and 10 annotations. This research has focused more in using CNNs and spectrogram images for the classification of foul language and normal speech using transfer learning. The proposed CNN models for the classification are well-known Deep CNNs including Alexnet (Krizhevsky, et al., 2012), VGG16 (Simonyan & Zisserman, 2015), Googlenet (Szegedy, et al., 2015), and Resnet50 (He, et al., 2016). Transfer learning theories have been applied to transfer the four CNN architectures knowledge. Target speech has been characterized by spectral content which was represented by a vector of consecutive spectral coefficients.

The spectrogram images have been compute based on the total duration of the speech clips, the duration of each spectrogram frame, the time shift between each spectrogram frame, and the number of frequency bands. Convolution process has extracted full features of the spectral content in both frequency and time domains. (Rawat & Wang , 2017), (Karpagavalli & Chandra, 2016). Training of the target model has been carried out in two different methods. In the first method, all layers have been finetuned for Alexnet and the parameters of all the convolutional layers have been fine-tuned

along with the fully connected layer. In the second method, the first 10 layers of VGG16, GoogLeNet and ResNet50 have been frozen. Adaptive Moment Estimation (Adam) method has been used to train both the models. Evaluation has been done based on two metrics: detection error rate (ER) and F1-Score. The proposed model has performed well in both 2-class and 10-class problem with ER ranging from 1.24 to 2.71 for 2 class, 5.49 to 8.30 for 10-class. ResNet50 model has outperformed the other models for all experiments based on ER and F1-score.

In the paper (Rawat & Wang, 2017), a methodology for recognizing and segmenting action from continuous audio-visual data has been discussed. The main contribution of the paper was mentioned as developing a methodology to support audio visual recognition of human manipulation actions. The audio has been represented in the terms of MFCC features. The paper has described more about extracting audio features using MFCC. Words have been considered as sub-actions and the phrases as complete actions. Left-to-right Hidden Markov Model (HMM) which is a widely used method in speech recognition has been used to recognize each single sub action.

Paper (Karpagavalli & Chandra, 2016) also has discussed about audio feature extraction. Their work has been focused on training very deep convolutional networks to add more expressive power and better generalization for end-to-end Automated Speech Recognition (ASR). Network-in-network principles, batch normalization, residual connections and convolutional LSTMs have been applied to build very deep recurrent and convolutional structures. The experiment has been carried out on the Wall Street Journal (WSJ) ASR task. Batch normalization, residual connections methods also have been used to build very deep convolutional towers to process the acoustic features.

Paper (Chaudhari, et al., 2021) has focused in developing an automated video moderation technique to handle the media which is being uploaded at the initial stage itself to provide a profane free platform to the viewers which ranges from kids to the adults. The paper has discussed about a software solution for analysing the video and censoring profane content. The user is also able to identify the profanity level in their video content. The output of this software has been explained as a video where the audio is muted for profane section and the lips of the speaker are pixelated to avoid any sort of reading. Computer vision has been used for the purpose.

The proposed system employs an approach to extract audio files from input videos and convert them into text using Speech-to-Text library, followed by segmentation of text into time frames of 3 seconds. The system then detects profanity in each segment, and if detected, further divides the 3-second segment into 2-second and 1-second segments to obtain the exact timing of the occurrence of the profane word. The system utilizes the identified time segments to mute the audio and blur the video. This approach offers an effective means of identifying profanity in videos, providing greater precision and control over the censoring process.

Paper (Ho Park & Fung, 2017) has described about abusive language detection and classification on twitter dataset. This study has employed three different convolutional neural network (CNN) models for the classification of abusive language into sexist and racist categories. The models have used both character level and word level inputs from the dataset. CharCNN has utilized one-hot encoding of 70 characters for each word in a sentence. WordCNN has converted each word into 300-dimensional embeddings using word2vec. HybridCNN has utilized both character and word-level inputs, and they have been fed into convolutional layers followed by max pooling and SoftMax layer. The models have been compared with Logistic regression, SVM, CharCNN, FastText, and WordCNN, and the two-step approach using HybridCNN has demonstrated the best F1 score and recall value. Specifically, the approach involves first classifying comments into non-abusive and abusive, and then further classifying abusive comments into sexist and racist categories.

In the paper (Zhang, et al., 2015), a robust sound event recognition using convolutional neural networks have been described. In this study, a CNN-based methodology has been proposed for image processing of spectrogram representation, which has been motivated by the similarity of spectrogram to an image. The traditional sound event recognition methods that rely on informative front-end features such as MFCC, with back-end sequencing methods such as HMM, have been shown to perform poorly in the presence of interfering acoustic noise. To overcome this limitation, the proposed method has utilized novel features derived from spectrogram energy triggering, in combination with a CNN, which has demonstrated superior performance under noise-corrupted conditions compared to state-of-the-art approaches. The study claims to be the first to apply CNN in this field, indicating the potential of this approach for improving sound event recognition in noisy environments.

Paper (Rajalingam, et al., 2021), has focused on automatic audio replacement of objectionable content for Sri Lankan Locale. This study proposes a methodology to detect objectionable content in Sinhala, Tamil, and English languages. The audio input has been converted into text format, and a preliminary filtering model has been designed to classify potentially offensive sentences through binary classification. For text classified as offensive, a secondary filtering process has been carried out using a multi-class text classification model that categorizes each word into racist, sexist, cursing, and non-offensive categories. The preliminary filtering process employs a Term Frequency-Inverse Document Frequency (TF-IDF) vectorizer and Support Vector Machine (SVM) algorithm. The multi-class classification models use a combination of Logistic Regression and Countvectorizer for Sinhala and a combination of Multinomial Naïve Bayes and TF-IDF vectorizer for Tamil. The final output replaces the objectionable audio with a predetermined audio input. The results demonstrate the effectiveness of the proposed methodology in identifying and filtering objectionable content in multiple languages.

In (Iyer, 2020), the project has attempted to focus on longer audio samples for profanity filtering. Pytorch and fastaudio tools have been used throughout the project. The project has used the swear words dataset from 'theabuseproject' which is available on GitHub. Multiprocessing and parallel processing methods have been used to improve the computation while training. Basic Mel Spectrogram with 512 Fourier Transforms have been used in the project to extract audio features. Longer audio clips have been split into smaller audio file segments of 0.2 seconds. Researcher has assumed that 0.2 seconds will be long enough in identifying a swear word. The project also has mentioned about the out-of-memory issues and the lesser accuracy on the output.

Paper (Bello, et al., 2005), has discussed about the onset detection and localization in music signals. The scope has been narrowed down to note onset detection in musical signals. This paper has further described about related concepts such as transients and attacks. The attack of the note is the time interval during which the amplitude envelope increases. Transient has been described as the period during with the excitation is applied and the damped leaving only the slow decay at the resonance frequencies of the bodies. Researcher has focused on a detection function-based methodology which is derived at a lower sampling rate to which a peak-picking algorithm is applied to locate the onsets.

2.3 Summary

Second chapter has described deeply about the available literature, related and similar work done in audio profanity filtering domain. There is not much research work done on this specific domain due to the domain complexity, data gathering and processing complexity and the need of prior domain expertise. Many of the relevant work has used Convolutional Neural Networks (CNN) along with Mel Frequency Cepstral Coefficients (MFCCs) methodologies in identifying profane audio content. Available work also has used complex methodologies and many of the datasets have not been exposed. Many have focused on speech to text conversion while carrying out the project. There were

only a limited number of researches that has focused on audio characteristics like Fourier transform and spectrogram in identifying profane content in audio speech files. The second step of the project: censoring the profane content after identification, has not been well explained. None of the similar work has extended their focus into the output audio file quality. This project tries to minimize the research gaps by focusing more into audio characteristics of the speech files and using light weight algorithms in identifying profane audio content. It will also be extended to retain the original quality of the output audio file.

Chapter 3. Methodology

3.1 Introduction

Chapter 3 describes about proposed methodology of the project. This will further describe the approach under 3 sub sections: data gathering, data pre-processing and data augmentation, model development and censoring. Each step will be discussed in the next sections.

3.2 Data Gathering

Data gathering for the project was carried out under following steps.

1. F-Words and clean words from the online available datasets were gathered.
2. F-Word and their variants were recorded from male and female participants.
3. Clean words were recorded from the male and female participants.
4. Rhyming words with the F-Word were recorded from the male and female participants.

3.2.1 Online available datasets

In this study, ‘The Abuse Project Audio Dataset (TAPAD)’ online available dataset has been used as the F-word dataset. TAPAD is the world’s largest profanity audio dataset with more than 26 000 iterations of swear words across many languages. Only the F-words were extracted from this dataset. 339 audio clips of F-words and its variants were extracted from the above dataset. The dataset consisted of multiple utterances for the F-words and its variants. Fig. 3.1 shows the homepage of the TAPAD open dataset with the project details.

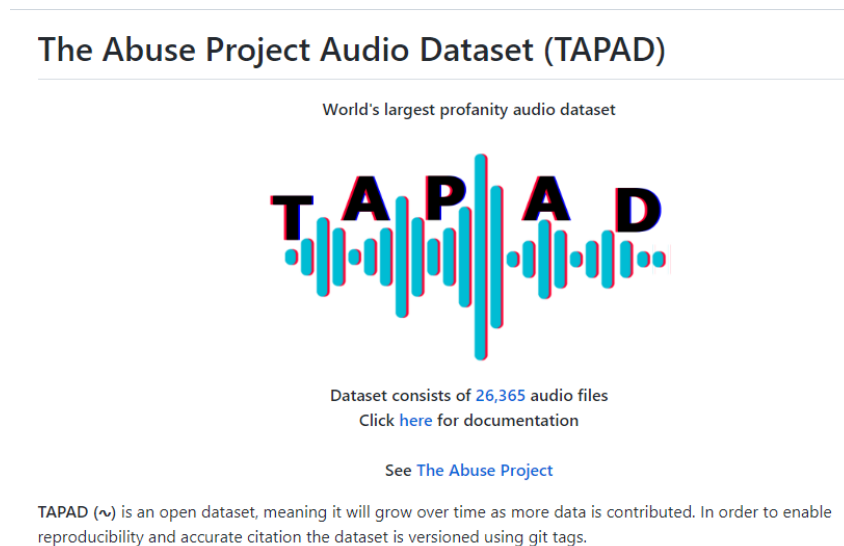


Figure 3.1 TAPAD open dataset home page view

For the clean word utterances, ‘Google Speech Commands’ dataset was used. This is a set of one-second .wav audio files, each containing a single spoken English word or background noise. These words are from a small set of commands and are spoken by a variety of different speakers. The dataset consisted of 105, 829 audio clips. 4000 random clips were selected to form the final dataset.

Other than online readily available datasets, individual audio mp3 files found online were also treated manually to create a dataset of 40 clean words and 15 swearwords.

3.2.2 Recorded dataset

Two Female and four male participants were selected and were asked to speak utterances of following words for F-words, clean words and rhyming words. 16 000 Hz sampling rate was used to record the 1 second long audio clips. This was carried out with the help of a custom python function.

F-words - *fuck, fuckass, fuckbag, fuckboy, fuckbrain, fuckbutt, fuckbutter, fucked, fuckface, fuckhead, fuckhole, fucking, fucknut, fuckoff, fucks, fuckstick, fucktard, fuckup, fuckward, fuckwitt*

Clean words - *backward, bed, bird, cat, dog, down, eight, five, follow, forward, four, go, happy, house, learn, left, marvin, nine, no, off, on, one, right, seven, sheila, six, stop, three, tree, two, up, visual, wow, yes, zero*

Rhyme words - *duck, luck, truck, pluck, buck, chuck, stuck, fun, fly, fuse, funny, fast, fuss*

Recorded audio clips were manually checked and handpicked considering the audio quality and the clearness of the audio. 800 clean words, 450 F-words and 150 rhyme words were recorded and selected for the final dataset. Fig. 3.2 and Fig. 3.3 show the dataset recording process for the F-words and the clean words.

```
In [4]: record_path = 'raw_data/ran_fwords'

isExist = os.path.exists(record_path)
if not isExist:
    os.makedirs(record_path)

In [5]: record_audio_all_and_save(record_path, swear, 16000, n_times=3, seconds = 1)

To start recording input your name: ran
Recording for the word fuck

Press enter to record next or to stop press ctrl + C (1/5):
Press enter to record next or to stop press ctrl + C (2/5):
Press enter to record next or to stop press ctrl + C (3/5):
Press enter to record next or to stop press ctrl + C (4/5):
Press enter to record next or to stop press ctrl + C (5/5):

Recording for the word fuckass

Press enter to record next or to stop press ctrl + C (1/5):
Press enter to record next or to stop press ctrl + C (2/5):
Press enter to record next or to stop press ctrl + C (3/5):
Press enter to record next or to stop press ctrl + C (4/5):
Press enter to record next or to stop press ctrl + C (5/5):
```

Figure 3.2 Recording the F-words dataset using a custom function

```
In [8]: record_path = 'raw_data/ran_cleanwords'

isExist = os.path.exists(record_path)
if not isExist:
    os.makedirs(record_path)

In [9]: record_audio_all_and_save(record_path, clean, 16000, n_times=3, seconds = 1, swear_clean = 'clz0_')

To start recording input your name: ran
Recording for the word backward

Press enter to record next or to stop press ctrl + C (1/5):
Press enter to record next or to stop press ctrl + C (2/5):
Press enter to record next or to stop press ctrl + C (3/5):
Press enter to record next or to stop press ctrl + C (4/5):
Press enter to record next or to stop press ctrl + C (5/5):

Recording for the word bed

Press enter to record next or to stop press ctrl + C (1/5):
Press enter to record next or to stop press ctrl + C (2/5):
Press enter to record next or to stop press ctrl + C (3/5):
Press enter to record next or to stop press ctrl + C (4/5):
Press enter to record next or to stop press ctrl + C (5/5):
```

Figure 3.3 Recording the clean words dataset using a custom function

3.3 Data pre-processing

Data pre-processing in audio files are different from the general data pre-processing techniques used in other Machine Learning approaches. These can be mainly categorized into two sections.

1. Formatting the audio file
2. Cleaning the audio file

In this project, multiple approaches have been used for the above mentioned two sections.

3.3.1 Formatting the audio

In the present study, the MP3 files were subjected to a compression format with loss of information. Thus, as a first step in formatting the audio clips, they were converted to the WAV format. The Pydub audio manipulation library, implemented in python, was employed for the MP3 to WAV conversion. The online data gathering method generated files in MP4 format as they were downloaded from YouTube. Hence, the youtube_dl python library was used for extracting the audio files directly from YouTube.

In terms of acoustic analysis, human language is primarily produced as mono signal. Consequently, mono sounds were preferred over stereo sounds. Stereo sounds involve recording and playback using two audio channels, while mono signals use a single audio channel. The Pydub python library was used for the mono conversion. Finally, all the WAV files were resampled to a 16 000 Hz sampling rate, which is a preferred rate for speech audio content. Fig. 3.4 illustrates a comparison between a stereo audio wave file and a mono audio wave file.

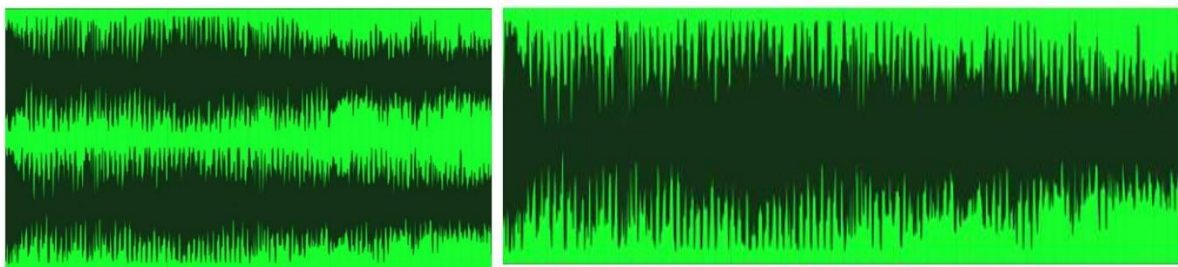


Figure 3.4 A stereo audio file and a mono audio file

3.3.2 Cleaning the audio

Audio files from online available datasets were mostly cleaned. But the recorded audio clips and online downloaded individual clips had background noise, random noise and other disturbances embedded into the audio file. These were handled using librosa and pydub python packages. Leading and tailing silences were detected and whitespaces were removed from the audio clips. A silence threshold value of -30.0 dB was used. The processed audio clips were then overlaid to a 1 second background noise audio clip for audio resizing.

Fig. 3.5 shows the flow diagram for the final dataset creation.

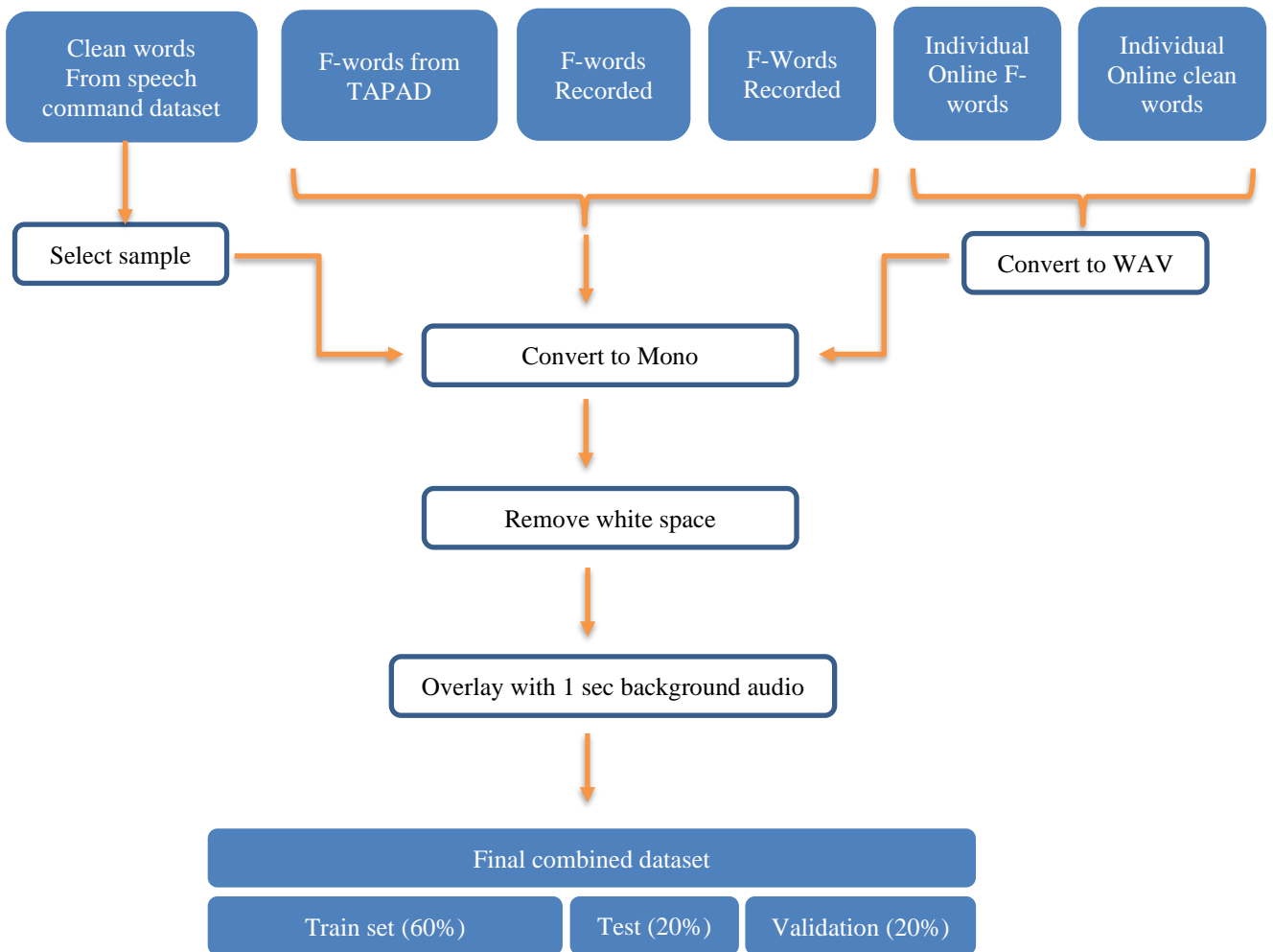


Figure 3.5 Flow diagram for dataset creation

3.4 Model Development and Censoring

This study was carried out as a classification problem to distinguish between a F-Word and a non-F-Word. The audio clips were renamed with ‘clz0’ and ‘clz1’ prefixes while pre-processing. These file paths were then passed through a ‘labeller’ function to generate the final metadata file. The metadata file was used as the reference while training the model. Fig. 3.6 shows the final metadata file created for the combined audio dataset.

file	target	targetnum
0 raw_data/combined_ds/clz0_clean_69363.wav	Clean_word	0
1 raw_data/combined_ds/clz0_clean_62145.wav	Clean_word	0
2 raw_data/combined_ds/clz0_clean_75285.wav	Clean_word	0
3 raw_data/combined_ds/clz0_clean_42998.wav	Clean_word	0
4 raw_data/combined_ds/clz0_clean_1143.wav	Clean_word	0
5 raw_data/combined_ds/clz0_clean_51989.wav	Clean_word	0
6 raw_data/combined_ds/clz0_clean_47686.wav	Clean_word	0
7 raw_data/combined_ds/clz0_chaveen_nine4.wav	Clean_word	0
8 raw_data/combined_ds/clz0_clean_46569.wav	Clean_word	0
9 raw_data/combined_ds/clz0_clean_13680.wav	Clean_word	0
10 raw_data/combined_ds/clz1_channaka_fuckstick2.wav	Swear_word	1
11 raw_data/combined_ds/clz0_channaka_learn3.wav	Clean_word	0
12 raw_data/combined_ds/clz0_clean_925.wav	Clean_word	0

Figure 3.6 Metadata file for the final dataset

For this study, a virtual anaconda environment was created with the python 3.7 and following packages were used.

- Audio manipulation – pydub, librosa, fastcore, sounddevice, youtube-dl
- General – pandas, numpy, ffmpeg, jupyter notebook
- Deep Learning – tensorflow, keras

An exploratory analysis was done to get more visibility and understanding on the two classes. Two different deep learning approaches were tested against their model performance in order to select the best model for the study. These approaches are listed below.

1. RNN approach with MFCC features as inputs
2. CNN approach with spectrogram features as inputs

Fig. 3.7 shows the model development flow diagram for the above described process.

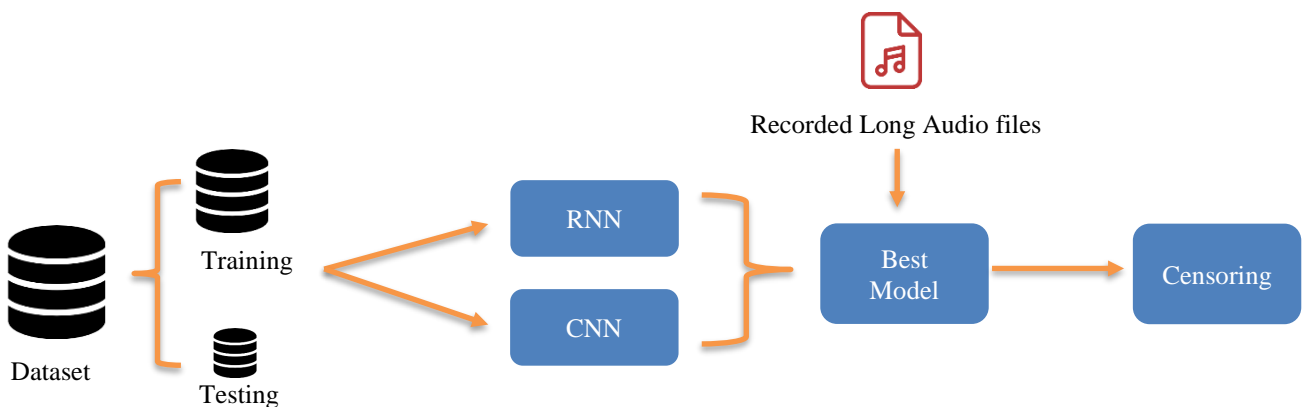


Figure 3.7 Model development flow diagram

3.4.1 RNN approach with MFCC features as inputs

In this approach, a batch transformation was carried out on the full dataset and MFCC features were extracted and saved in the pickle file format. Fig. 3.8 shows the high-level process for obtaining MFCC features.

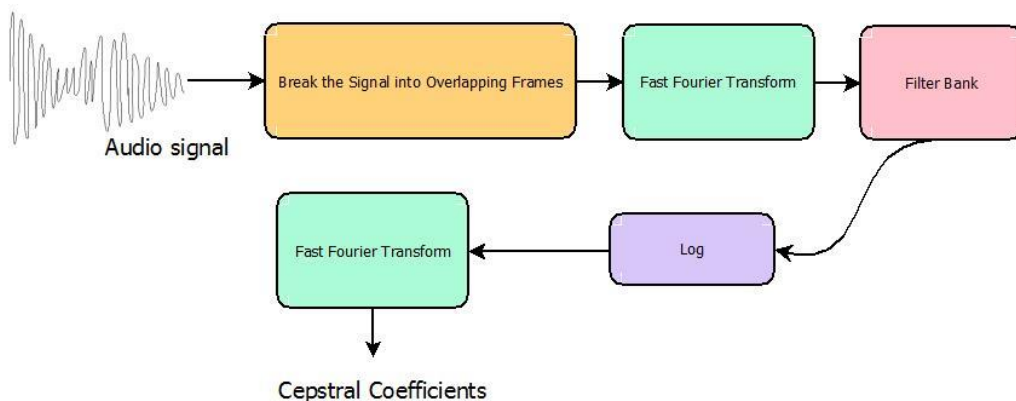


Figure 3.8 High-level process for obtaining MFCC features

The MFCC feature extraction technique is the widely used technique for extracting the features from audio signals. The feature extraction process includes windowing the signal, applying the Discrete Fourier Transform (DFT), taking the log value of the magnitude and then warping the frequencies on a Mel scale followed by a Discrete Cosine Transform (DCT). The MFCC model takes the first twelve coefficients of the signal after applying the Inverse Discrete Fourier Transform (IDFT) operations. Along with these twelve coefficients, energy of the signal is also considered as a feature. First order derivatives and second order derivatives of these thirteen features were also considered in the study. Derivatives were calculated by taking the difference of the coefficients between the samples of the audio signal. Fig shows the defined function for MFCC feature extraction in python. Here, Librosa packages was used for the MFCC feature extraction. Fig. 3.9 shows MFCC feature extraction fuction.

```
def extract_mfcc(meta_file_name):
    print('MFCC extraction begins...\n')

    meta_file = pd.read_csv(meta_file_name)
    meta_dict = meta_file[['targetnum', 'file']]

    all_data = []

    train_meta_dict = {
        0: meta_dict[meta_dict['targetnum'] == 0].file.values.tolist(),
        1: meta_dict[meta_dict['targetnum'] == 1].file.values.tolist()
    }

    for class_label, list_of_files in train_meta_dict.items():
        for single_file in list_of_files:
            audio, sample_rate = librosa.load(single_file, sr=None)
            mfcc = librosa.feature.mfcc(y=audio, sr=sample_rate, n_mfcc=40)
            mfcc_processed = np.mean(mfcc.T, axis=0)
            all_data.append([mfcc_processed, class_label])
            print(f"Info: Succesfully Preprocessed Class Label {class_label}")

    df = pd.DataFrame(all_data, columns=["feature", "class_label"])

    print('\nMFCC extraction completed !!!')

    return df

file_name = 'train.csv'
df_train = extract_mfcc(file_name)

MFCC extraction begins...

Info: Succesfully Preprocessed Class Label 0
Info: Succesfully Preprocessed Class Label 1

MFCC extraction completed !!!
```

Figure 3.9 MFCC feature extraction function

Audio data set was split into training, testing and validation datasets considering 60% of data for the training set, 20% of data for the testing set and 20% of data for the validation set. A Recurrent Neural Network with a dense layer of 256 neurons with ReLU activation was trained for the audio MFCC features. Dropout was used as 0.5 to avoid model overfitting. A layer with two neurons and softmax activation was used as the output layer. Dataset was trained for multiple iterations and the best model was selected for prediction on longer audio clips. For audio censoring, a chime sound and 30 second recorded audio speech files were used. The 30 second file was split in to one second audio chunks and the one second audio chunks were predicted for a F-word or a clean word by using the best model trained. Audio chunks predicted as F-words were overlayed with the chime.wav file. Finally, all the audio chunks were combined in the correct order to generate the censored output audio file.

3.4.2 CNN approach with spectrogram features as inputs

In the CNN approach, the audio waveforms were transformed from the time-domain signals into the frequency-domain signals by computing the Short-Time Fourier Transform (STFT) in order to convert the waveforms as spectrograms. Spectrogram shows frequency change over time and it can be represented as 2D images. In the FFT operation, all time information is lost. Hence, a STFT operation was used. STFT splits the signal into windows of time and runs a Fourier Transform on each window, preserving some time information.

124 features were extracted from the spectrogram images and a simple Convolutional Neural Network was trained. The model was enriched with resizing, normalization, conv2D, carpooling, dropout, flattened and dropout layers followed by an output layer of 2 neurons. The model was configured with Adam optimizer and the cross-entropy loss and trained for multiple iterations with early stopping conditions. For audio censoring, a chime sound and 30 second recorded audio speech files were used similar to the RNN approach described above.

3.4.3 Onset detection for long audio speech files

In this study, an onset detection methodology has been tested to generate a better censored speech audio output. Pydub detect_nonsilent package has been used with a silence threshold setting of -30 dB and a minimum silence length setting on 300 milliseconds. This helped to identify the starting and ending timestamps of the speech words considering the silence threshold and minimum silence between two words. Starting and ending timestamps were used to split the long audio files into audio chunks containing only the identified word. Fig. 3.10 show the output of the custom function with starting and ending timestamps for the audio chunks.

```
In [89]: for chunks in nonsilent_data:
         print( [chunk for chunk in chunks])

[1088, 1100]
[1198, 1563]
[1906, 1949]
[2603, 3044]
[3262, 3478]
[4487, 4549]
[4985, 5012]
[5412, 5435]
[5719, 5857]
[6774, 6955]
[7182, 7230]
[7823, 8213]
[8997, 9009]
```

Figure 3.10 Identified starting and ending timestamps of the words

3.5 Summary

Chapter 3 has discussed on the proposed methodology, technical knowledge, and tools to be used while developing the project. It further described about the steps from data gathering to censoring the final audio output. Next chapter will focus on the high-level systematic approach and the evaluation criteria of the project.

Chapter 4. Evaluation and Results

4.1 Introduction

This chapter is mainly focused on the end-to-end systematic approach of the project and evaluation of the results.

4.2 Systematic Approach

Fig. 4.1 shows the systematic approach that has been used in this project. This is a more generalized version of the methodologies described in chapter 3. The project will follow the steps from data gathering to final audio output censoring. The best model will be selected considering the performance evaluations.

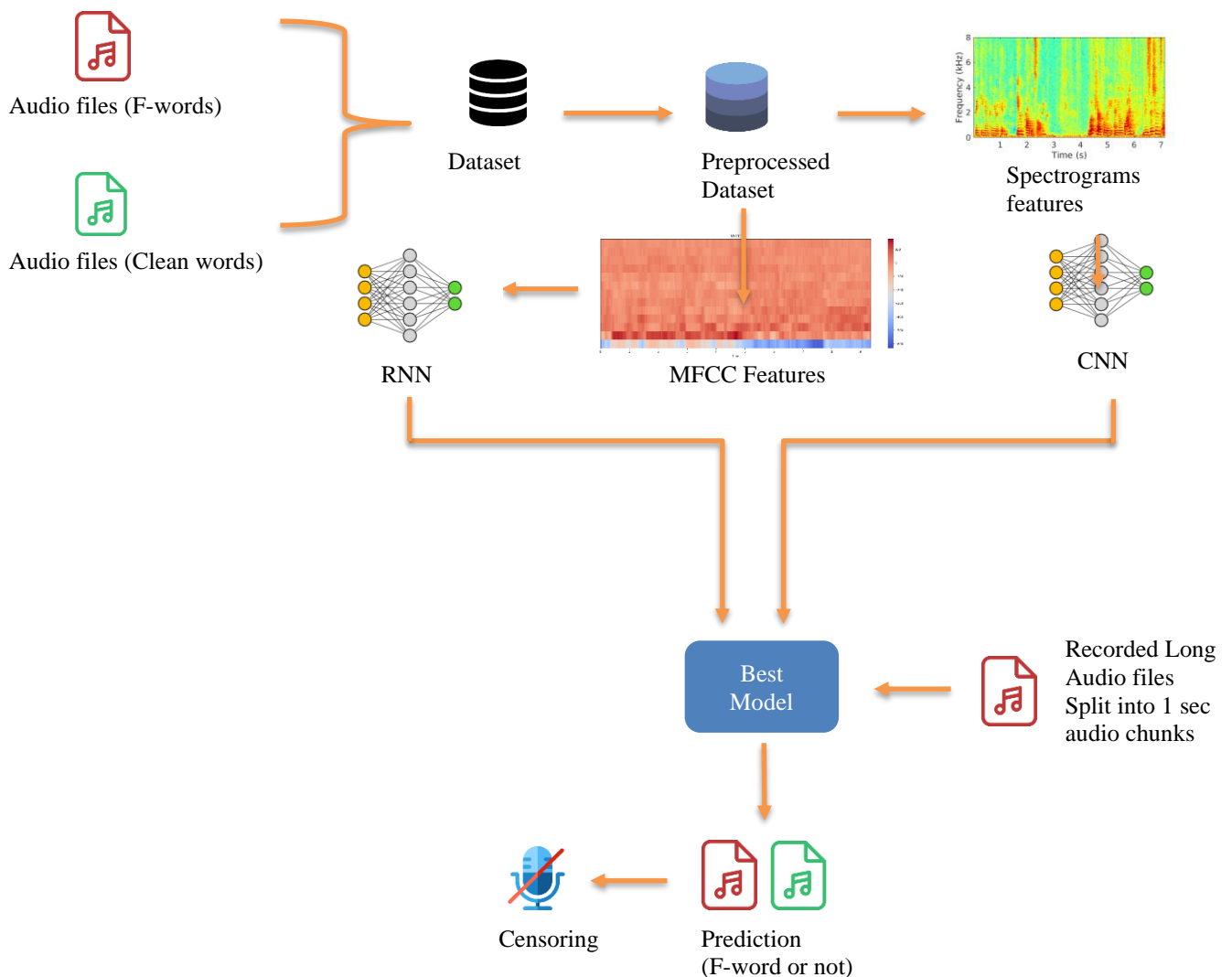


Figure 4.1 Systematics approach flow diagram

4.3 Exploratory Analysis Results

Audio waveforms for F-Words and the clean words were plotted and observed. Fig. 4.2 shows how the waveform changes for a set of random F-words and clean words.

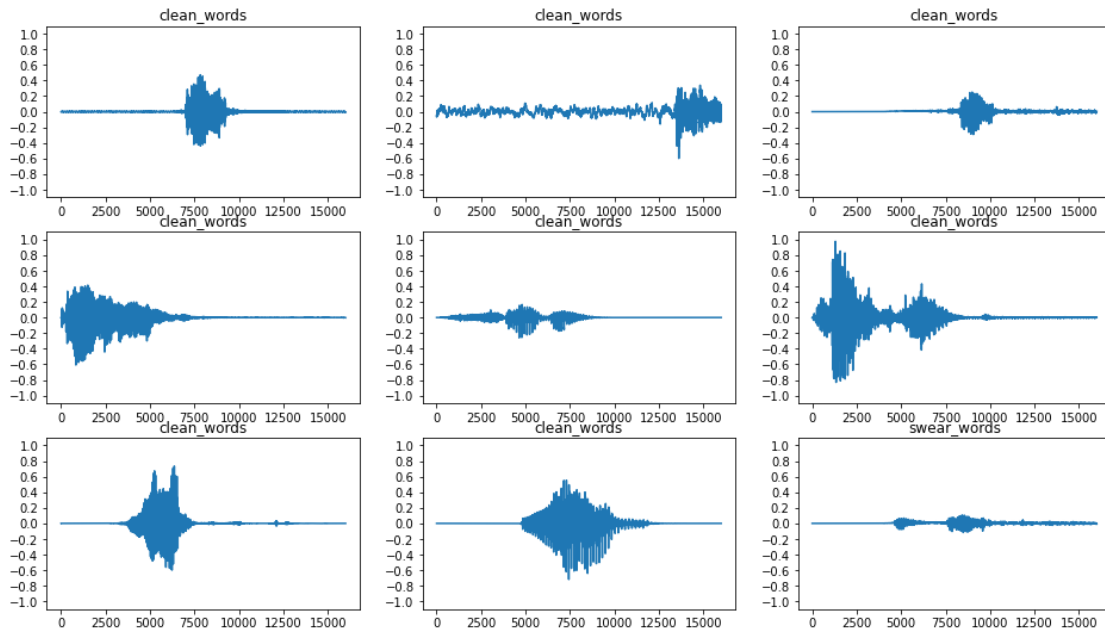


Figure 4.2 Waveforms for F-words and clean words

Along with the audio waveform, the spectrogram and the MFCC plots were visualized to understand the distribution of frequencies along the time domain. Fig. 4.3 shows the Mel spectrogram plots for a random F-word and Fig. 4.4 shows the MFCC plots for a random clean word.

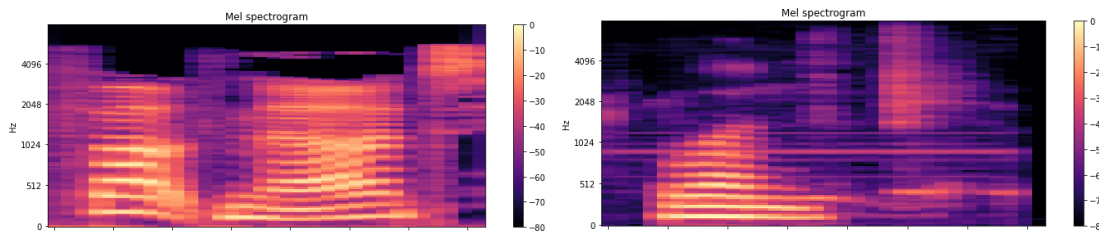


Figure 4.3 Mel spectrogram plots for a F-word and a clean word

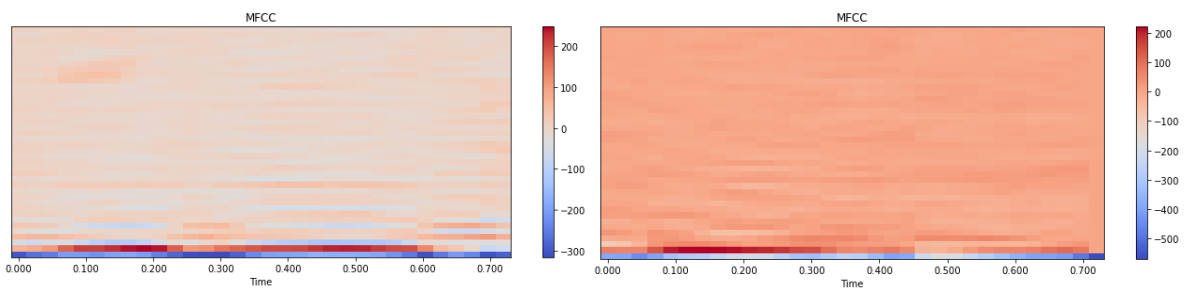


Figure 4.4 MFCC plots for a F-word and a clean word

The color dimension on the spectrogram gives the information on the amplitude of the audio waveform. In the MFCC plot, it gives details about the feature values for the MFCC features. It can be observed that the Mel spectrogram and the MFCC plots are different for F-words and clean words.

4.4 Performance Evaluation

In the RNN approach, three models were evaluated with respect to the classification performance for MFCC features 13, 26 and 39. In the CNN approach, the model was trained for 10 iterations and checked for the model performance metrics. To handle the biasness of the dataset, a k-fold cross validation was used. This helped overcoming the variance issue by spinning the dataset into k number of equal subsets. Here, 5-fold cross validations were applied and at each iteration, one subset was kept as the validation set while the other 4 subsets were used for training. Fig. 4.5 shows the k-fold cross validation process.

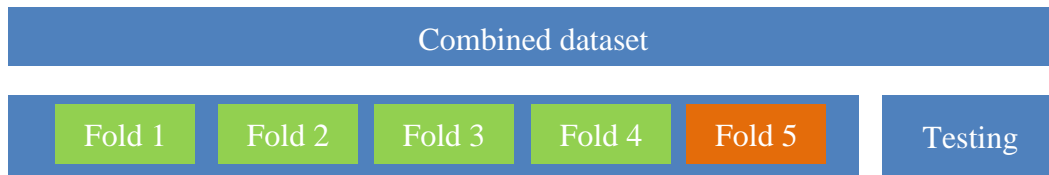


Figure 4.5 K-fold cross validation

Furthermore, the study used a classification report for evaluating the classification metrics. This includes Precision, Recall and F1-score. Accuracy was not given much weightage considering the class imbalance of the dataset. Following were considered while calculating Precision, Recall and the F1-Score.

- True Negative (TN): Actual clean word predicted as clean word
- True Positive (TP): Actual F-word predicted as F-word
- False Negative (FN): Actual F-word predicted as clean word
- False Positive (FP): Actual clean word predicted as F-word

True Positives and True False Negatives represent the actual F-words and they were prioritized while selecting the model evaluation metrics.

4.4.1 Accuracy

Accuracy is the ratio of correctly predicted observation to the total observations. Here, accuracy is not considered for model performance evaluation considering the imbalance nature of the dataset.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

4.4.2 Precision

This is the ratio between correctly predicted positive observations to the total predicted positive observations. This can be seen as a measure of quality.

$$Precision = \frac{TP}{TP + FP}$$

4.4.3 Recall

Also known as Sensitivity. This is a measure of quantity. Recall is the ratio of correctly predicted positive observations to the all observations in actual positive class.

$$Recall = \frac{TP}{TP + FN}$$

4.4.4 F1-Score

F1 Score is the weighted average of Precision and Recall. This metric score takes both false positives and false negatives into account. F1 is usually more useful than accuracy, especially if you have an uneven class distribution.

$$F1\ Score = \frac{2 * Recall * Precision}{Recall + Precision}$$

4.5 Summary

Fourth chapter has described on the end-to-end systematic approach of the project. Furthermore, it has extended the focus on model performance evaluation methodologies and on the progress evaluation of the project.

Chapter 5. Results

Before predicting on long recorded speech audio clips, the model was tested with the testing and validation datasets with one second audio clips. In the RNN approach with MFCC features as the inputs to the model, three models were trained for 10 iterations considering inputs as 13 MFCC features, 26 MFCC features and 39 MFCC features. Table 5.1 show the precision, recall and F1 metric values for the testing data and validation data.

Table 5.1 Evaluation metrics for the RNN model

Dataset	MFCC count	Precision	Recall	F1-score
testing	13	0.93	0.95	0.94
	26	0.97	0.98	0.98
	39	0.96	0.98	0.97
validation	13	0.87	0.93	0.90
	26	0.95	0.97	0.96
	39	0.94	0.97	0.95

Above results depict that for the RNN approach, using 26 MFCC features gives the best model performance for the classification task. One of the last steps in the MFCC's calculation is measuring the energy in the filter banks. This step is followed in order to reduce the dimensionality of input vector (amplitude spectrum), as well as capture its envelope. Those triangular filters are spaced over the Mel scale as shown in Fig. 5.1 below.

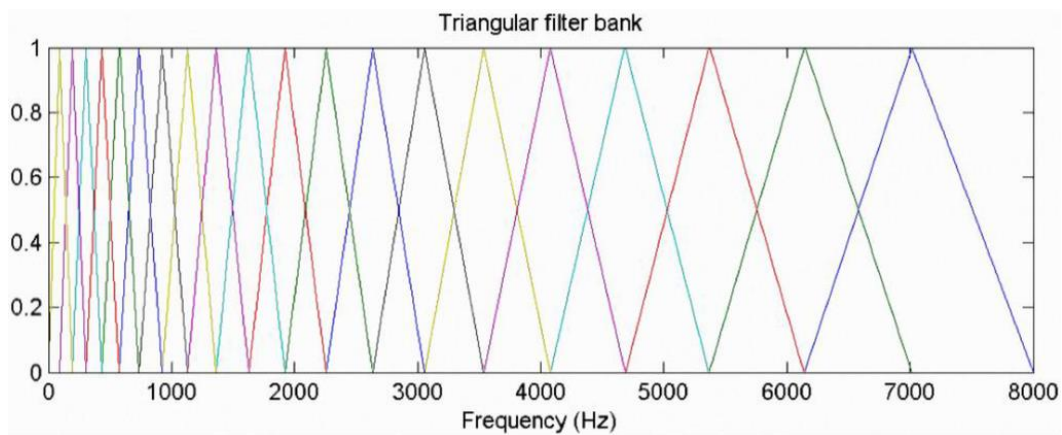


Figure 5.1 Mel filter bands

It can be observed that a very good resolution is represented in low frequencies. Exactly opposite is true for higher frequencies. MFCC's are suited for speech-related tasks and most of the information is in lower frequencies. This has made the Hidden Markov Model Toolkit to use 20 filter banks as the default value for speech audio files. In this study, the model with 26 MFCC features has performed better than the other two models. This can be described by the random higher frequencies generated while uttering the F-Words.

For the CNN approach with spectrogram features as inputs, table 5.2 shows the precision, recall and F1 metric values for the testing data and validation data.

Table 5.2 Evaluation metrics for the CNN model

Dataset	Precision	Recall	F1-score
testing	0.92	0.94	0.93
validation	0.90	0.91	0.90

Other than the pre-processed audio clips, the two models were manually evaluated on non-pre-processed audio clips to check for the generalization capabilities of the two models. These audio clips were recorded at 16 000 Hz and none of the pre-processing steps were applied as pre-processing steps. Three 30 second sentences with different number of F-words and clean words were recorded and used for the manual evaluation process. Table 5.3 shows the evaluation results for the model generalizability.

Table 5.3 Model generalizability results for RNN and CNN approaches

Model	Recorded Sentence	Tested F-words	Correct Predictions	Tested Clean Words	Correct Predictions
RNN	Sentence1	10	6	48	42
	Sentence2	11	6	52	46
	Sentence3	12	8	55	49
CNN	Sentence1	10	4	48	43
	Sentence2	11	5	52	46
	Sentence3	12	6	55	50

Comparing the RNN and CNN approach results from the above table, it was observed that the CNN model had performed better with the clean not pre-processed words than the RNN approach. However, for both pre-processed audio clips and not pre-processed audio clips, it was concluded that the RNN model performed well with the intended task of identifying the F-words in speech audio clips. RNN approach with 26 MFCC features as inputs was selected as the best model.

Fig 5.1 shows the loss and accuracy learning curves for the best model for 10 epochs and Fig. 5.2 shows the confusion matrix for testing dataset with 26 MFCC features.

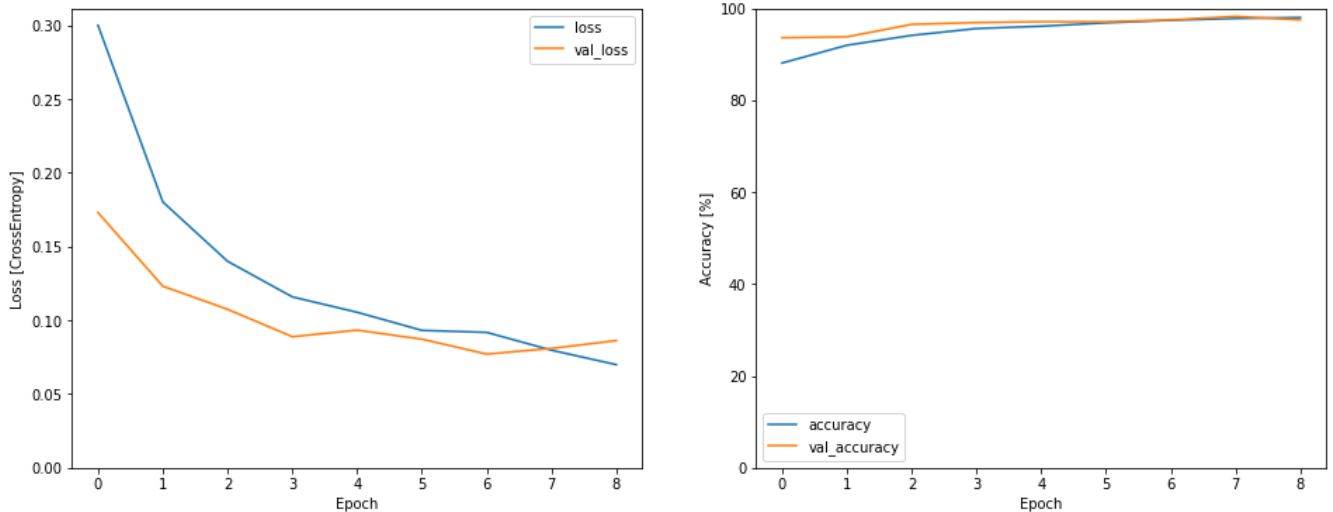


Figure 5.2 Loss and accuracy learning curves for the best model

```

30/30 [=====] - 0s 878us/step
           precision    recall  f1-score   support

     0       0.99       0.99       0.99       780
     1       0.96       0.97       0.97       158

 accuracy          0.99          938
 macro avg         0.98          938
 weighted avg      0.99          938

Confusion matrix, without normalization
[[774  6]
 [ 4 154]]

```

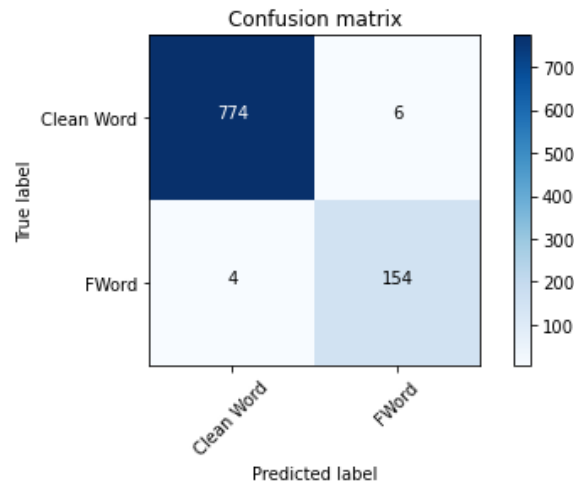


Figure 5.3 Confusion matrix for testing dataset with 26 MFCC features

Fig. 5.3 and Fig. 5.4 shows how the best model performed on one second pre-processed F-words audio clips and clean words audio clips respectively.

Predicting for preprocessed FWords

```

In [7]: filename = 'raw_data/Validation_WAV/clz1_channaka_fuck3.wav'
        pu.make_prediction_preprocessed(filename, model1)

[[8.696508e-08 9.999999e-01]]
FWord Detected for (raw_data/Validation_WAV/clz1_channaka_fuck3.wav)
Confidence: [0.9999999]

In [8]: filename = 'raw_data/Validation_WAV/clz1_chaveen_fuckward4.wav'
        pu.make_prediction_preprocessed(filename, model1)

[[4.0327891e-04 9.9959677e-01]]
FWord Detected for (raw_data/Validation_WAV/clz1_chaveen_fuckward4.wav)
Confidence: [0.9995968]

In [9]: filename = 'raw_data/Validation_WAV/clz1_fuckboy (13).wav'
        pu.make_prediction_preprocessed(filename, model1)

[[0. 1.]]
FWord Detected for (raw_data/Validation_WAV/clz1_fuckboy (13).wav)
Confidence: [1.]

```

Figure 5.4 Predicted results for pre-processed F-words

Prediction for preprocessed clean words

```
In [4]: filename = 'raw_data/Validation_WAV/clz0_clean_39.wav'
        pu.make_prediction_preprocessed(filename, model1)

[[1. 0.]]
FWord Not Detected
Confidence: [1.]

In [5]: filename = 'raw_data/Validation_WAV/clz0_clean_48407.wav'
        pu.make_prediction_preprocessed(filename, model1)

[[1.000000e+00 4.676099e-17]]
FWord Not Detected
Confidence: [1.]

In [6]: filename = 'raw_data/Validation_WAV/clz0_clean_96243.wav'
        pu.make_prediction_preprocessed(filename, model1)

[[1.000000e+00 1.130780e-26]]
FWord Not Detected
Confidence: [1.]
```

Figure 5.5 Predicted results for pre-processed clean words

Fig. 5.5 and Fig. 5.6 shows how the best model performed on not pre-processed F-words audio clips and clean words audio clips respectively.

Predicting for non preprocessed FWords

```
In [13]: filename = 'raw_data/Validation_WAV/Not_Preprocessed/clz1_channaka_fuck3.wav'
        pu.make_prediction_preprocessed(filename, model1)

[[2.156743e-10 1.000000e+00]]
FWord Detected for (raw_data/Validation_WAV/Not_Preprocessed/clz1_channaka_fuck3.wav)
Confidence: [1.]

In [14]: filename = 'raw_data/Validation_WAV/Not_Preprocessed/clz1_chaveen_fucknut4.wav'
        pu.make_prediction_preprocessed(filename, model1)

[[0.0021655 0.9978345]]
FWord Detected for (raw_data/Validation_WAV/Not_Preprocessed/clz1_chaveen_fucknut4.wav)
Confidence: [0.9978345]

In [15]: filename = 'raw_data/Validation_WAV/Not_Preprocessed/clz1_kan_fuckup3.wav'
        pu.make_prediction_preprocessed(filename, model1)

[[2.4774565e-06 9.9999750e-01]]
FWord Detected for (raw_data/Validation_WAV/Not_Preprocessed/clz1_kan_fuckup3.wav)
Confidence: [0.9999975]

In [19]: filename = 'raw_data/Validation_WAV/Not_Preprocessed/clz1_fuckhead (7).wav'
        pu.make_prediction_preprocessed(filename, model1)

[[9.9999881e-01 1.1747688e-06]]
FWord Not Detected
Confidence: [0.9999988]
```

Figure 5.6 Predicted results for non-pre-processed F-words

Predicting for non preprocessed CleanWords

```
In [11]: filename = 'raw_data/Validation_WAV/Not_Preprocessed/clz0_chaveen_zero4.wav'  
         pu.make_prediction_preprocessed(filename, model1)  
  
         [[9.9999845e-01 1.5715626e-06]]  
         FWord Not Detected  
         Confidence: [0.99999845]
```

```
In [12]: filename = 'raw_data/Validation_WAV/Not_Preprocessed/clz0_zero_fbf3dd31_nohash_0.wav'  
         pu.make_prediction_preprocessed(filename, model1)  
  
         [[1.000000e+00 9.146558e-10]]  
         FWord Not Detected  
         Confidence: [1.]
```

Figure 5.7 Predicted results for non-pre-processed clean words

Chapter 6. Conclusion

In order to solve the project of identifying F-words within an audio speech file, 3 Recurrent Neural Network models and a CNN model for binary classification were implemented. For the RNN models, each model was fed with different MFCC features. The model with 26 MFCC features performed well with a precision of 95% and a recall of 97% on the validation audio dataset. When compared to the best model in RNN approach, the CNN model showed lesser performance. Hence, RNN model for 26 MFCC features was selected as the best model for the study carried out.

Although RNN performed well for the dataset, it cannot be generalized that RNN performs better over CNN for profanity detection in speech audio files. A larger audio dataset than the dataset used for this study, should be trained for multiple iterations to compare between RNN and CNN model performance for profanity identification task on audio speech files. This study was limited to a smaller dataset due to the computational complexities involved with audio processing. One of the future extensions towards this study would be to use a larger dataset of F-Words for the profanity filtering task. While creating the dataset, the participants should be selected considering the factors such as gender, age and accent. This will also help to reduce the model overfitting as well.

Working with audio data was very challenging considering the data availability, computation time, domain knowledge, package dependencies etc. In this project, a methodology for identifying and censoring F-words using speech audio characteristics have been proposed and tested successfully. This project also presents a novel dataset (F-Word audio dataset) for audio profanity identification. As for the author's knowledge, this is the first dataset focused on F-word utterances in English language.

By observing the censored long audio speech clips, it can be observed that for some F-words, the chime sound does not sync perfectly. The reason behind the issue was the difficulty to identify the exact starting point of the F-word within the one second audio clip. To address the issue, this study has focused on using an onset detection methodology to identify the exact starting time and end time of a word. A silence threshold setting of -30 dB and a minimum silence length setting of 300 milliseconds was used along with the Pydub detect_nonsilent package. However, the dataset did not perform well with the onset detections due to the recorded audio dataset was not generated focusing this special requirement. This also can be considered for future work to this project. Furthermore, this project can be extended to cover the full spectrum of profanity words and an advance implementation to the project would be predicting audio profanity in real-time.

List of References

- Bello, J. et al., 2005. A tutorial on onset detection in music signals. *IEEE Transactions on Speech and Audio Processing*, 13(5), pp. 1035-1047.
- Chaudhari, A., Davda, P., Dand , M. & Dholay, S., 2021. *Profanity Detection and Removal in Videos using Machine Learning*. s.l., 2021 6th International Conference on Inventive Computation Technologies (ICICT), pp. 572-576.
- Coyne, S., Stockdale, L., Nelson, D. & Fraser, A., 2011. Profanity in Media Associated With Attitudes and Behavior Regarding Profanity Use and Aggression. *Pediatrics*, Volume 128, pp. 867-72.
- Geeks of geeks, 2020. *Video to Audio convert using python*. [Online] Available at: <https://www.gee.org/video-to-audio-convert-using-python> [Accessed 10 March 2022].
- He, K., Zhang, X., Ren, S. & Sun, J., 2016. *Deep residual learning for image recognition*. s.l., Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition.
- Ho Park, J. & Fung, P., 2017. *One-step and Two-step Classification for Abusive Language Detection on Twitter*. s.l., 2017 Computing Research Repository (CoRR).
- Iyer, N., 2020. *Auto Censoring Swear Words in Fastaudio*. [Online] Available at: https://github.com/neelriyer/spiyer99.github.io/blob/master/_posts/2020-12-20-Auto-Censoring-Swear-Words-using-Fastaudio.md [Accessed 10 March 2022].
- Karpagavalli , S. & Chandra, E., 2016. A Review on Automatic Speech Recognition Architecture and Approaches. *Int. J. Signal Process. Image Process. Pattern Recognit*, 9(4), pp. 393-404.
- Krizhevsky, A., Sutskever, . I. & E. Hinton, G., 2012. *Imagenet classification with deep convolutional neural networks*. s.l., NIPS.
- Maryam Mohsin, 2021. *10 YouTube statistics every marketer should know in 2021*. [Online] Available at: www.ob.com/blog/youtube-statistics [Accessed 09 March 2022].
- Matthew Speiser, 2015. *The most popular curse words in America, according to twitter*. [Online] Available at: <https://www.businessinsider.com/most-popular-curse-words-in-america-2015-8> [Accessed 10 March 2022].
- Rajalingam, G. et al., 2021. *Automatic Audio Replacement of Objectionable Content for Sri Lankan Locale*. s.l., Lecture Notes on Data Engineering and Communications Technologies.
- Rawat, W. & Wang , . Z., 2017. Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review. *Neural Comput*, Volume 29, pp. 2352-2449.
- SeedScientific, 2021. *How Much Data is Created Every Day?*. [Online] Available at: <https://seedscientific./how-much-is-created-every-day> [Accessed 08 March 2022].

Simon, J., Helter, T. & White, R., 2021. *Impacts of the Covid-19 lockdown and relevant vulnerabilities on capability well-being, mental health and social support: an Austrian survey study*. s.l., BMC Public Health.

Simonyan, K. & Zisserman, A., 2015. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. s.l., 3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings.

Szegedy, C. et al., 2015. *Going deeper with convolutions*. s.l., Conference on Computer Vision and Pattern Recognition (CVPR).

Wazir, A. et al., 2020. *Spectrogram-Based Classification of Spoken Foul Language Using Deep CNN*. s.l., IEEE 22nd International Workshop on Multimedia Signal Processing (MMSP), pp. 1-6.

Wazir, A. et al., 2021. *Design and implementation of fast spoken foul language recognition with different end-to-end deep neural network architectures*. s.l., Sensors, pp. 1-17.

Zhang, H., McLoughlin, I. & Song, Y., 2015. *Robust sound event recognition using convolutional neural networks*. s.l., IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 559-563.

Appendix A












<input type="checkbox"/>	 0.1 Record Single Audio Files.ipynb
<input type="checkbox"/>	 0.2 Record Total Dataset.ipynb
<input type="checkbox"/>	 0.3 Creating chime sound.ipynb
<input type="checkbox"/>	 1 Audio preprocessing for dataset generation.ipynb
<input type="checkbox"/>	 2 Generating Metadata file and EDA.ipynb
<input type="checkbox"/>	 3 Extracting MFCC Features.ipynb
<input type="checkbox"/>	 4 Model Development.ipynb
<input type="checkbox"/>	 5 Prediction.ipynb
<input type="checkbox"/>	 6 Prediction on long-form audio sample.ipynb
<input type="checkbox"/>	 7 Onset Detection.ipynb
<input type="checkbox"/>	 8 Testing Real time fword detection.ipynb

Figure A.6.1 Pipeline for the F-Word detection



Figure A.6.2 Supportive custom functions used

```
In [1]: import sounddevice as sd
import glob
import os
from scipy.io.wavfile import write

In [2]: swear = ['fuck', 'fuckass', 'fuckbag', 'fuckboy', 'fuckbrain', 'fuckbutt', 'fuckbutten', 'fucked', 'fuckface', 'fuckhead',
'fuckhole', 'fucking', 'fucknut', 'fuckoff', 'fucks', 'fuckstick', 'fucktard', 'fuckup', 'fuckward', 'fuckwitt']

In [3]: def record_audio_all_and_save(save_path, array, fs = 16000, n_times=2, seconds = 1, swear_clean = 'clz1_'):
    userID = input("To start recording input your name: ")
    n_times = n_times*2
    for word in array:
        print(f'Recording for the word {word}\n')
        for i in range(n_times):
            myrecording = sd.rec(int(seconds * fs), samplerate=fs, channels=2)
            sd.wait()
            write(save_path + '/' + swear_clean + userID + '.' + word + str(i) + ".wav", fs, myrecording)
            input(f'Press enter to record next or to stop press ctrl + C ({i + 1}/{n_times}): ')
        print('\n')

In [4]: record_path = 'raw_data/raw_fwords'
isExist = os.path.exists(record_path)
if not isExist:
    os.makedirs(record_path)

In [5]: record_audio_all_and_save(record_path, swear, 16000, n_times=3, seconds = 1)

To start recording input your name: ran
Recording for the word fuck

Press enter to record next or to stop press ctrl + C (1/5):
Press enter to record next or to stop press ctrl + C (2/5):
Press enter to record next or to stop press ctrl + C (3/5):
Press enter to record next or to stop press ctrl + C (4/5):
Press enter to record next or to stop press ctrl + C (5/5):
```

Figure A.6.3 Recording the audio datasets

preprocessing fword audio data set

```
In [4]: fwords_dir      = 'raw_data/SwearWord_WAV/'  
changed_sr_fwords = 'raw_data/SwearWord_16000_WAV/'  
rem_silence_path  = 'raw_data/SwearWordSR_16000_WAV/'  
overlayed_path    = 'raw_data/SwearWordOL_16000_WAV/'  
background_file   = 'onsec_16hz_background.wav'
```

```
In [5]: preprocessing_audio(fwords_dir, changed_sr_fwords, rem_silence_path, overlayed_path, background_file, 16000)  
  
preprocessing begins.....  
  
frequency details:  
unique values are: dict_keys([24000, 11025])  
unique values counts are: dict_values([338, 1])  
  
audio length details:  
unique values are: dict_keys([0.816, 0.792, 0.744, 0.5091156462585034, 1.176, 1.128, 0.984, 1.008, 0.96, 0.864, 0.84, 1.032,  
1.104, 0.888, 1.2, 1.056, 1.248, 0.936, 0.912, 1.152, 1.08])  
unique values counts are: dict_values([22, 9, 19, 1, 14, 8, 34, 33, 17, 17, 8, 2, 22, 32, 14, 27, 2, 37, 10, 4, 7])  
  
converting the audio files to the 16000 HZ sampling rate.....  
  
converting completed. files saved to raw_data/SwearWord_16000_WAV/  
  
starting silence removal.....  
  
silence removal done ! files saved in raw_data/SwearWordSR_16000_WAV/  
  
starting audio overlay.....  
  
overlaying completed ! files saved to raw_data/SwearWordOL_16000_WAV/  
  
preprocessing completed !!!
```

Figure A.6.4 Pre-processing of the speech audio clips

<input type="checkbox"/>	📁 CleanWords_WAV
<input type="checkbox"/>	📁 CleanWordsSample_16000_WAV
<input type="checkbox"/>	📁 CleanWordsSample_WAV
<input type="checkbox"/>	📁 CleanWordsSampleOL_16000_WAV
<input type="checkbox"/>	📁 CleanWordsSampleSR_16000_WAV
<input type="checkbox"/>	📁 combined_ds
<input type="checkbox"/>	📁 RecCleanWord_16000_WAV
<input type="checkbox"/>	📁 RecCleanWordOL_16000_WAV
<input type="checkbox"/>	📁 RecCleanWords_WAV
<input type="checkbox"/>	📁 RecCleanWordSR_16000_WAV
<input type="checkbox"/>	📁 record_10sec
<input type="checkbox"/>	📁 Recorded_Audio
<input type="checkbox"/>	📁 RecSwearWord_16000_WAV
<input type="checkbox"/>	📁 RecSwearWordOL_16000_WAV
<input type="checkbox"/>	📁 RecSwearWords_WAV
<input type="checkbox"/>	📁 RecSwearWordSR_16000_WAV
<input type="checkbox"/>	📁 SwearWord_16000_WAV
<input type="checkbox"/>	📁 SwearWord_WAV
<input type="checkbox"/>	📁 SwearWordOL_16000_WAV
<input type="checkbox"/>	📁 SwearWordSR_16000_WAV
<input type="checkbox"/>	📁 Validation_WAV
<input type="checkbox"/>	📄 audio_data.csv

Figure A.6.5 Pre-processed outputs

```
In [7]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
In [8]: print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
```

```
(4316, 40) (1080, 40) (4316, 2) (1080, 2)
```

```
In [9]: model = Sequential([
    Dense(256, input_shape=X_train[0].shape),
    Activation('relu'),
    Dropout(0.5),
    Dense(256),
    Activation('relu'),
    Dropout(0.5),
    Dense(2, activation='softmax')
])
```

Figure A.6.6 RNN Model development

```
In [71]: def _splitter(i, file, folder, increment):
    t1 = i
    t2 = i + increment
    output = Path(f'{folder}/audio_{i}{Path(file).suffix}')
    audio = AudioSegment.from_wav(file)
    audio[t1:t2].export(output, format=f'{Path(file).suffix[1:]}')
```

```
In [72]: folder = 'testing_split'
    file = long_audio
```

```
In [87]: _splitter(1198, file, folder, 466)
```

```
In [152]: from pydub import AudioSegment
    from pydub.silence import split_on_silence

    sound_file = AudioSegment.from_wav(long_audio)
    audio_chunks = split_on_silence(sound_file,
        # must be silent for at least half a second
        min_silence_len=300,

        # consider it silent if quieter than -50 dBFS
        silence_thresh=-30
    )
```

```
In [153]: for i, chunk in enumerate(audio_chunks):
    out_file = "./testing_split/chunk{0}.wav".format(i)
    print("exporting", out_file)
    chunk.export(out_file, format="wav")
```

```
exporting ./testing_split/chunk0.wav
exporting ./testing_split/chunk1.wav
```

Figure A.6.7 Onset detection using silence thresholds

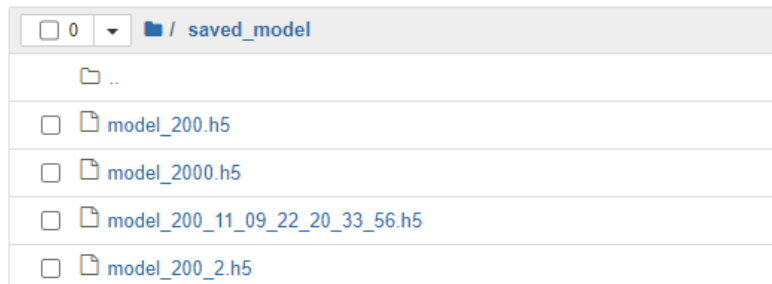


Figure A.6.8 Saved RNN models



Figure A.6.9 Long audio file split to one second clips

```
[ ] train_spectrogram_ds = train_spectrogram_ds.cache().shuffle(10000).prefetch(tf.data.AUTOTUNE)
    val_spectrogram_ds = val_spectrogram_ds.cache().prefetch(tf.data.AUTOTUNE)
    test_spectrogram_ds = test_spectrogram_ds.cache().prefetch(tf.data.AUTOTUNE)

[ ] input_shape = example_spectrograms.shape[1:]
    print('Input shape:', input_shape)
    num_labels = len(commands)

    # Instantiate the `tf.keras.layers.Normalization` layer.
    norm_layer = layers.Normalization()
    # Fit the state of the layer to the spectrograms
    # with `Normalization.adapt`.
    norm_layer.adapt(data=train_spectrogram_ds.map(map_func=lambda spec, label: spec))

    model = models.Sequential([
        layers.Input(shape=input_shape),
        # Downsample the input.
        layers.Resizing(32, 32),
        # Normalize.
        norm_layer,
        layers.Conv2D(32, 3, activation='relu'),
        layers.Conv2D(64, 3, activation='relu'),
        layers.MaxPooling2D(),
        layers.Dropout(0.25),
        layers.Flatten(),
        layers.Dense(128, activation='relu'),
        layers.Dropout(0.5),
        layers.Dense(num_labels),
    ])

    model.summary()
```

Figure A.6.10 CNN model development

```
[ ] x = '/content/drive/MyDrive/Research_Nov_15/mono_converted2/clean_words/clz0_channaka_right3.wav'
x = tf.io.read_file(str(x))
x, sample_rate = tf.audio.decode_wav(x, desired_channels=1, desired_samples=16000,)
x = tf.squeeze(x, axis=-1)
waveform = x
x = get_spectrogram(x)
x = x[tf.newaxis,...]

prediction = model(x)
plt.bar(commands, tf.nn.softmax(prediction[0]))
plt.title('No')
plt.show()

display.display(display.Audio(waveform, rate=16000))
```

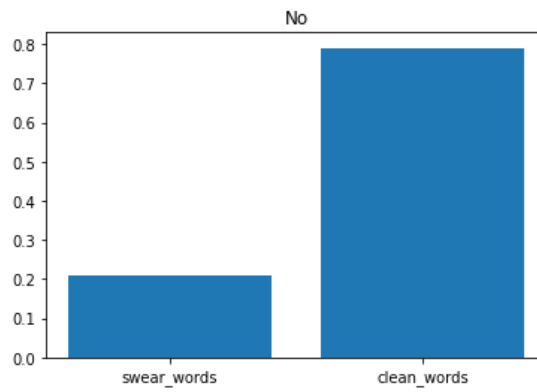


Figure A.6.11 CNN model prediction for a clean word