Development of Keyword Spotting System to detect Spam Calls

A Summer Internship Report

Submitted by:

Shubham Kumar (1607020)
Gaurav Upadhyay (1607031)
Ayush Gupta (1607040)

of

Information Technology

Under the Supervision of:

Dr. Jyoti Prakash Singh
HOD, Department of CSE

National Institute of Technology, Patna

National Institute of Technology Patna

Department of Computer Science and Engineering

May-July, 2019

NATIONAL INSTITUTE OF TECHNOLOGY PATNA

(An Institute under Ministry of HRD, Govt. of India)

ASHOK RAJPATH, PATNA-800005 (Bihar)

CERTIFICATE

This is to certify that the dissertation entitled **"Development of Keyword Spotting System to detect spam calls"** submitted by **Shubham Kumar [1607020], Gaurav Upadhyay [1607031], and Ayush Gupta [1607040]** as their Summer Internship 2019 project in partial fulfillment of the requirements of the degree of Bachelor of Technology in Computer Science and Engineering at the National Institute of Technology Patna is an authentic work carried out by them under my supervision.

(Signature)
Dr. Jyoti Prakash Singh
HOD, Department of CSE, NIT PATNA

राष्ट्रीय प्रौद्योगिकी संस्थान पटना
NATIONAL INSTITUTE OF TECHNOLOGY PATNA

Declaration

We hereby declare that the project work entitled "**Development of Keyword Spotting System to detect spam calls**" is an authentic record of our own work carried out at National Institute of Technology Patna as a requirement for Summer Internship the award of degree of Bachelor of Technology in Computer Science and Engineering at National Institute of Technology Patna, under the supervision of **Dr. Jyoti Prakash Singh,** HOD, Department of Computer Science and Engineering, National Institute of Technology Patna.

Shubham Kumar (1607020)

Gaurav Upadhyay (1607031)

Ayush Gupta (1607040)

Date:
Place: Patna

Acknowledgement

We hereby take the privilege to express our gratitude to all the people who were directly or indirectly involved in the execution of this work, without whom this project would not have been a success.

We extend our deep gratitude, respect, and obligation to our project supervisor, **Dr. Jyoti Prakash Singh**, HOD, Department of CSE, National Institute of Technology, Patna for his timely suggestions and encouragement.

Our heartiest thanks to Karabi Maity, PhD Scholar, NIT Patna for his valuable support and guidance. We would also like to thank our institution and the faculty members without whom this project would have been a distant reality.

Shubham Kumar (1607020)
Gaurav Upadhyay (1607031)
Ayush Gupta (1607040)

Date:
Place: Patna

Index

1. Abstract

In general terms, keyword spotting (KWS) is the task of detecting specific words/phrases of interest within a continuous speech with the aid of machines. KWS is noted to be useful in a variety of applications such as telephone call center systems, surveillance applications, audio retrieval etc. Even though keyword spotting is very much similar to the task of speech transcription i.e., automatic speech recognition (ASR), the processing requirements of KWS are significantly less in comparison to the ASR. Consequently, KWS can run at considerably faster speeds. Real-time monitoring of calls in the Indian context is a very challenging task.This is so because, in general, the conversations over the mobile/landline phone happen in the local language along with words from the English language. In India, a very large number of local languages are spoken. In addition to that, each language has a number of dialects as well.Consequently, such call
monitoring systems should be multilingual. In other words, the KWS module should be capable of detecting specific words/phrases/sentences from all the spoken languages simultaneously. This problem can be simplified up to an extent by developing KWS modules that are designed to cater to one particular local language(Hindi) along with English, i.e.,a bilingual KWS.

In this project, our system takes speech signal as input and predicts the if its spam or not as output, based on certain set of keywords. First the speech signal goes through the feature extractor and subsequently the text generated is fed into a classifier which predicts the speech is fraudulent or not. The feature extractor we have used is Mel-frequency Cepstral Coefficient (MFCC) which gives a corresponding feature vector that is given to a Deep Neural Network (DNN).

2. Objective

Our main objective is to detect whether the input speech signal is spam or not. For doing this task we divide our work into two parts, the first part is automatic speech recognition which gives us generated text from corresponding speech, and in the second part we classify the text generated from part one using keyword spotting.

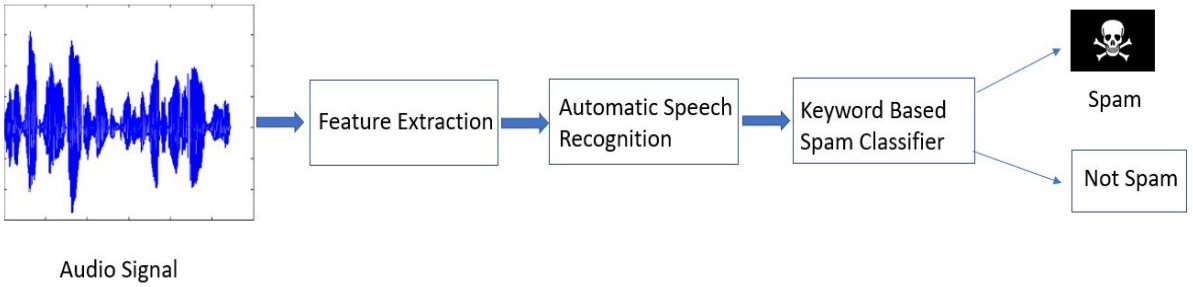

Fig i. Objective

3. Speech Signal

The speech signal, as it emerges from a speaker's vocal tract, is a one-dimensional function (air pressure) of time. The input to the system is a speech signal. The speech signal, as it emerges from a speaker's vocal tract, is a one-dimensional function (air pressure) of time. It has to be sampled and quantized before doing any further work. A signal is **sampled** by measuring its amplitude at a particular time. Amplitude measurements are **quantized** and are stored as either as 8-bit or 16-bit number. The quantized waveform is used for extracting MFCC features.

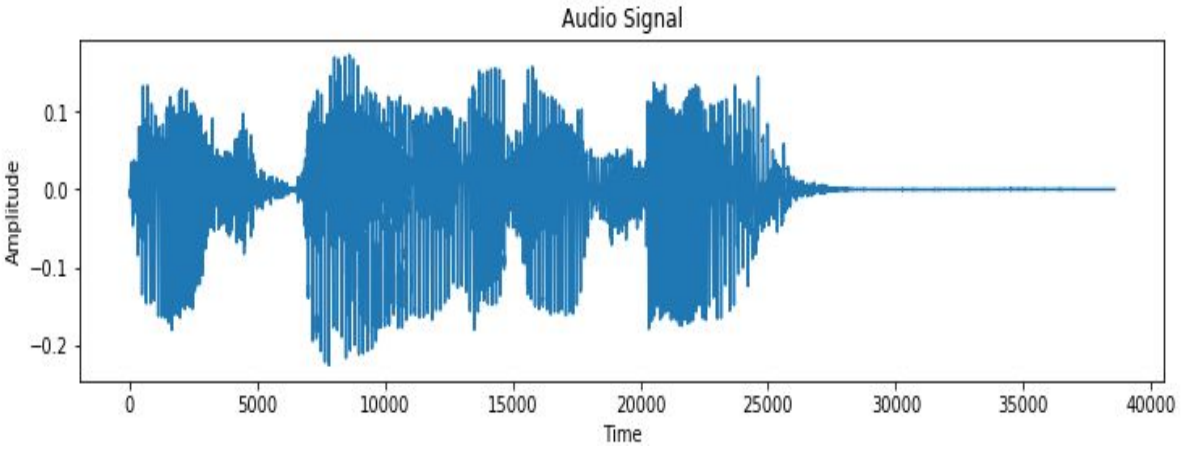

Fig ii. Audio Signal

4. Proposed Approach

Speech Recognition part is divided into three steps:

i. Feature Extraction

ii. Acoustic Model

iii. Decoder

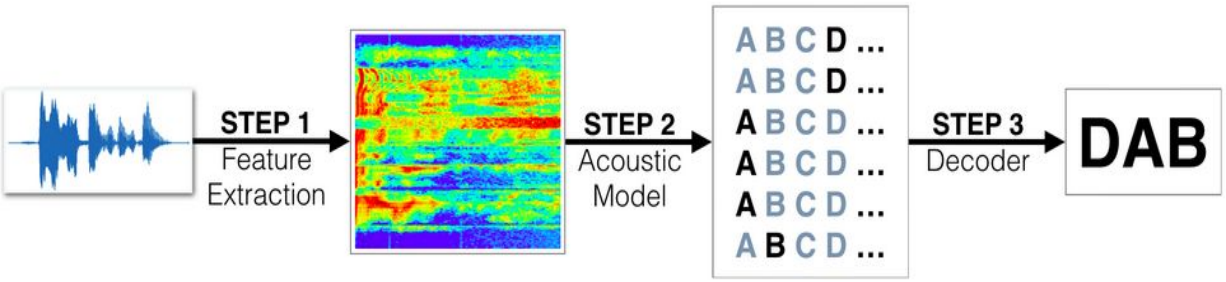

Fig iii. Proposed Approach

4.1 Feature Extraction:

Feature Extraction - It is a data pre-processing step that converts raw audio to feature representations that are commonly used for ASR. In our case for feature extraction we use MFCC(Mel-Frequency Cepstral Coefficients).

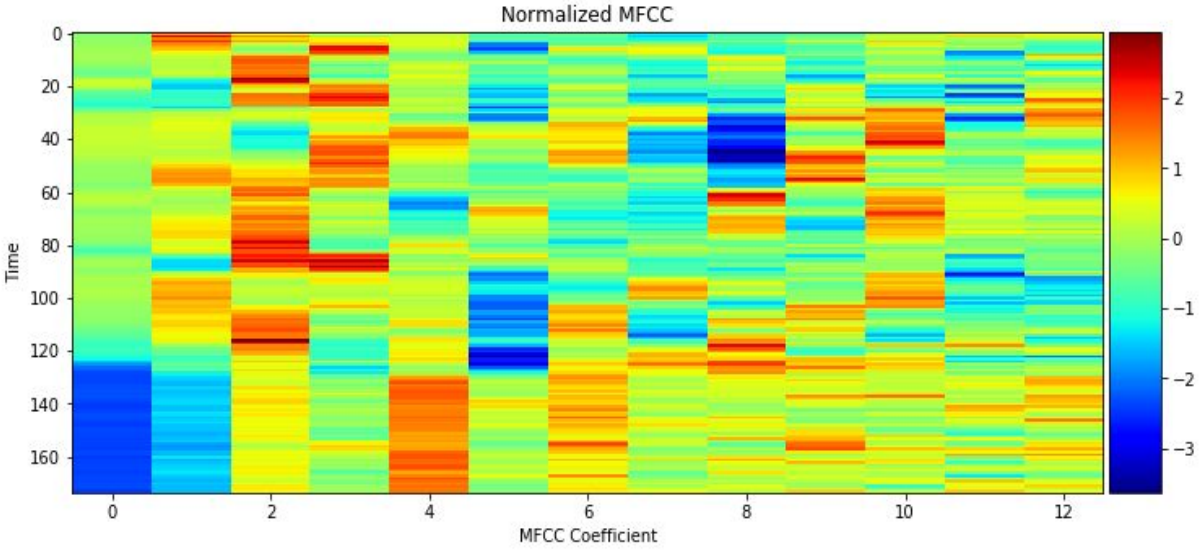

Fig iv. MFCC Coefficient

MFCC are coefficients that collectively make up an MFC(Mel-frequency cepstrum). They are derived from a type of cepstral representation of the audio clip (a nonlinear "spectrum-of-a-spectrum"). The difference between the cepstrum and the mel-frequency cepstrum is that in the MFC, the frequency bands are equally spaced on

the mel scale, which approximates the human auditory system's response more closely than the linearly-spaced frequency bands used in the normal cepstrum. This frequency warping can allow for better representation of sound.

4.2 Acoustic Model :

Acoustic Model - It accepts audio features as input and returns a probability distribution over all potential transcriptions. We use different types of neural networks to work on acoustic modeling.

Fig v. Acoustic Model

In our case we have used following neural networks:

i. Simple RNN model

ii. RNN + TimeDistributed Dense

iii. Bidirectional RNN + TimeDistributed Dense

iv. Deep bidirectional RNN

4.3 Decoder:

In this part we got predicted transcription from output of acoustic modeling.

Fig vi. Decoder

5. Dataset :

- Dataset is of NIT Patna : All the speakers are from NIT Patna campus with different educational backgrounds (schooling and college), language accents, Further they are from different parts of India. This was done to include the dialect variabilities in the collected speech data. The speech data was recorded using mobile phone of student volunteers. In dataset collection process total 377 students participated.
- Our dataset is divided into two parts training and testing.
- Training part consists 3761 audio file(.wav format) and its corresponding transcription.
- Testing part consists of 771 audio file(.wav format) and its corresponding transcription.
- eg :- “apke pass kaunse bank ka atm card h”.

6. Methodology

6.1.Automatic Speech Recognition

6.1.1.Python Based

i) Model 0 : Simple RNN model

Since the data is sequential we tried using simple RNN as the first Acoustic Model. At each time step, the speaker pronounces one of 28 possible characters, including each of the 26 letters in the English alphabet, along with a space character (" "), and an apostrophe (').

The output of RNN at each time step is a vector of probabilities with 29 entries.

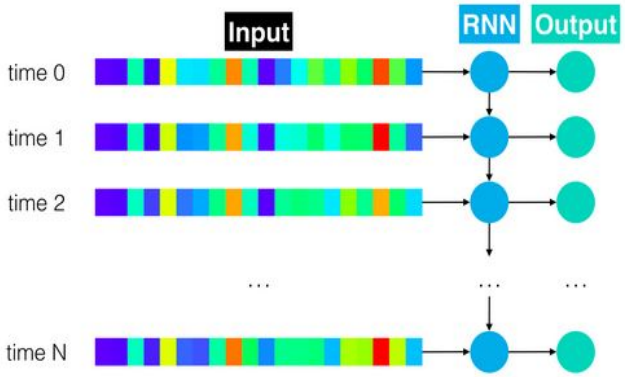

Fig vii. Simple RNN Model

Key Features:

Input: MFCC features vector of dimension 13

Number of GRU unit : 29

Activation Function: Softmax

Minibatch Size: 20

Optimizer: SGD with learning rate 0.02

No of Epochs: 20

ii) Model 1 : RNN + TimeDistributed Dense

The first layer of the neural network should be an RNN (SimpleRNN, LSTM, or GRU) that takes the time sequence of audio features as input. Whereas the architecture in simple_rnn_model treated the RNN output as the final layer of the model, you will use the output of your RNN as a hidden layer. Use TimeDistributed to apply a Dense layer to each of the time steps in the RNN output. Ensure that each Dense layer has o/p_units.

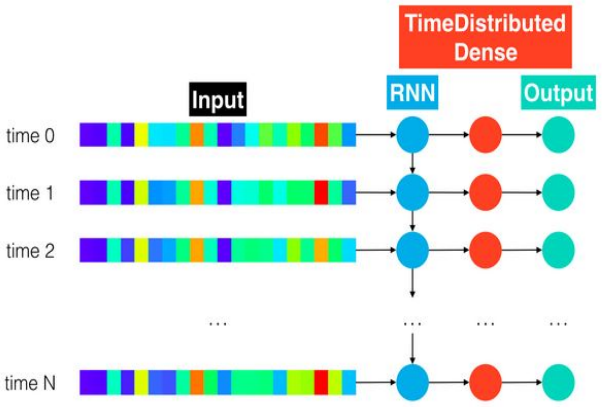

Fig viii. RNN + TimeDistributed Dense

Key Features:

Input: MFCC features vector of dimension 13

Number of GRU unit : 200

No of Dense unit : 29

Activation Function: Softmax

Minibatch Size: 20

Optimizer: SGD with learning rate 0.02

No of Epochs: 20

iii) Model 2 : Bidirectional RNN + TimeDistributed Dense

Bidirectional recurrent neural networks(RNN) are really just putting two independent RNNs together. The input sequence is fed in normal time order for one network, and in reverse time order for another. The outputs of the two networks are usually concatenated at each time step

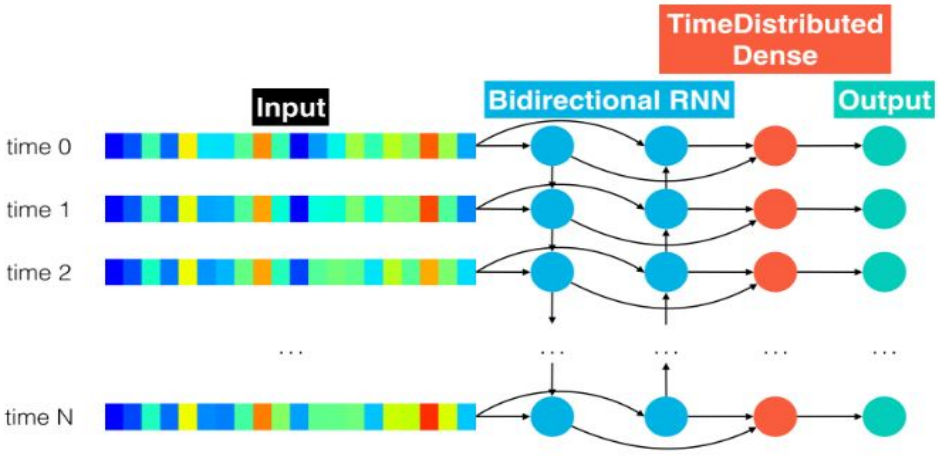

Fig xi. Bidirectional RNN + TimeDistributed Dense

Key Features:

Input: MFCC features vector of dimension 13

Number of bidirectional GRU unit : 400

No of Dense unit : 29

Activation Function: Softmax

Minibatch Size: 20

Optimizer: SGD with learning rate 0.02

No of Epochs: 10

iv) Model 3 : Deep bidirectional RNN

Bidirectional recurrent neural networks(RNN) are really just putting two independent RNNs together. The input sequence is fed in normal time order for one network, and in reverse time order for another. The outputs of the two networks are usually concatenated at each time step. Here we put a stack of Bidirectional RNN.

Key Features:

Input: MFCC features vector of dimension 13

Number of bidirectional GRU unit : 400

Number of bidirectional layer: 2

No of Dense unit : 29

Activation Function: Softmax

Minibatch Size: 20

Optimizer: SGD with learning rate 0.02

No of Epochs: 10

6.1.2 Kaldi Toolkit Based

Kaldi is an open source toolkit made for dealing with speech data. it's being used in voice-related applications mostly for speech Recognition but also for other tasks — like speaker diarisation. It is written mainly in C/C++, but the toolkit is wrapped with Bash and Python scripts. For basic usage this wrapping spares the need to get in too deep in the source code.

There are four parts of kaldi :

i. Preprocessing and Feature Extraction

We mostly deals with two things:

1. Identifying the sound of human speech
2. Discarding any unnecessary noise.

In kaldi feature extraction is done using MFCC(*Mel-Frequency Cepstral Coefficients)* . And other than MFCC their are also two other features:

1. CMVNs: It is used for better normalization of MFCC. MFCC and CMVN are used for representing the content of each audio utterance.
2. I-Vectors: They are used for representing the style of each audio utterance or speaker.

In pre-processing the speech data was first pre-emphasized using a filter with pre-emphasis factor being 0.97. Overlapping Hamming windows that were 20 ms in duration with an overlap of 10 ms were used for creating short-time frames. A 40 channel Mel-filterbank was used to extract the 13-dimensional base MFCC features which were then spliced in time considering a context size of 9 frames. Next, using linear discriminant analysis and maximum likelihood linear transform 40-dimensional vectors were derived.

ii. Defining Model

Kaldi's model can be divided into two main components:

1. Acoustic Model: It is based on Deep Neural Network and in this part that model will transcribe the audio features that we created into some sequence of context-dependent phonemes.
2. Decoding Graph: In this part phonemes are used as an input and turns them into lattices. A lattice is a representation of the alternative word-sequences that are likely for a particular audio part. This is generally the output that you want to get in a speech recognition system. The decoding graph takes into account the grammar of your data, as well as the distribution and probabilities of contiguous specific words (n-grams).

iii. The Training Part

Steps performed for training:

1. This is the trickiest part. In Kaldi we'll need to order our transcribed audio data in a specific order.
2. After ordering our data, we'll need a representation of each word to the phonemes that create them. for eg. eight -> ey t, five -> f ay v .
3. Next step is to align the phonemes into the audio sound with GMM.
4. After the alignment we will create the DNN that will form the acoustic Model, and we will train it to match the alignment output. After creating the acoustic model we can train the WFST to transform the DNN output into the desired lattices.
5. For training we keep total epochs equal to 20 in which first 15 epochs learning rate is reducing. Where in next 5 epochs learning rate is set fixed. Initial learning rate is set to 0.04 where final learning rate is 0.004. Number of hidden layer is set to 3 where each hidden layer consist of 1024 dimensions, sample per iteration is equal to 200000 and min batch size equal to 256.

iv. Decoding Part

In this part we feed audio signal to the trained model and its gives transcripted text of audio signal as an output.

6.2. Keyword Spotting Based Spam Classifier

In this part we put output of ASR (transcripted text of call) as an input in a python script which keep count of spam keywords like(atm, pin, card, etc). We define some threshold if count of spam keyword are greater than threshold then we classify that audio as spam otherwise it is non spam.

7. Implementation

7.1. Libraries Used:

```
import keras
import numpy as np
import tensorflow as tf
from data_generator import AudioGenerator
from keras import backend as K
from keras.backend.tensorflow_backend import set_session
from keras import backend as K
from data_generator import plot_mfcc_feature
from data_generator import plot_spectrogram_feature
from data_generator import vis_train_features
from keras.models import Model
from keras.models import Model
from utils import int_sequence_to_text
from IPython.display import Audio
from IPython.display import Markdown, display
from keras.layers import (BatchNormalization, Conv1D, Dense, Input, TimeDistributed, Activation, Bidirectional,
                          SimpleRNN, GRU, LSTM, Dropout, MaxPooling1D)
```

7.2. Models Used :

i) Model 0: RNN :

```
def simple_rnn_model(input_dim, output_dim=29):
    input_data = Input(name='the_input', shape=(None, input_dim)) # Add recurrent layer
    simp_rnn = GRU(output_dim, return_sequences=True, implementation=2, name='rnn')(input_data)# Add softmax activation layer
    y_pred = Activation('softmax', name='softmax')(simp_rnn)# Specify the model
    model = Model(inputs=input_data, outputs=y_pred)
    model.output_length = lambda x: x
    print(model.summary())
    return model
```

ii) Model 1 : RNN + TimeDistributed Dense

```
def rnn_model(input_dim, units, activation, output_dim=29):
    input_data = Input(name='the_input', shape=(None, input_dim))
    gru_layer=GRU(units, activation=activation,return_sequences=True,implementation=2, name='rnn')(input_data)#Add recurrent laye
    bn_gru_layer = BatchNormalization(name = 'bn_gru_layer')(gru_layer)  # Add batch normalization
    time_dense = TimeDistributed(Dense(output_dim))(bn_gru_layer)   # Add a TimeDistributed(Dense(output_dim)) layer
    y_pred = Activation('softmax')(time_dense)  # Add softmax activation layer
    model = Model(inputs=input_data, outputs=y_pred)  # Specify the model
    model.output_length = lambda x: x
    print(model.summary())
    return model
```

iii) Model 2: Bidirectional RNN + TimeDistributed Dense

```
def bidirectional_rnn_model(input_dim, units, output_dim=29):
    input_data = Input(name='the_input', shape=(None, input_dim)) # TODO: Add bidirectional recurrent layer
    bidir_rnn = Bidirectional(GRU(units, activation='relu',return_sequences=True, implementation=2, name='bidir_rnn'))(input_data
    time_dense = TimeDistributed(Dense(output_dim))(bidir_rnn)# Add a TimeDistributed(Dense(output_dim)) layer
    y_pred = Activation('softmax', name='softmax')(time_dense)# Add softmax activation layer
    model = Model(inputs=input_data, outputs=y_pred) # Specify the model
    model.output_length = lambda x: x
    print(model.summary())
    return model
```

iv) Model 3: Deep bidirectional RNN

```
def final_model(input_dim, units, output_dim=29):
    input_data = Input(name='the_input', shape=(None, input_dim))
    # ============== 1st bidirectional recurrent Layer ============== #
    bidirectional_rnn = Bidirectional(GRU(units, activation=None,return_sequences=True, implementation=2, name='bidir_rnn'))
                        (input_data)
    batch_normalization = BatchNormalization(name = "batch_normalization_bidirectional_rnn")(bidirectional_rnn)
    activation = Activation('relu')(batch_normalization)  # Add activation function
    # ============== 2nd  bidirectional recurrent Layer ============== #
    bidirectional_rnn = Bidirectional(GRU(units, activation=None,return_sequences=True, implementation=2, name='bidir_rnn'))
                        (activation)
    batch_normalization = BatchNormalization(name = "bn_bidir_rnn_2")(bidirectional_rnn)
    activation = Activation('relu')(batch_normalization)  # Add activation function
    # ============== 3rd Layer TimeDistributed(Dense(output_dim)) layer ============== #
    time_dense = TimeDistributed(Dense(output_dim))(activation)
    y_pred = Activation('softmax', name='softmax')(time_dense)  # Add softmax activation layer
    model = Model(inputs=input_data, outputs=y_pred)
    model.output_length = lambda x: x
    print(model.summary())
    return model
```

7.3. Training

```
train_model(input_to_softmax=model_1,     #change to model_(0,1,2,3) for corresponding models
            pickle_path='model_1.pickle',
            save_model_path='model_1.h5',
            spectrogram=False, epochs=10) # change to False if you would like to use MFCC features
```

7.4. Text Prediction

```
get_predictions(index=0,  #change partition = (train, validation, test)
                partition='train', #change model path to according to model
                input_to_softmax=simple_rnn_model(input_dim=13),
                model_path='./results/model_0.h5')
```

7.5. Spam Detection using KeyWord Spotting

```
class SpamCheck:
    SpamDict = {
        'atm' : 0, 'block' : 0, 'unblock' : 0, 'mobile' : 0, 'verify' : 0, 'card' : 0, 'cvv' : 0, 'expiry' : 0,
         'otp' : 0, 'pin' : 0, 'validity': 0, 'number' : 0, 'month' : 0, 'year' : 0, 'date' : 0, 'bank' : 0
        }
    def __init__ (self, text):
        self.text = text


    def generateList(self):
        self.wordlist = list(self.text.split(' '))


    def generateDict(self):
        self.generateList()
        for word in self.wordlist:
            try:
                if self.SpamDict[word] == 0:
                    self.SpamDict[word] = 1
            except KeyError:
                pass

    def checkSpam(self):

        self.generateDict()
        self.count = 0
        for word in self.SpamDict:
            if self.SpamDict[word] == 1:
                self.count += 1
                if self.count > 3:                                          #Here we have taken 3 as threshold
                        return 'Off the phone call, It is spam call'            #then it is spam call

        return 'Continue Phone call'
```

8. Results

8.1. Using Python Based Approach

In this approach we did not get expected results.

i) For training data set we got the following result :

```
--------------------------------------------------------------------------------
True transcription:

which plan did you choose
--------------------------------------------------------------------------------
Predicted transcription:

which plan did no choose
--------------------------------------------------------------------------------
```

ii) For testing data we got the following result:

```
--------------------------------------------------------------------------------
True transcription:

aapke atm card ki expiry date kya hai month and year batayiye
--------------------------------------------------------------------------------
Predicted transcription:

aaedan kar chiye haitet kyahe fone td ye re a
--------------------------------------------------------------------------------
```

Model	Training Loss	Validation Loss
Simple RNN	547.1277	560.7871
RNN + TimeDistributed Dense	135.2153	167.0604
Bidirectional RNN + TimeDistributed Dense	96.1184	175.6786
Deep Bidirectional RNN	74.2804	167.3498

Table i. Result using Python Based Approach

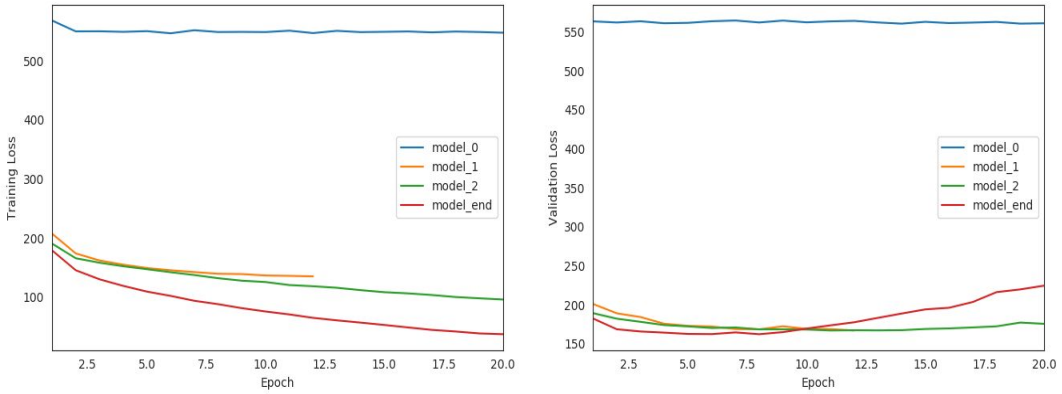

Fig x. Loss Curve of Training and Validation Data vs Epoch

Possible Reasons :

1. This approach does not use phonemes.
2. The data set has both dialects and bilingual variabilities.
3. Background noise.

8.2. Using Kaldi-Based Approach

Model	Hidden Layer Dimension	Batch Size	Word Error Rate(WER)
DNN	256	32	21.98
DNN	1024	256	17.31

Table ii. Result Using Kaldi Based Approach

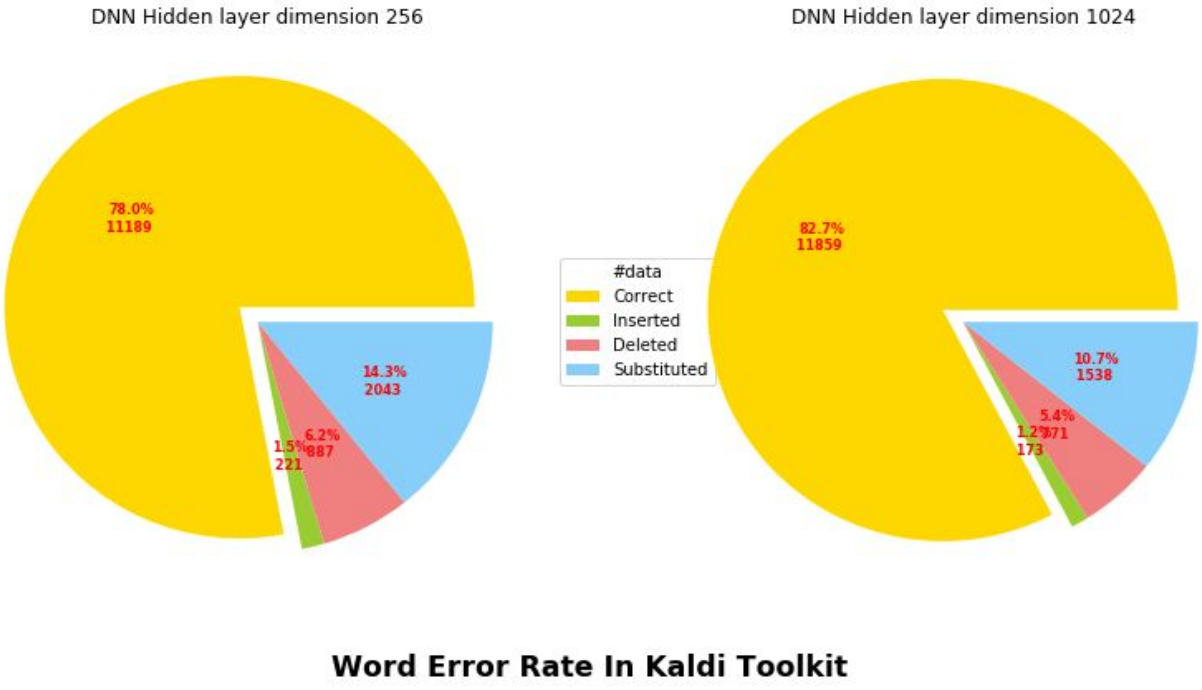

Fig xii. Result Comparison of Models in Kaldi

9. Conclusion

Keyword Spotting System to detect the Spam calls using Speech Signal is implemented by us takes the audio waveform as the input and predicts the corresponding text which subsequently is run over a python script and detects whether the call is malicious/spam or not. For text prediction the audio is fed into the neural networks.

The models used in the python based approach :

i. Simple RNN model
ii. RNN + TimeDistributed Dense
iii. Bidirectional RNN + TimeDistributed Dense
iv. Deep bidirectional RNN

We have also used the Kaldi based approach, a prototype KWS system is developed using DNN-HMM acoustic modeling using MFCC as the feature vectors. The efficiency of the system is studied using the bilingual speech data keywords. The keywords are selected based on their possible occurrence in different Fraud/Spam calls targeted in this work. A fraud/Spam call detection system is developed employing bilingual keyword search. The Kaldi - toolkit was used for learning the HMM parameters for the phoneme models. At the same time, both GMM and DNN-based acoustic modelling were explored for capturing the observation probabilities. In accordance to the goal of the project we further aim to increase the datafile and make the database available to the research community to promote efforts in this area.

It is implemented in Python because of its vast libraries and communities that it provides.The motivation behind such system arose due to rapid increase in the number of spam or fraudulent calls in the past few years. Fraud calls asking for details like password/PIN for the bank, lucrative job calls are also becoming rampant. A keyword spotting system is capable of detection of such fraud calls and provides authentic solution to the problem.

10. References

[1] H. Sakoe and S. Chiba, “A dynamic time warping approach to continuous speech recognition,” in *Proc. Seventh International Congress on Acoustics*, 1971.

[2] R. C. Rose and D. B. Paul, “A hidden markov model based keyword recognition system,” in *Proc. ICASSP*, April 1990, pp. 129–132.

[3] G. Chen, C. Parada, and G. Heigold, “Small-footprint keyword spotting using deep neural networks,” in *Proc. ICASSP*, 2014, pp. 4087–4091.

[4] T. N. Sainath and C. Parada, “Convolutional neural networks for small-footprint keyword spotting,” in *Proc. INTERSPEECH*, 2015, pp. 1478–1482.

[5] H. Hermansky, “Perceptual linear predictive (PLP) analysis of speech,” *The Journal of the Acoustical Society of America*, vol. 57, no. 4, pp. 1738–1752, April 1990.

[6] S. Davis and P. Mermelstein, “Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences,” *IEEE Transactions on Acoustic, Speech and Signal Processing*, vol. 28, no. 4, pp. 357–366, August 1980.

[7] A. Graves, A. R. Mohamed, and G. Hinton, “Speech recognition with deep recurrent neural net- works,” in *Proc.International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2013, pp. 6645–6649.

[8] F. Jelinek, “Continuous speech recognition by statistical methods,” *Proceedings of the IEEE*, vol. 64, no. 4, pp. 532–556, April 1976.

[9] L. R. Rabiner, “A tutorial on hidden Markov models and selected applications in speech recognition,” *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, February 1989.

[10] A. P. Dempster, N. M. Laird, and D. B. Rubin, “Maximum likelihood from incomplete data via the em algorithm,” *Journal of the Royal Statistical Society, Series B*, vol. 39, no. 1, pp. 1–38, 1977.