MULTIMODAL SARCASM DETECTION

Report submitted in partial fulfillment of the requirements for the B. Tech. degree in Computer Science & Engineering

By

NAME OF THE STUDENT	Roll No.
1. Zeel Prajapati	17BCP042
2. Prerak Shah	17BCP044

Under the supervision
Of
Dr. Santosh Kumar Bharti

SCHOOL OF TECHNOLOGY
PANDIT DEENDAYAL ENERGY UNIVERSITY
GANDHINAGAR, GUJARAT, INDIA

May-June-2021

MULTIMODAL SARCASM DETECTION

Report submitted in partial fulfillment of the requirements for the B. Tech. degree in Computer Science & Engineering

By

NAME OF THE STUDENT	Roll No.
1. Zeel Prajapati	17BCP042
2. Prerak Shah	17BCP044

Under the supervision
Of
Dr. Santosh Kumar Bharti

SCHOOL OF TECHNOLOGY
PANDIT DEENDAYAL ENERGY UNIVERSITY
GANDHINAGAR, GUJARAT, INDIA

May-June-2021

Student Declaration

I, the undersigned, hereby declare that the project report entitled Multimodal Sarcasm Detection submitted by me to the Department of Computer Science and Engineering, School of Technology, Pandit Deendayal Petroleum University, in partial fulfilment of the requirement for the award of the degree of B. Tech in Computer Engineering is a record of bona fide project work carried out by me, under the guidance of Dr. Santosh Kumar Bharti. I further declare that the work reported in this record has not been submitted, and will not be submitted, either in part or full, for the award of any other degree/diploma in this Institute or any other Institute or University.

Date: Name and Sign of the Candidate

Place:

CERTIFICATE

This is to certify that the report on "Multimodal Sarcasm Detection" submitted by the students, as a requirement for the degree in Bachelor of Technology (B. Tech) in Computer Science & Engineering, under my guidance and supervision for the session 2019-2020.

Name of the student	**Roll No.**	**Signature**
1. Zeel Prajapati	17BCP042	
2. Prerak Shah	17BCP044	

Date:

Place:

Signature of the Supervisor

(Dr. Santosh Kumar Bharti)

PREFACE

For the duration of five months from 18th January, 2021 to 10th May, 2021 we did a comprehensive project on the topic Multimodal Sarcasm Detection. This comprehensive project is a part of our B.Tech. in Computer Engineering at Pandit Deendayal Energy University. We were fortunate enough to receive guidance from mentor Dr. Santosh Kumar Bharti (Academic Mentor) who helped us complete this internship with ease with their instructions and feedback.

The main objective of our project was to detect sarcasm from text as well as audio. Until now, most of the work was done to detect sarcasm from text. Very few researchers had tried to detect sarcasm from audio. As a result, we decided to focus more on the audio cues along with the traditional textual cues. From the results it was noted that audio and text both play an important role in detecting sarcasm. Audio and textual cues compensate each other's shortcomings and as a result we were able to detect sarcasm effectively.

ACKNOWLEDGEMENT

We would like to thank our Faculty Advisor, Dr. Santosh Kumar Bharti for the valuable advice, guidance, encouragement, and technical support provided during the monthly evaluations and otherwise.

We would like to express our deep gratitude to the HOD - Department of Computer Engineering, Dr. Samir Patel who provided us the platform to pursue the opportunity to work on the final semester project. Further, we would also like to take this opportunity to thank all the faculty members and staff at the University for their support and help.

Name of Student Signature

1. Zeel Prajapati
2. Prerak Shah

ABSTRACT

Sarcasm is often used by people to taunt or pester others. It is frequently expressed through inflection in speech or by using lexical, pragmatic, hyperbolic, and other features present in the text. Most of the work done by the researchers has been focused more on detecting sarcasm from the latter part, that is text data rather than the former that is audio. In this paper, we show that it is important to use both textual as well as audio features while detecting sarcasm. To carry out this task we propose a hybrid model, which takes a combined vector of extracted audio features as well as extracted text features from their respective models as an input. Our results indicate that the shortcomings of text were compensated by audio and vice-versa. The hybrid model outperforms both models where text and audio features were used individually.

CONTENTS

List of Tables

List of Images

List of Abbreviations

- MFCC – Mel Frequency Cepstral Coefficients
- MFC – Mel Frequency Cepstrum
- LSTM - Long Short-Term Memory
- SVM - Support Vector Machine
- RNN - Recurrent Neural network
- ReLU - Rectified Linear Unit
- BERT - Bidirectional Encoder Representations from Transformers
- GloVe - Global Vectors for Word Representation
- ELMo – Embeddings from Language Models
- tf-idf - term frequency–inverse document frequency
- CNN – Convolutional Neural Network
- NLP – Natural Language Processing
- HNR – Harmonics to Noise Ratio

CHAPTER 1 INTRODUCTION

1.1 BRIEF OF THE PROBLEM

With the growing number of virtual assistants with voice-to-voice interaction, detecting sarcasm has become crucial. AI assistants like Siri, Alexa, Cortana, etc. should be able to identify sarcasm in the user's speech. In 2018, Google used an AI assistant to book an appointment for a haircut by a voice call. So, in such scenarios it becomes very important that it is able to understand that humans are using sarcastic speech or not. Companies like Amazon, Flipkart etc., rely heavily on customer feedback and reviews in order to improve their products and services. Based on these reviews they take data driven decisions, so while analyzing them it becomes very important that they are able to identify the exact intent behind it. In such scenarios, they should be able to detect whether the given text review is sarcastic or not.

Macmillan English dictionary defines sarcasm as "the activity of saying or writing the opposite of what you mean, or of speaking in a way intended to make someone else feel stupid or show them that you are angry". For example in this statement " I love the pain present in the breakups" a shift in sentiment can be observed. Even though the sentence tries to convey that one loves the pain present in breakups, but the actual meaning that it tries to convey is the exact opposite. While such patterns are an indication of the presence of sarcasm in a statement, other lexical and pragmatic features as shown by [1],[2] also play an important role in the detection of sarcasm.

1.2 STUDY OF EXISTING SOLUTIONS

Majorly the work done by researchers has focused on detecting sarcasm through the help of textual cues. To detect sarcasm they manually extracted lexical, pragmatic, hyperbolic and other features from text. In order to extract these features from texts, techniques like tf-idf were used by them. Researchers have tried to find specific textual patterns to detect sarcasm like polarity shift in sentiment, etcetera. Researchers have also used deep-learning methods for features engineering tasks. Through the help of word embeddings the text could be represented in the form of vectors so they can be used as an input to these deep-learning models. Most of these works done by researchers used datasets which were created from twitter. Some work done by researchers has also used audio features like pitch, intonation, etcetera to detect sarcasm.

1.3 RESEARCH GAP

There are some cases where only text won't suffice in detecting sarcasm and additional cues would be required to detect it accurately. For example the phrase "yeah right" would have different meanings which depends on how the person says it and what is the context behind it [3]. In such cases, factors like changes in tone, overemphasis of words, drawn-out syllables etcetera would play an important role in deciding whether it is sarcastic or not. Similarly there are cases exactly opposite to this in which only audio would not be enough to detect sarcasm. For example if a person says "I love being ignored" in a flat monotone way then it would not be possible to detect sarcasm by only using audio. In such cases text needs to be analyzed and textual patterns needs to be found which would help in detecting sarcasm. Currently the major work done in detecting sarcasm is focussed on detecting sarcasm in text, especially in twitter datasets. Very few

researchers have focused on using audio in detecting sarcasm or using a hybrid approach to detect sarcasm.

CHAPTER 2 LITERATURE REVIEW

Most of the work done till date has focussed majorly on textual cues in order to detect sarcasm. While working with text, researchers used to carry out feature engineering on the dataset in order to detect sarcasm. In early days, sarcasm was detected by observing some particular patterns like +ve comment in -ve situatutions [4]. Researchers have also used lexical features like unigram, bigram, trigram etc. in order to detect sarcasm. The importance of lexical features in detecting sarcasm and irony was first observed by Kreuz et al [5]. Features like punctuation symbols and interjection also play an important role in recognizing sarcasm [2]. Along with these, features like quotes, intensifiers etc can be broadly classified into hyperbolic features. Utsumi [6] showed that by using such features stress can be given on a particular word present in the text, which will act as a cue for the presence of sarcasm. With the help of pragmatic features like emoticons, smilies and special punctuations, Carvalho et al. [7] detected irony from newspaper text data. The usage of such features would be highly effective while detecting sarcasm from tweets, as they usually contain such features.

However, recently researchers have left the task of feature engineering on the deep learning models itself rather than performing it manually [8]. While working with these deep-learning models preprocessing is required for the input text. For each word present in the sentence it is converted into it's embedding vector. Such embedding vectors can be generated with the help of pre-trained models like GloVe [9]. Mandal et al. [10] achieved an accuracy of 86.6% while detecting sarcasm from news headlines. Their model consisted of a CNN-LSTM neural network with inputs as word

embedding vectors of news headlines. Zhang et al. [8] used bi-directional gated recurrent neural networks and discrete models to detect sarcasm. Their results show that the neural models outperform their discrete models while detecting sarcasm from tweets. Researchers have seen an improvement in the accuracy of NLP tasks while using pre-trained word embeddings [11],[12].

While detecting sarcasm, humans try to look for cues like changes in tone or frequency of the sound(or audio). Researchers have exploited these methods to get the models detect sarcasm from audio. Acoustic features like mean f0, standard deviation of f0, f0 range, mean amplitude, amplitude range, speech rate, harmonics-to-noise ratio (HNR), and one third octave spectral values (as a measure of nasality) can be used to detect sarcasm in audio [13]. Rachel Rakov et al. [14] have developed a model for automatic sarcasm detection. By using k-means clustering they applied sequential modeling of categorical representations of pitch and intensity contours. They discovered that some particular intensity and pitch contours are indicative of sarcastic speech. Vocal cues were utilized by Rockwell et al [15] in order to detect sarcasm. Their results indicated that intense low pitched utterance with slow cadence has higher probability of being sarcastic. Lœvenbruck, Hélène, et al. [16] discovered that irrespective of linguistic context acoustic features can be used to detect sarcasm even in French speech. They also observed that the majority of sarcastic utterances had some common properties like increased f0 modulations, and utterance lengthening. Woodland and Voyer [17] showed that prosodic features like intonation and stress play an important role in detecting sarcasm.

Another method to extract audio features is to use Mel-Frequency Cepstral Coefficients (MFCCs). Every sound signal can be uniquely represented as an envelope of time power spectrum, which is

also known as Mel-Frequency Cepstrum (MFC). Human's vocal tract takes an unique shape for unique sounds and this shape is represented in the time power spectrum of the sound. A MFC is represented by it's coefficients which are known as MFCCs.

Tiwari et al. [18] have used MFCCs, as a compact and an effective way, to represent the audio files for a speech processing task.

According to Castro, Santiago, et al. , multimodal cues can help to improve sarcasm detection [19]. They have observed a reduction in relative error rate of sarcasm when using multimodal cues instead of individual modalities. A hierarchical multimodal architecture has been used by Gu, Yue, et al. [20] for classifying sentiment and emotion from text and audio. Cai et al [21] developed a multimodal sarcasm detection model to detect sarcasm from tweets. They considered three modalities, text features, image features and image attributes and constructed a hierarchical fusion model to carry out sarcasm detection.

The types of features used for detecting sarcasm in text and audio are summarized in Figure 1.

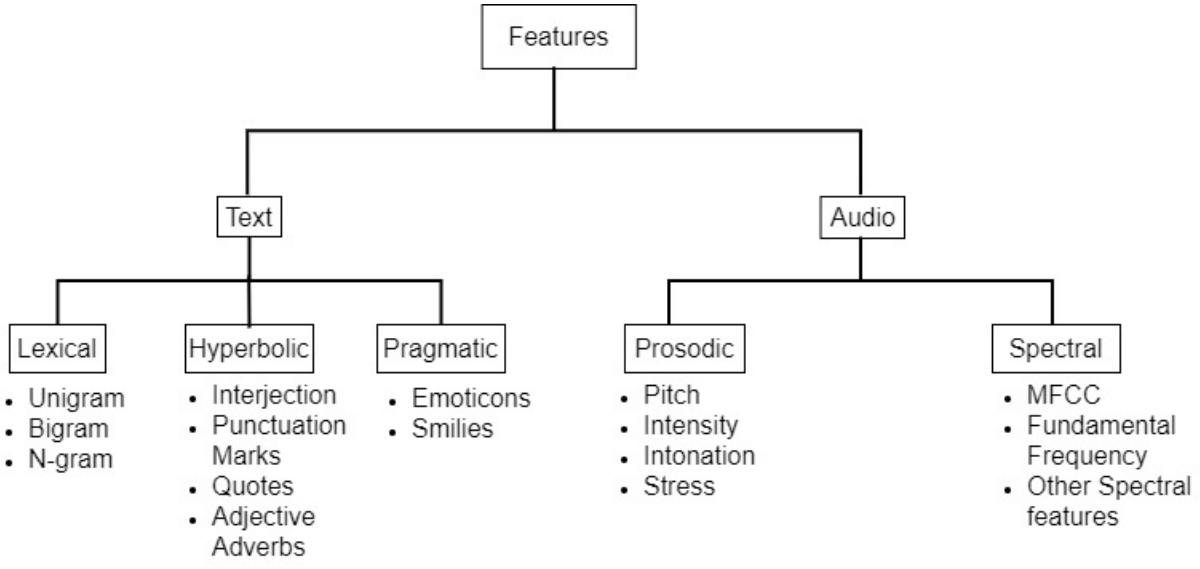

Figure 1. Types of text and audio features used for sarcasm detection

CHAPTER 3 METHODOLOGY

3.1 PROPOSED SYSTEM

3.1.1 Audio Model

In the preprocessing section for the Audio model, the mp4 files are converted to .wav files. Mfcc are created from the wav files and sliced into 1 second segments. These segments are trained on a LSTM Deep Learning model. The last layer of the rnn model is removed to convert it to an encoder and the encoded vectors of all segments are fetched and averaged out to create the vectors for the entire audio files. After this the vectors are trained in the audio model to predict the sarcasm for the entire audio files. The visual representation of this model is shown in Figure 2.

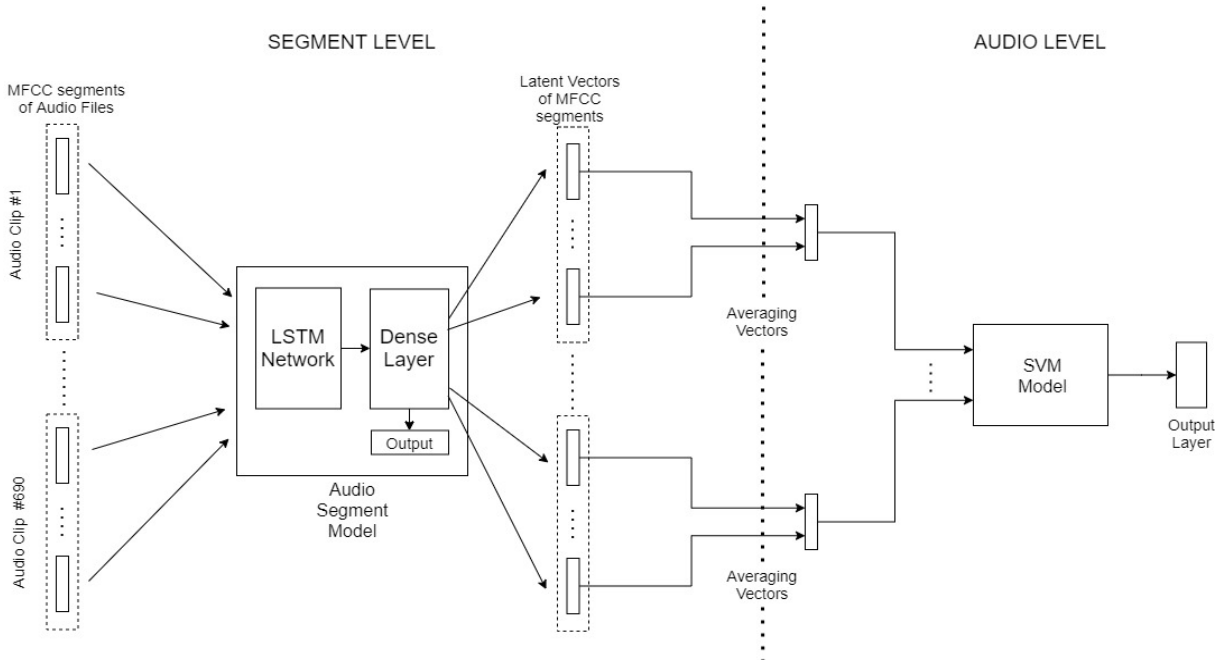

Figure 2. Audio Model

3.1.2 Text Model

In the text model, first preprocessing was performed to remove unwanted characters from the text and clean it. Next through the help of a tokenizer, tokens were generated for the text. Further through the help of pre-trained word embeddings the text is converted into it's vectors format. Now through the help of padding all the vectors are converted to a uniform length vector. Finally these padded vectors are used to train a CNN model in order to detect sarcasm from text. The visual representation of this model is shown in Figure 3.

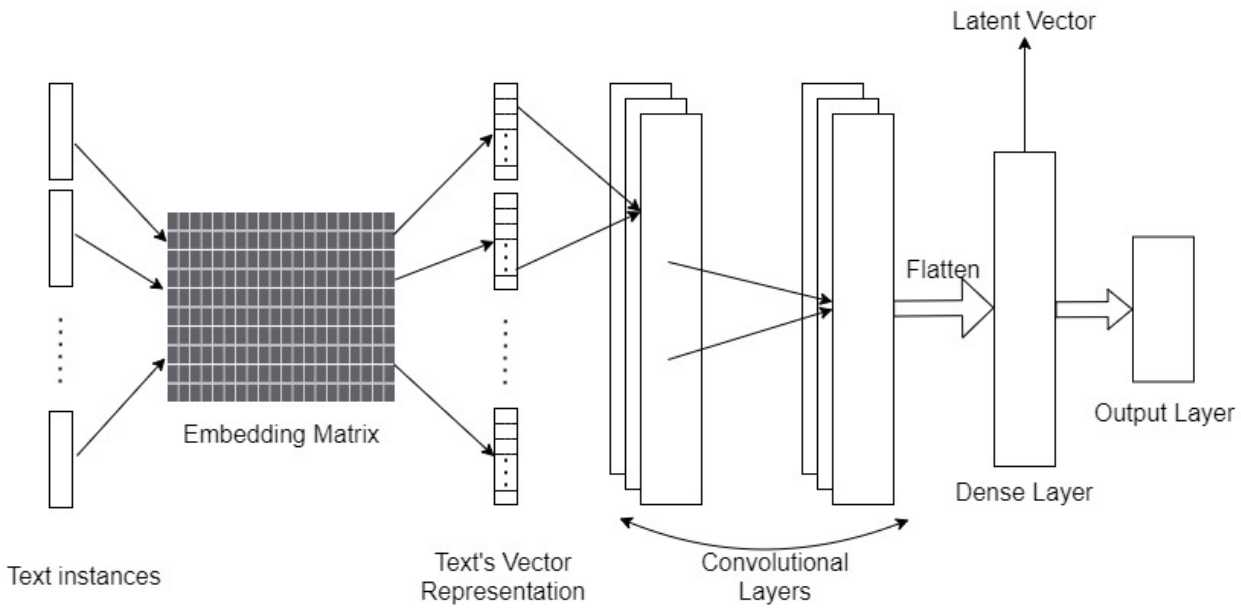

Figure 3. Text Model

3.1.3 Hybrid Model

In the hybrid model the latent vectors generated in the audio model and the text model are fetched and combined. A 164 length vector is created from a 64 length audio vector and 100 length text vector. These vectors are then trained on a classifier to detect the sarcasm from both audio and text files. The visual representation of this model is shown in Figure 4.

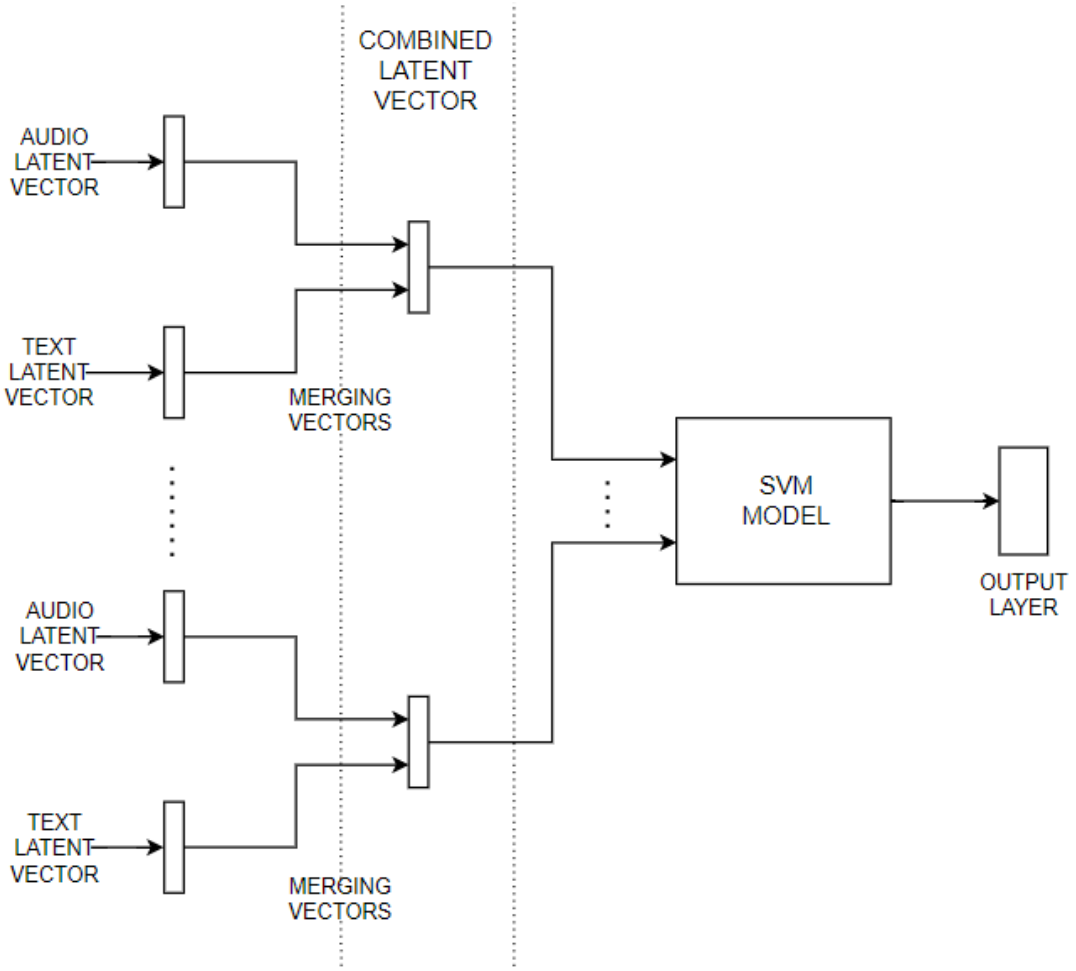

Figure 4. Hybrid Model

3.2 TOOLS AND TECHNOLOGIES INVOLVED

Coding Language: Python3

Coding Environments: Google Colab (.ipynb files)

3.2.1 Libraries

Following libraries were used in our project:

1. Librosa

   It provides the necessary requirements to carry out audio and music analysis for information retrieval tasks.

2. Json

   It is used to handle .json files.

3. Keras

   It provides a Python interface for artificial neural networks.

4. Numpy

   It is a basic python library for adding support for large arrays and matrices. It also provides a large collection of high-level mathematical functions to operate on them.

5. Sklearn

   It is a library for machine learning techniques in Python

6. Pandas

Through the help of its data structures numerical tables and time series can be manipulated.

7. Matplotlib

   It helps in creating various visualizations in Python.

8. PyTorch

   It is used for machine learning applications for computer vision and natural language processing.

9. Spacy

   It is used for advanced natural language processing.

10. Pickle

    It is used to transform a complex object into a byte stream and vice versa.

11. re (regex)

    It provides regular expressions for pattern matching within strings.

12. Tensorflow

    It is used for machine learning tasks with a major focus on training and inference of deep neural networks.

3.2.2 Word Embeddings

Word embeddings are used to represent the words in a vector form for nlp tasks. The vectors are created in such a way that the words with similar meaning will have closer vectors. They are created from a model which is pretrained on a different huge and standard corpus.

Following are the word embeddings used in this project:

1. Bidirectional Encoder Representations from Transformers (BERT)
2. Gensim Word2Vec
3. Embeddings from Language Models (ELMo)

CHAPTER 4 IMPLEMENTATION

4.1 AUDIO MODEL

4.1.1 Data Preprocessing

```
import os
for j in filenames:
  k = f1+'/'+str(j)
  k2 = '/content/drive/MyDrive/Sarcasm/Dataset/mustard/utterance_audio'+'/'+str(j)
  os.system("ffmpeg -i "+k+" "+k2[:-4]+".wav")
```

Figure 5. Read audio file

Each of the 690 files of the dataset were converted from .mp4 format to .wav format.

4.1.2 Create Mel Frequency Cepstral Coefficients (MFCCs)

```
def create_MFCC(data):
  hop_length = 512
  n_fft = 2048
  X = []
  y = []
  for i in data['name']:
    f = "/content/drive/MyDrive/Sarcasm/Dataset/mustard/utterance_audio/"+str(i)+".wav"
    signal, sample_rate = librosa.load(f, sr=22050)
    MFCCs = librosa.feature.mfcc(signal, sample_rate, n_fft=n_fft, hop_length=hop_length, n_mfcc=13)
    X.append(MFCCs)
  return(X)
```

Figure 6. Create MFCC

Using Python's Librosa library each of the .wav files was converted into their respective Mel-frequency Cepstral Coefficients (MFCC). The files were read by the “librosa.load()” function with a sampling rate of 22050 Hz. Each loaded file was converted into it’s MFCC representation using the function “librosa.feature.mfcc()” having parameter values as number of fast fourier transform windows (n_fft)=2048, hop_length=512 and number of MFCCs (n_mfcc)=13.

4.1.3 Slice Mel Frequency Cepstral Coefficients (MFCCs)

```
def slice_MFCC(data, X):
  l = data['name']
  X_seg = []
  y_seg = []
  win_count_train = []
  for i in range(len(X)):
    s = 0
    e = s+46
    p = 1
    x = X[i].T
    win_count_train.append(0)
    while(e<x.shape[0]):
      win_count_train[-1] += 1
      X_seg.append(np.array(x[s:e]))
      if( int(data[data['name']==l[i]]['start_win']) <=p and int(data[data['name']==l[i]]['end_win']) >=p):
        y_seg.append(1)
      else:
        y_seg.append(0)
      s,e,p = e, e+46, p+1
  return(X_seg, y_seg, win_count_train)
```

Figure 7. Slice MFCC

Due to the varying size of audio clips, it’s MFCC representation would be of different lengths. In order to make the input uniform for the training model, the MFCCs were sliced according to the window size of (13,46), which is equivalent to the size of a 1 second audio clip. These 1 second segmented audio clips would map one-to-one with the output labels which were created while modifying the dataset.

4.1.4 Long Short-Term Memory (LSTM) Model

```
def build_model(input_shape):
    model = keras.Sequential()
    model.add(keras.layers.LSTM(64, input_shape=input_shape, return_sequences=True))
    model.add(keras.layers.LSTM(64))
    model.add(keras.layers.Dense(64, activation='relu'))
    model.add(keras.layers.Dropout(0.3))
    model.add(keras.layers.Dense(2, activation='softmax'))

    return model

model = build_model((46,13))
optimiser = keras.optimizers.Adam(learning_rate=0.0001)
model.compile(optimizer=optimiser,loss='sparse_categorical_crossentropy', metrics=['accuracy'])

model.summary()
```

Figure 8. Create audio segment model

The above generated MFCC segments along with their output labels are used to train a deep learning model (also known as Audio Segment Model) which detects sarcasm in an audio clip of 1 second duration. This Audio Segment Model consists of a recurrent neural network (RNN). The input layer of this model is a Long Short-Term Memory (LSTM) layer that would accept a 46x13 sized matrix and would convert it to a 1-d array of 64 length. The output from this layer would be passed as an input to one more LSTM layer which also generates a 1-d array of 64 length. The next layer is a dense layer having activation function as Rectified Linear Unit (Relu) and a dropout value of 30%. The final layer is a dense layer with a softmax activation function which classifies whether the MFCC segment is sarcastic or not.

4.1.5 Feature Conversion

```
matrix_size = model.layers[-2].output.shape[1]
new_model = Model(model.inputs, model.layers[-2].output)
lT = new_model.predict(np.array(X_seg))
lt = new_model.predict(np.array(X_test))

new_out_train = []
new_y_train = []
j = 0
for i in win_count_train:
  new_out_train.append(np.mean(np.array(lT[j:j+i]), axis=0))
  j += i
  new_y_train.append(1)

new_out_test = []
j = 0
for i in win_count_test:
  new_out_test.append(np.mean(np.array(lt[j:j+i]), axis=0))
  j += i
```

Figure 9. Feature Conversion

The Audio Segment Model works on audio clips of fixed length 1 second, but in the dataset the audio clips are of varied lengths.So, in order to classify the entire audio clip into sarcastic or not sarcastic, the last layer from the audio segment model is removed. After removing the last layer, it would encode the (46,13) input audio segment to a 64 length array. As the audio segment model works on 1 second segments of audio clips, the output of all segments of a clip would be averaged out to generate a single output for the entire clip.

4.1.6 Audio Model Training

```
from sklearn.model_selection import KFold
final_X = np.concatenate((X_final_train,X_final_test), axis=0)
final_Y = np.concatenate((y_final_train,y_final_test), axis=0)
kf = KFold(n_splits=5, shuffle=True)
TA = []
F1 = []
PR = []
RL = []
for train_index, test_index in kf.split(final_X):
  kfX_train, kfX_test = final_X[train_index], final_X[test_index]
  kfy_train, kfy_test = final_Y[train_index], final_Y[test_index]
  dfT = pd.DataFrame(kfX_train)
  dft = pd.DataFrame(kfX_test)
  dfT = dfT.fillna(0)
  dft = dft.fillna(0)
  clf1 = svm.SVC()
  clf1.fit(dfT, kfy_train)

  TA.append(clf1.score(dft,kfy_test))
  y_final_pred_1 = clf1.predict(dft)
  F1.append(f1_score(y_final_pred_1, kfy_test))
  PR.append(precision_score(y_final_pred_1, kfy_test))
  RL.append(recall_score(y_final_pred_1, kfy_test))
```

Figure 10. Train audio model

These averaged outputs of all segments for each audio clip would be given as an input to a SVM model, which would classify the audio clip into either sarcastic or not sarcastic.

4.2 TEXT MODEL

4.2.1 Preprocessing the input text

```
t = Tokenizer()
t.fit_on_texts(text)
vocab_size = len(t.word_index) + 1
train = t.texts_to_sequences(text)
max_length = 60
padded_comments_train = pad_sequences(train, maxlen=max_length, padding='post')
y_train = to_categorical(out, num_classes=2)
```

Figure 11. Text Preprocessing

For each audio file present in the dataset, there was a corresponding transcript available, from which utterance-text and sarcasm labels were fetched and stored into a pandas dataframe. The sarcasm label was modified using label encoder where 'True' was converted to 1 and 'False' to 0. By using the tokenizer available from the Keras preprocessing text library, all the texts were tokenized. Then, "texts_to_sequences" function of the tokenizer was used to vectorize each sentence. Due to varying lengths of texts, the vectors generated are also of different lengths and as a result padding would be required to make the input size uniform. Since the majority of vector length was less than 60, the input length of the model was fixed at 60. And, as a result those vectors having a size less than 60 were padded.

4.2.2 Embedding matrix

```
w2v_model = KeyedVectors.load_word2vec_format('GoogleNews-vectors-negative300.bin.gz', binary=True)
embedding_matrix_w2v = np.zeros((vocab_size, 300))
for word, i in t.word_index.items():
    try:
        embedding_vector = w2v_model[word]
    except:
        embedding_vector = [0]*300
    if embedding_vector is not None:
        embedding_matrix_w2v[i] = embedding_vector
```

Figure 12. Embedding matrix

After each instance of the dataset is preprocessed and padded, it used to train a model which would detect whether the text is sarcastic or not. The first layer of the model is an input layer, which would accept a vector of length 60. This layer would be connected to an embedding layer which would convert the 60 length array to a 300 length array. Instead of randomly assigning weight values to the (60,300) matrix, these weights were fetched from the pretrained model of the gensim package. If the word from the vocabulary is present in the gensim model, then it's vector is fetched otherwise it will be filled with zeros.

4.2.3 Text Model Training

```
input_data = Input(shape=(max_length,), name='main_input')
embedding_layer = Embedding(vocab_size, 300, weights=[embedding_matrix_w2v], trainable=False)(input_data)
conv_1 = Conv1D(filters=50, kernel_size=4, activation='relu')(embedding_layer)
max_1 = MaxPooling1D(pool_size=2)(conv_1)
conv_2 = Conv1D(filters=100, kernel_size=3, activation='relu')(max_1)
max_2 = MaxPooling1D(pool_size=2)(conv_2)
lstm_layer = Bidirectional(LSTM(128,return_sequences=True))(max_2)
flatten = Flatten()(lstm_layer)
dense = Dense(100, activation='relu', name='fully_connected')(flatten)
out = Dense(2, activation='softmax')(dense)
model_01 = Model(inputs=[input_data], outputs=[out])
```

Figure 13. Train text model

The embedding layer is followed by 2 conv1d layers, each of which is followed by a max_pooling layer. Now the output is flatten and it is followed by a dense layer with Relu activation function. The final layer is also a dense layer with a softmax classifier, which identifies whether the input text is sarcastic or not.

4.3 HYBRID MODEL

```
X_final_train = np.concatenate((X_text_train, X_audio_train), axis=1)
X_final_test = np.concatenate((X_text_test, X_audio_test), axis=1)
final_X = np.concatenate((X_final_train,X_final_test), axis=0)
final_Y = np.concatenate((y_final_train,y_final_test), axis=0)
kf = KFold(n_splits=5, shuffle=True)

TA = []
F1 = []
PR = []
RL = []
for train_index, test_index in kf.split(final_X):
  kfX_train, kfX_test = final_X[train_index], final_X[test_index]
  kfy_train, kfy_test = final_Y[train_index], final_Y[test_index]
  dfT = pd.DataFrame(kfX_train)
  dft = pd.DataFrame(kfX_test)
  dfT = dfT.fillna(0)
  dft = dft.fillna(0)
  clf1 = svm.SVC()
  clf1.fit(dfT, kfy_train)

  TA.append(clf1.score(dft,kfy_test))
  y_final_pred_1 = clf1.predict(dft)
  F1.append(f1_score(y_final_pred_1, kfy_test))
  PR.append(precision_score(y_final_pred_1, kfy_test))
  RL.append(recall_score(y_final_pred_1, kfy_test))
```

Figure 14. Train hybrid model

In order to identify sarcasm by using both text and audio, the last layer from each of the models is removed to convert the models into encoders. The audio encoder model will be used to generate

the vector representation of the entire audio file. Similarly, after removing the last layer from the text model, it would encode the text input to a 100 length array.

The output of both the encoder models would be combined to form a 164 length array representation of audio and text features of the clip. Now, this array would be given as input to a Support Vector Classifier which would classify it into either sarcastic or non sarcastic.

CHAPTER 5 EXPERIMENTAL RESULTS

The experimental results of the proposed methodology have been discussed in this section. Firstly, we detected sarcasm by using only audio. In Table. 1 results of our audio model were compared to the model proposed by Castro, Santiago, et al. [19], for 80:20 ratio of training and testing data, in which our audio model outperformed the other with almost 7% higher F-score.

Model	**Precision**	**Recall**	**F-Score**
Santiago, et al. [19]	65.10	62.60	62.70
Audio Model	**73.66**	**66.36**	**69.71**

Table 1. Audio Model Comparison

Next, only text was used to detect sarcasm. For this task, three different pre-trained word embeddings were used in the text model and its effects were observed. The comparison for these cases are being shown in table 2.

Word Embedding	**Precision**	**Recall**	**F-Score**
Gensim Word2Vec	**67.5**	**66.66**	**67.08**
BERT	61	61	61

ELMo	54.85	54.34	54.36

Table 2. Text Model with different word embedding

Finally, both audio and text modalities were used to detect sarcasm. In order to train the hybrid model, the dataset was randomly split into training and testing sets with 80% of the dataset used for training. In order to evaluate the hybrid model we performed a 5-Fold cross validation. This would ensure that our model performs well even with unseen data. Table 3 contains the averaged results of 5 folds for hybrid models with different classifiers.

Classifiers	**Precision**	**Recall**	**F-Score**	**Accuracy**
SVM	**73.26**	65.39	**69.01**	**67.10**
Logistic regression	67.24	**66.49**	66.64	66.38
Gaussian Naive Bayes	67.73	62.23	64.77	63.33
Multi-layer Perceptron	62.45	60.70	61.27	60.72

Table 3. Hybrid Model with different classifiers averaged across 5-Folds

The F-Score along different K folds for all classifiers are shown in Figure 6 and the Accuracy is shown in figure 7.

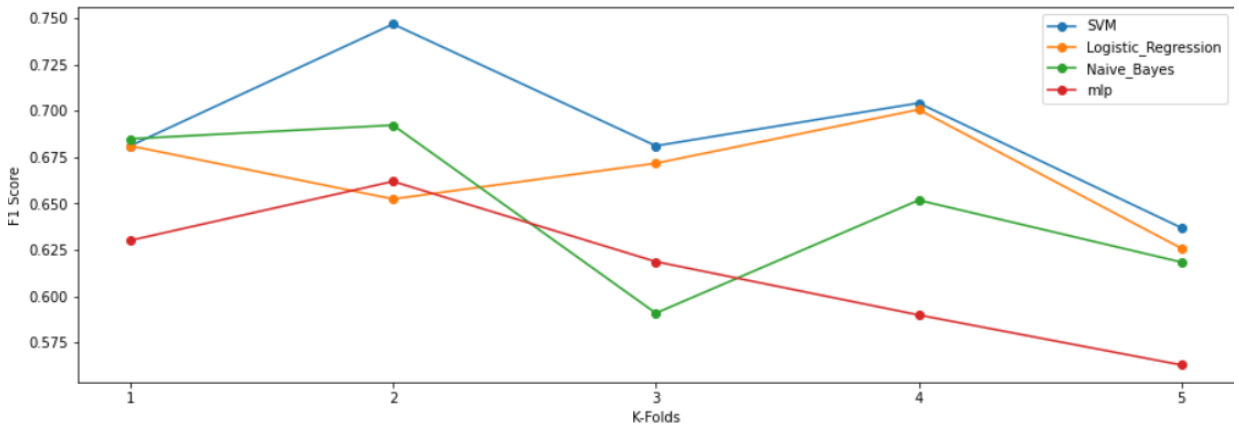

Figure 15. F1 scores across different k folds for various classifiers

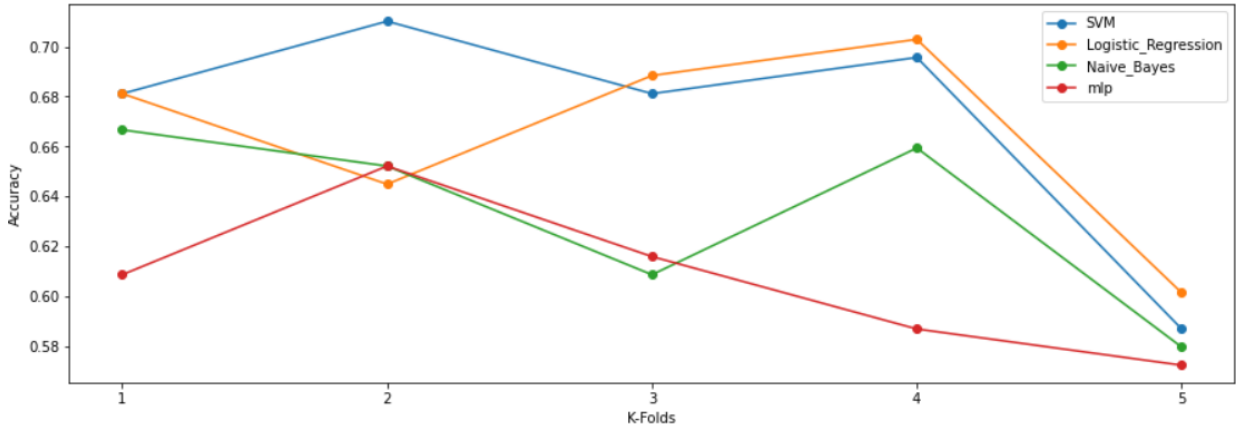

Figure 16. Accuracy across different k folds for various classifiers

Our best models of each modalities are compared with the models proposed by Santiago, et al. [19] in table 4.

Modality	Model	Precision	Recall	F-Score
Text (T)	Text Model	67.5	**66.66**	67.08
	Santiago, et al. [19]	60.9	59.6	59.8
Audio (A)	Audio Model	73.66	66.36	69.71
	Santiago, et al. [19]	65.1	62.6	62.7
Hybrid (T+A)	Hybrid Model	**75.11**	66.28	**70.35**
	Santiago, et al. [19]	64.7	62.9	63.1

Table 4. Results of hybrid model with different classifiers

In the model in which only text is used as an input to detect sarcasm, a F-score of 67.08 is achieved. Similarly, when only audio is used we get a F-score value of 69.71. In the case of the hybrid model, where both audio and text are used as an input to detect sarcasm, a F-score of 70.35 was observed.

On further analyzing the instances of the dataset where the hybrid model correctly identifies sarcastic utterance whereas the text model is unable to do so, it was found that the majority of these text instances, by itself, cannot be considered sarcastic. In these cases it is the way in which the sentence is spoken which makes it sarcastic. Some of these cases are shown in Table 5.

ID	Text
1_5699	Congratulations and welcome temporarily aboard. Here's your I.D. card, your key and your lapel pin. Which Leonard was too cool to wear.
2_102	No Blanche she is upset because they keep changing the taste of coke.
2_485	You got off to a really good start with the group.
2_287	Oh I am sorry, do you need a break?
2_92	No, no, I am just looking for a man to draw on me with chalk.

Table 5. Instances where audio features were more useful than text features (Stress was given to the bold part of the text)

In the examples shown in Table 5 we observe that it is not possible to detect sarcasm just from text. By observing the audio of these instances, it was found that it was the stress given by the speakers on the bold part of the sentence which made it sarcastic.

Similarly, opposite scenarios are also possible where sarcasm cannot be detected by using only audio features, in these cases it is the text itself which makes it sarcastic and not the way of text delivery. Some of these examples are shown in Table 6. In these examples the sentences are said with a straight face so it would be difficult to detect sarcasm by using audio only. It is the textual

features like polarity change, presence of contradicting sentiments, etcetera which make it sarcastic. For example, considering waitressing a complex and critical activity or needing to listen to snores while sleeping.

ID	**Text**
1_11177	I'm listening to you snore. I'm wondering how I'll ever sleep without it.
1_2853	Obviously, waitressing at The Cheese Cake Factory is a complex socio-economic activity that requires a great deal of analysis and planning.
2_323	That monkey has got a Ross on its ass!
1_9087	Why? Because I got an ugly, itchy sweater, and my brother got a car? No, I was her favorite.
1_5964	Smart. Whisper so the deaf chick can't hear you.

Table 6. Instances where text features were more useful than audio features

CHAPTER 6 CONCLUSION

In this paper we propose using both text and audio features to detect sarcasm. Three models were created, first the text model which works with only textual features, second the audio model which works with only audio features and third the hybrid model which works with both text and audio features. The text and audio features for the hybrid model were fetched as latent vectors from the other two models respectively. From the results as shown in Table 1 it can be seen that the Hybrid model gives the best performance among the three models. This is due to text features and audio features compensating for each other's shortcomings. So, these results support our hypothesis that using both text and audio increases the probability of detecting sarcasm.

References

1. González-Ibáñez, Roberto, Smaranda Muresan, and Nina Wacholder. "Identifying sarcasm in Twitter: a closer look." Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies. 2011.
2. Kreuz, Roger, and Gina Caucci. "Lexical influences on the perception of sarcasm." Proceedings of the Workshop on computational approaches to Figurative Language. 2007.
3. Tepperman, Joseph, David Traum, and Shrikanth Narayanan. "" Yeah Right": Sarcasm Recognition for Spoken Dialogue Systems." Ninth international conference on spoken language processing. 2006.
4. Riloff, E., Qadir, A., Surve, P., De Silva, L., Gilbert, N. and Huang, R., 2013, October. Sarcasm as a contrast between a positive sentiment and negative situation. In Proceedings of the 2013 conference on empirical methods in natural language processing (pp. 704-714).
5. R. J. Kreuz and R. M. Roberts, “Two cues for verbal irony: Hyperbole and the ironic tone of voice,” Metaphor and symbol, vol. 10, no. 1, pp. 21–31, 1995.
6. Utsumi, A. (2000). Verbal irony as implicit display of ironic environment: Distinguishing ironic utterances from nonirony. Journal of Pragmatics, 32(12), 1777–1806. doi:10.1016/S0378-2166(99)00116-2
7. P. Carvalho, L. Sarmento, M.J. Silva, E. De Oliveira, Clues for detecting irony in user-generated contents: oh...!! it's so easy;-), in: Proceedings of the 1st International CIKM Workshop on Topic-Sentiment Analysis for Mass Opinion, ACM, 2009, pp. 53–56.
8. Zhang, Meishan, Yue Zhang, and Guohong Fu. "Tweet sarcasm detection using deep neural network." Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: technical papers. 2016.
9. Pennington, Jeffrey, Richard Socher, and Christopher D. Manning. "Glove: Global vectors for word representation." Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). 2014.
10. Mandal, Paul K., and Rakeshkumar Mahto. "Deep CNN-LSTM with Word Embeddings for News Headline Sarcasm Detection." 16th International Conference on Information Technology-New Generations (ITNG 2019). Springer, Cham, 2019.
11. Ronan Collobert, Jason Weston, Leon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. ´ Natural language processing (almost) from scratch. The Journal of Machine Learning Research, 12:2493–2537.
12. Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 740–750, Doha, Qatar, October. Association for Computational Linguistics.

13. Cheang, H.S. and Pell, M.D. “The sound of sarcasm”, Speech Commun. 50, 366-381, 2008.
14. Rakov, Rachel, and Andrew Rosenberg. "" sure, i did the right thing": a system for sarcasm detection in speech." Interspeech. 2013.
15. Patricia Rockwell. 2000. Lower, slower, louder: Vocal cues of sarcasm. Journal of Psycholinguistic Research, 29(5):483–495.
16. Lœvenbruck, Hélène, et al. "Prosodic cues of sarcastic speech in French: slower, higher, wider." 14th annual conference of the international speech communication association (interspeech 2013). 2013.
17. Jennifer Woodland and Daniel Voyer. 2011. Context and intonation in the perception of sarcasm. Metaphor and Symbol, 26(3):227–239
18. Tiwari, V., 2010. MFCC and its applications in speaker recognition. International journal on emerging technologies, 1(1), pp.19-22.
19. Castro, Santiago, et al. "Towards multimodal sarcasm detection (an _Obviously_ perfect paper)." arXiv preprint arXiv:1906.01815 (2019).
20. Gu, Yue, et al. "Multimodal affective analysis using hierarchical attention strategy with word-level alignment." Proceedings of the conference. Association for Computational Linguistics. Meeting. Vol. 2018. NIH Public Access, 2018.
21. Cai, Yitao, Huiyu Cai, and Xiaojun Wan. "Multi-modal sarcasm detection in twitter with hierarchical fusion model." Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. 2019.

Project Group Personal Details

1. Name of the Student: Zeel Prajapati

Permanent Address: A-11, Vrundavan Flats, Nr. Vaibhav Tower,
Premchand Nagar Road, Satellite,
Ahmedabad
Email: prajapatizeel75@gmail.com
Mobile no: 918128533225

2. Name of the Student: Prerak Shah

Permanent Address: B/5, Avani Apartment, Keshavnagar,
Ahmedabad

Email: prerakjaiminshah@gmail.com
Mobile no: 917984386313