Spring 5-20-2019

Detection of Hate Speech in Videos Using Machine Learning

Unnathi Bhandary
San Jose State University

Follow this and additional works at: https://scholarworks.sjsu.edu/etd_projects

Part of the [Artificial Intelligence and Robotics Commons](#)

Recommended Citation

Bhandary, Unnathi, "Detection of Hate Speech in Videos Using Machine Learning" (2019). *Master's Projects*. 691.

DOI: <https://doi.org/10.31979/etd.5efp-73s4>

https://scholarworks.sjsu.edu/etd_projects/691

This Master's Project is brought to you for free and open access by the Master's Theses and Graduate Research at SJSU ScholarWorks. It has been accepted for inclusion in Master's Projects by an authorized administrator of SJSU ScholarWorks. For more information, please contact scholarworks@sjsu.edu.

Writing Project

Detection of Hate Speech in Videos Using Machine Learning

Final Report

Author

Unnathi Bhandary

CS 298

May 2019

Advisor

Dr. Mike Wu

Detection of Hate Speech in Videos Using Machine Learning

A Writing Project Presented to The Faculty of the Department of Computer Science
San Jose State University

In Partial Fulfillment of the Requirements for the Degree: Master of Science

© 2019

Unnathi Bhandary

ALL RIGHTS RESERVED

The Designated Committee Approves the Master's Project Titled

Detection of Hate Speech in Videos Using Machine Learning

By

Unnathi Bhandary

Approved for the Department of Computer Science

San José State University

May 2019

Dr. Mike Wu

Department of Computer Science

Dr. Robert Chun

Department of Computer Science

Dr. Samuel Chen

PG&E

Acknowledgments

I would like to express my utmost gratitude to my project advisor, Dr. Mike Wu, for providing invaluable guidance, constant support, encouragement and time that enabled me to complete this project successfully.

I would also like to thank my committee members, Dr. Robert Chun and Dr. Samuel Chen, for their valuable input and suggestions throughout of duration of my project.

Lastly, I would like to extend my heartfelt gratitude to my friends and family for their continuous support, encouragement, and blessings without which I would not able to finish my project and graduate successfully.

ABSTRACT

With the progression of the Internet and social media, people are given multiple platforms to share their thoughts and opinions about various subject matters freely. However, this freedom of speech is misused to direct hate towards individuals or group of people due to their race, religion, gender etc. The rise of hate speech has led to conflicts and cases of cyber bullying, causing many organizations to look for optimal solutions to solve this problem.

Developments in the field of machine learning and deep learning have piqued the interest of researchers, leading them to research and implement solutions to solve the problem of hate speech. Currently, machine learning techniques are applied to textual data to detect hate speech. With the ample use of video sharing sites, there is a need to find a way to detect hate speech in videos.

This project deals with classification of videos into normal or hateful categories based on the spoken content of the videos. The video dataset is built using a crawler to search and download videos based on offensive words that are specified as keywords. The audio is extracted from the videos and is converted into textual format using a Speech-to-Text converter to obtain a transcript of the videos.

Experiments are conducted by training four models with three different feature sets extracted from the dataset. The models are evaluated by computing the specified evaluation metrics. The evaluated metrics indicate that Random Forrest Classifier model delivers the best results in classifying videos.

Table of Contents

1. Introduction	10
1.1 Background	10
1.2 Machine Learning	11
1.3 Deep Learning	12
1.4 Problem Statement	12
2. Literature Review	15
2.1 Classification of Videos	17
3. Methodology	19
3.1 Dataset Creation	19
3.2 Dataset Processing	22
3.3 Model Training	25
3.3.1 Feature Engineering	25
3.3.2 Models	27
3.4 Sentiment Analysis	30
4. Evaluations and Results	31
4.1 Experiments	31
4.2 Evaluation Metrics	32
4.3 Results	34
4.4 Comparison with Existing Approaches	42
5. Conclusion	46
References	47
Appendix	50

Tables of Figures

Figure 1 Example of Hate Speech	10
Figure 2 Example of Online Hate Speech	13
Figure 3 Hate Speech Detection Framework	19
Figure 4 Example of Offensive Videos	20
Figure 5 Dataset Creation Module	21
Figure 6 Dataset Processing Module	22
Figure 7 A snippet of Snippet-to-Text code	24
Figure 8 Model Training Module	25
Figure 9 Random Forrest Classifier	28
Figure 10 A Simple Diagram of SVM	29
Figure 11 A Simple Recurrent Neural Network	29
Figure 12 Dataset Distribution for Two Labels	31
Figure 13 Dataset Distribution for Three Labels	32
Figure 14 Accuracy of Models Given Two Labels	35
Figure 15 Accuracy of Models Given Three Labels	35
Figure 16 Precision Score of Models Given Two Labels	36
Figure 17 Precision Score of Models Given Three Labels	36
Figure 18 Recall Score of Models Given Two Labels	37
Figure 19 Recall Score of Models Given Three Labels	37
Figure 20 F1 Score of Models Given Two Labels	38
Figure 21 F1 Score of Models Given Three Labels	38
Figure 22 ROC Curve for Multinomial Naïve Bayes	39
Figure 23 ROC Curve for Linear SVM	40
Figure 24 ROC Curve for Random Forrest Classifier	40
Figure 25 ROC Curve for RNN	41
Figure 26 Feature set Comparison for Two Labels	41
Figure 27 Feature set Comparison for Three Labels	42
Figure 28 Comparison of Average Precision Scores	43
Figure 29 Comparison of Average Recall Scores	43

Figure 30 Comparison of Precision Scores	44
Figure 31 Comparison of Recall Scores	44
Figure 32 Comparison of F1 Scores	45

Table of Tables

TABLE 1. Classification of Videos	22
TABLE 2. Example Results of Speech-to-Text Conversion	23
TABLE 3. Example of Sentiment Analysis	29
TABLE 4. Confusion Matrix of a Binary Classifier	33
TABLE 5. Average Results of Experiment 1	34
TABLE 6. Average Results of Experiment 2	34

1. INTRODUCTION

1.1 Background

Everyone has the right to freedom of speech. However, this right is being misused to discriminate and attack others, physically or verbally, in the name of free speech. This discrimination is known as hate speech. Hate speech can be defined as speech used to express hate towards a person or a group of people based on characteristics such as race, religion, ethnicity, gender, nationality, disability and sexual orientation. According to Nockleby [1], hate speech can be defined as “any communication that disparages a person or a group on the basis of some characteristic such as race, color, ethnicity, gender, sexual orientation, nationality, religion, or other characteristic”. It can be expressed as speech, writing, gesture or display that attacks individuals because of the group they belong to. Some of the examples of hate speech are shown below in figure 1.

“Queers are an abomination and need to be helped to go straight to hell!”

“We have to kill all the Palestinians unless they are resigned to live here as slaves.”

“If you aren’t born here, pack your bags”

“Women shouldn’t talk sports on tv. They belong in the kitchen.”

Figure 1. Examples of Hate Speech

Over the years, there have been hundreds of incidents related to hate crimes that have taken place and have led to fights, riots and multitudes of casualties. Although authorities have tried to combat and contain this problem, by issuing laws or taking severe action against agitators, they haven’t been able to find any solid solutions to control or terminate this problem.

With widespread use of the Internet, large numbers of users take to various social media and online forums to express their opinions and thoughts on numerous subject matters. However,

the resulting drawback is the increasing amount of hate speech. Social media provides benefits such as anonymity which allows people misuse the freedom of speech to convey hatred towards others.

As this is a serious issue, various social media corporations such as YouTube, Instagram, Facebook and Twitter are continuously looking for ways to detect hate speech. Previously they relied on users to report such content. As artificial intelligence is on the rise, these companies took to machine learning techniques to optimize hate speech detection.

1.2 Machine Learning

Machine Learning is a methodology wherein computer systems make use of certain algorithms to parse the data, learn from it and utilize whatever it has learnt to perform specific tasks [2]. Machine learning algorithms find applications in a wide range of areas such as financial market analysis, recommendations, bioinformatics, fraud detection, malware classification and so on. These algorithms are categorized into three groups namely supervised learning, unsupervised learning and reinforcement learning.

Supervised learning deals with building models for data which consists of inputs and expected outputs, known as training data. These algorithms make use of labeled data. Examples include Decision Trees, Random Forests, Support Vector Machines (SVM), Logistic Regression and k-Nearest Neighbors. Unsupervised learning, on the other hand, deals with using non-labeled input data and find some structure in the data to get the desired output. Examples include clustering models such as k-Means, DBSCAN etc. In reinforcement learning, models try to maximize their rewards by using trial and error method.

Supervised Learning algorithms can be used to solve various problems that can be mainly categorized as classification and regression [3]. Classification deals with differentiating entities based on certain patterns or features of the entities. Some examples include image classification, text classification, handwriting analysis, face detection, spam detection etc. Regression deals with prediction of quantities of real-valued data. This project can be reduced to a problem of text classification as we focus on using the text derived from the audio content of videos to categorize the videos. For a given classification problem, the performances of the classifier models are different for different training datasets [4]. Although models such as Naïve Bayes and SVMs are known to be used for classification, the accuracy depends on the input data.

1.3 Deep Learning

Deep Learning is a subcategory of machine learning, where we build models to analyze data using a logical structure analogous to human thinking [5]. In deep learning, the data is passed through these layers of neural networks and each layer learns to transform the data into the desired output. Deep learning models are built on the basis of Artificial Neural Networks (ANNs), whose design is stimulated from the neural network of the human brain. An ANN with multiple intermediate layers is called a Deep Neural Network (DNN). Other types of neural networks include Convolutional Deep Neural Networks (CNNs), Recurrent Neural Networks (RNN) and Long Short-Term Memory (LSTM).

Neural Networks are some of the widely used models for classification and natural language processing (NLP) problems. Neural Networks work on the principle of using what they have learnt from input data to make predictions. A deep Neural Network consists of multiple layers of nodes, where each layer uses what it has learnt from the previous layer's output to train on discrete features. A node is a computational unit of a Neural Network layer and is similar to a neuron. There are many Neural Networks that are commonly used for classification. However, RNNs, in particular, are widely used to solve these problems as their recurrent nature can handle the variable length of the input data [6]. RNNs can predict the next word in a sequence by learning the context of the given sequence, which traditional Neural Networks are unable to do. Thus, RNNs are a better choice for text classification.

1.4 Problem Statement

Hate speech in social media could be expressed in the form of posts on Facebook and other online forums, tweets on Twitter, comments as well as videos on YouTube and so on. With the advantage of anonymity, users are able to create fake profiles and whole personas without giving out any personal identification. People make use of these accounts to spread violence, causing disturbances online and scam others. Cyber bullying is one of the major problems of social media as well. Some of the examples of online hate speech are shown below in figure 2. Although these social media platforms provide regulatory rules and laws, that if broken can result in suspension of the account, the problem of hate speech is still prevalent online and seems to be growing every day. These platforms have implemented detection algorithms using various machine learning

models and other frameworks to combat hate speech. However, these algorithms can be bypassed most of the time. Thus, there is an increasing need to find better solutions to solve this problem.

Lets kill jews and kill them for fun

[#killjews](#)

7/20/14, 8:05 AM

Dear black women,

When will y'all understand that I
don't hate y'all a lot of y'all just ugly as
shit compared to white girls

8/14/16, 3:22 AM

Trannies, your families will never love
you. You are living a lie & you know it.
End your miserable existence.
Commit suicide now.

Figure 2. Examples of Online Hate Speech

Most of the current hate detection methods focus more on textual data such as posts, comments or tweets. However, people can also make hateful videos and post them on video sharing sites. Video hosting services such as YouTube are powerful form of communication used by people all over the world. Aside from video content from music artists and other such professionals, people can upload video blogs about their daily life, video clips showing their

personal talents, reactions to other video content such as music and or movies and so on. Other users can view, like, share and comment on these uploaded videos. Thus, numerous users are giving their opinion on various topics. These opinions, while usually peaceful or offensive at times, can turn hateful at times.

Hate speech detection is a relatively new research area. With social media being used on a daily basis, the usage of hate speech has increased as well. Social media companies rely on users to report hateful content as well as manual filtering. However, this doesn't efficiently solve the problem as manual filtering of hate speech is costly and time consuming. Thus, researchers are determined to find better ways to detect hate speech. With major research in detecting hate speech in textual format, there is a need for a method to combat the hateful opinions presented in videos as well.

Detection of hate speech deals with identifying text and classifying it into hateful and not hateful speech. Current models also classify it into normal, offensive and hateful speech. This, however, is not an easy task as we need to differentiate between actual hate speech and general profanity. One observation is that someone speaking offensive language is not always hateful. For example, people might say words that are usually considered as offensive but might be used in a playful tone. Some of the techniques currently used to detect hate speech are Natural Language Processing, classifiers such as Deep Neural Networks, Deep Learning models such as Convolutional Neural Networks and Recurrent Neural Networks.

2. LITERATURE REVIEW

The purpose of this literature review is to focus on the usage of machine learning and deep learning techniques to resolve the problem of detecting hate speech in videos. The articles selected for this literature review include research papers, conference proceedings and journal articles.

Some of the earliest known works in detecting hateful online content is given by A. Razavi et al., [7] and Z. Xu et al., [8]. A. Razavi et al., [7] implemented an automatic flame detection model that makes use of multi-level classification to detect flames such as taunts, rants and squalid phrases in messages. They implemented a three-level classifier consisting of a Complement Naïve Bayes classifier, a Multinomial Updatable Naïve Bayes classifier and a Decision Table/Naive Bayes hybrid classifier along with Insulting or Abusive Language Dictionary (IALD). Z. Xu et al., [8] proposed an approach to implement an automatic sentence-level filtering approach to detect and remove offensive language from YouTube comments.

T. Davidson et al., [9] conducted research focusing on the separation of hateful speech and offensive language. They made use of a crowd-sourced hate speech lexicon to search hate speech keywords and classify tweets into hate speech, offensive language or neither. They experimented with several classification models such as Linear SVMs, Logistic Regression, Decision Trees and Random Forests. Results showed that racist and homophobic tweets were classified as hate speech, but sexist tweets were classified as offensive.

N. D. Gitari et al., [10] implemented an approach that uses sentiment analysis techniques to perform subjectivity detection to not only detect hate speech but also rate the polarity of the sentiment expressions. They generated lexicons related to hate speech using the semantic and subjective features to classify blog postings into not hateful, weakly hateful and strongly hateful classes. W. Warner et al., [11] proposed an approach to detect hate speech in online text using linear SVM classifier on Yahoo! News groups posts. They used the Parts of Speech tagging for each sentence to obtain the features used to train the model.

Research conducted by NDjuric et al., [12] concentrates on high-dimensionality and Sparsity issues that impact the current state-of-the-art detection systems. They proposed a two-step method for detecting hate speech. First, they used the paragraph2vec for joint modeling of comments and words, along with the continuous BOW (CBOW) neural language model. Thus, a low-dimensional text embedding is obtained, which is then used to train a binary classifier on Yahoo! Finance website comments dataset. C. Nobata et al., [13] implemented a machine learning

based approach to detect abusive language in online comments. They obtained the Yahoo! Finance and News comments for two different time periods and extracted different features from them. They experimented with several NLP features like lexicons, token N-grams, character n-grams, word2vec embedding and comment2vec embedding which were given to a supervised classification model.

F.D. Vigna et al., [14] proposed the first hate speech classifier focusing on Italian texts. They implemented a model to classify comments of public Italian Facebook pages into strong hate, weak hate and no hate categories. They used two different classifiers namely, SVM and LSTM with word embedding lexicons and sentiment polarity as the features obtaining effective results. Research conducted P. Badjatiya et al., [15] focused on using deep learning models to classify tweets as racist, sexist or neither. They used various tweet semantic embeddings such as char n-grams, word Term Frequency-Inverse Document Frequency (TF-IDF) values, Bag of Words and task-specific embeddings learned by the FastText, CNNs and LSTM models to train classifiers such as Gradient Boosted Decision Trees (GBDTs), Logistic Regression, Random Forest, SVMs, and DNNs. They obtained a F1 score ~18 points higher than the state-of-the-art methods. L. Gao et al., [16] proposed an approach wherein they detected hate speech in Fox News user comments by considering the context in which the comments were made. They trained two types of models namely Logistic Regression and a Neural Network consisting of Bi-directional LSTMs to obtain results that showed an increase of 3% – 4% in F1 score as compared to the existing baseline models.

S. Biere [17] researched about using Natural Language Processing (NLP) techniques to detect hate speech in tweets. They used a CNN to classify every tweet as hate, offensive language and neither classes. The tweets are preprocessed to get the word embeddings which are then given to the CNN model, resulting in an accuracy of about 91%. S. Malmasi et al., [18] implemented a hate speech detection model that classify text as hate, offensive and ok. They used surface n-grams and word-skip grams as features which are then passed to an SVM with LIBLINEAR kernel to train on hate speech detection dataset.

2.1 Classification of Videos

While there is a large amount of research undertaken to detect hate speech, most of it is focused on textual data such as comments, posts, blogs, tweets etc. Since videos can be used to spread hate speech as well, research needs to be conducted to find a way to detect hate speech in videos. Several approaches have been implemented to generally classify videos. However, there is very little research wherein videos are specifically classified as containing hate speech or not.

M.S. Barakat et al., [19] implemented a Dynamic Time Wrapping (DTW) based approach to detect offensive words in video blogs using the audio as the feature. They proposed a model that uses speaker independent keyword spotting that is applied to the audio content. They compared keyword templates with audio segments of the videos using the DTW algorithm to detect offensive words. This keyword spotting approach used on speech data to identify specific words that are spoken, was found to provide high accuracy. R. Kandakatla [20] proposed a framework to determine offensive YouTube videos. They made use of Naïve Bayes and SVM to detect if a video is offensive based on the content and metadata of the video such as title, description, number of views and comments. The models were trained on 300 videos and tested on 86 videos, which resulted in SVM providing the best performance. They concluded that the most offensive videos had the most negative comments.

S. Parameswaran et al., [21] have conducted in depth analysis on various machine learning techniques that can be used to classify videos. Based on the type of video data used, these techniques can be divided into four approaches namely – text based, audio based, visual based and combined approaches. The text-based approach uses the viewable text or the transcripts for classification. Bag of words and TF-IDF are the commonly used models for this approach. The audio-based approach deals with extracting the audio from the videos and classifying them based on time domain and frequency domain features. The video-based approach deals with classifying the videos based on visual features such as color, object, motion, shot etc. Various supervised learning classifiers such as HMM, Gaussian Mixture Model (GMM) and SVM are used for this approach. The combined approach deals with classifying videos using a combination of audio, text and visual features. CNN and RNN are the most commonly used models for this approach.

Huang J. et al., [22] provided some of the earliest work on classification of videos where they used HMM to classify based on audio and visual features. For every new feature discovered, a new HMM was built, thus, increasing the performance. M. J. Roach et al., [23] used motion

information such as the motion of the foreground object as well the motion of the background camera as feature given to a GMM for classifying videos. A. Karpathy et al., [24] implemented a CNN for large scale classification of 1 million YouTube videos belonging to 487 classes.

L. Kaushik et al., [25] implemented a system to detect sentiment from YouTube videos. They make use of a text-based sentiment model that use Maximum Entropy classifier and Parts of Speech (POS) tagged features. The raw text is first processed to get the POS tags. These features are then used to train a Maximum Entropy model. This, however, results in large amounts of redundant features. Thus, they employ an iterative feature reduction during training. Lastly, the sentiment models are used with Automatic Speech Recognition (ASR) to perform sentiment analysis on videos.

Kale. A et al., [26] develop a system that classifies videos using the embedded audio. The developed system is based on the client server architecture, where the server-side deals with extracting the audio content from the video and then they convert the audio into text using an APIs. This text is factorized to obtain keywords that are stored in a database. Thus, each video is categorized based on the keywords. In a client side a web application is implemented, where a user can search and play their desired videos. When a user wishes to search for a video, they enter a keyword. This keyword is then searched among the list of keywords stored in the database to retrieve the desired video.

Through various research materials, it is possible to identify the most suitable machine learning models as well the best features to be used to give the best results, when detecting hate speech.

3. METHODOLOGY

In this section, we discuss the approach used to solve the problem of detecting hate speech in videos. The main goal of this project is to implement a framework to detect hate speech in the spoken content of videos as shown below in figure 3.

Figure 3. Hate Speech Detection Framework

The entire process can be divided into three main parts:

1. Build the video dataset.
2. Extract audio from the video dataset and convert into textual format.
3. Train machine learning models over the dataset and classify videos as normal or hateful.

3.1 Dataset Creation

There are many video datasets available. However, there is no particular dataset available that would be suitable for this project. Thus, the dataset was manually collected. We considered YouTube as the primary source since it is one of the most popular video sharing websites. This project focuses on what is being said in the videos rather the images displayed in the videos or the comments posted under the videos. Videos containing normal speech as well as videos containing offensive terms were selected to form the dataset. There are different categories of offensive videos. For this project, we focused on videos with racist and sexist speech. Examples of offensive videos are shown below in figure 4.

Mel Gibson's Explosive Racist and Sexist Rant

**Racist White Teen Girls Goes
On A Rant About Blacks**

Figure 4. Examples of Offensive Videos

Figure 5. Dataset Creation Module

To construct the dataset, we have implemented a crawler that searches for videos on YouTube and downloads them. To help in searching for the videos, we used the YouTube Data API [27]. YouTube provides developer with tools and resources such as the YouTube Data API to access YouTube video and channel data and video statistics. It contains libraries in various languages that can be used to integrate YouTube data into websites or applications. It provides users with libraries to search, delete, upload video content and so on. The API provides the search by keyword feature which searches and returns the videos whose video title, channel name or description contain the given keyword. We used multiple offensive words such as racist rants, racial slur, sexist comments, sexism and so on, as keyword to the API which would return the video title as well as the unique video id. Since the YouTube Data API does not have any provisions to download the videos, the Pytube library [28] was used to download videos in mp4 format. Pytube is a python-based library which is specifically used to download videos from YouTube. The unique video ids generated, are passed to the downloader function and the video is downloaded. Each of the videos is manually classified as normal, racist or sexist as shown in Table 1.

TABLE 1
CLASSIFICATION OF VIDEOS

Video	Category
https://youtu.be/Ugl44YO5nw4	Normal
https://youtu.be/CAR2h5aSQQ4	Racist
https://youtu.be/x9RhW-_QrmY	Sexist

3.2 Dataset Processing

Figure 6. Dataset Processing Model

Since this project deals with what is being said in the video, we need to extract the audio content from the video. This is done by using the FFmpeg API [29]. The FFmpeg API is a multimedia framework that allows users to encode, decode and convert media between different formats. Using this API, we can convert the videos into any audio format. We are primarily focusing on text-based features to train the machine learning models. Thus, once the audio is

extracted, it must be converted into textual format. This can be done using the Speech-to-Text conversion APIs and frameworks that are readily available. For the purposed of our project, we use the Google Cloud Speech-to-Text API. The Google Cloud Speech-to-Text API makes use of Neural Network models that enables users to convert audio files into textual scripts [30]. To use this API, one must acquire the correct credentials and authorization key. The API also required the audio files to be of FLAC or LINEAR format. Thus, the mp4 videos are initially converted to FLAC format. Since the API requires the audio files to have only mono channel, these FLAC formatted audios with stereo channel are converted into mono channel. The API works well on single speaker audios with less background noise. An example of the Speech-to-Text conversion result is shown below in Table 2.

The first clip contains very clear audio and is spoken by a single speaker. The resulting text is very clear and grammatically correct. The second clip contains a lot of background noise and multiple speakers. Thus, the resulting text is not clear and does not make sense, which would not be suitable for detection purpose.

TABLE 2
EXAMPLE RESULTS OF SPEECH-TO-TEXT CONVERSION

Video Clip	Converted Text
https://www.youtube.com/watch?v=Ugl44YO5nw4	this weekend for the first time when I'm not writing and kind of see how it goes there is caffeine in it so I just need to be very careful with that so I'm going to give it a try but if it stimulates me too much it's like game over but not Sarah okay so are on the podcast on Sunday you can find the no sugarcoating podcast recording into one word we talked about keto for children I think it was this week or am I getting my things messed up YouTube video on YouTube for kids to be on keto if I had children I would encourage them to eat low-carb high-fat
https://www.youtube.com/watch?v=KPuDkz8TApA	how can I get him a message stop. At the movies latest movies latest

Once the videos are converted to the required audio format, the videos are required to be uploaded to Google Storage Bucket before being passed to the Speech-to-Text API if the length of the audio files exceeds one minute. For simplicity of conversion, all videos are uploaded to the Google Storage Bucket before running the Speech-to-Text API. The API returns the converted text in the form of a transcript which is stored in a document for future experimentation. A snippet of the Speech-to-Text API is shown below in figure 7.

```
client = speech.SpeechClient()

for file_name in mylist:
    if(file_name):
        gcs_uri = "gs://videos_upload/v1/"+ file_name
        audio = types.RecognitionAudio(uri=gcs_uri)
        config = types.RecognitionConfig(
            encoding=enums.RecognitionConfig.AudioEncoding.FLAC,
            #sample_rate_hertz=48000,
            language_code='en-US')

        # Detects speech in the audio file
        operation = client.long_running_recognize(config, audio)
        response = operation.result(timeout=90)

        #print transcript
        output = file_name+":"

        for result in response.results:
            output += result.alternatives[0].transcript
        print output
```

Figure 7. Snippet of Speech-to-Text Code

3.3 Model Training

Figure 8. Model Training Module

3.3.1 Feature Engineering

The resulting transcript from the Google Cloud Speech-to-Text API is further formatted to remove stop words and convert the text into lower case using NLTK libraries. Since the dataset is in textual format, it needs to be further processed before being passed to the machine learning models. A simple technique to convert text into numbers for machine learning models would be the Bag of Words method wherein each word is assigned a unique number. For this purpose, the CountVectorizer and TfidfVectorizer libraries are used [31]. The CountVectorizer converts the input text into tokens by computing the word counts of all the words in the input text and uses it as a vocabulary to translate other text documents. While word counts work well, the better option would be to count the frequencies of each word in the text document. This can be done by using the TfidfTransformer. It works similar to the TfidfVectorizer library where TFIDF stands for Term Frequency Inverse Document Frequency. Term Frequency indicates the word counts of all words in the input document and Inverse Document Frequencies are computed for each word in the input

wherein the most frequently occurring words will be assigned a lower score and the least occurring words will be assigned a higher score. This `TfidfTransformer` returns a normalized vector wherein the highest score is 0 and lowest score is 1. Using TFIDF, the input text is converted into tokens and used to create a vocabulary which is then used to encode other text documents. Another method to encode text as numbers would be to compute the n-grams of the words in the data set. N-grams refers to the sequences of n objects derived from a given text. Using `TfidfVectorizer`, we can specify the value of n to determine the number of words in a gram sequence. If n is equal to 1, each sequence contains one word and is known as Uni-grams. If n is equal to 2, each sequence contains pairs of words and is known as Bi-grams and so on. We then build a vocabulary consisting of these Uni-gram and Bi-gram objects and compute the Inverse Document Frequency of all the objects in the vocabulary and obtain the normalized scores.

Consider the following example:

The dog ran and jumped into the water.

Using `CountVectorizer` we get:

```
['the': 5, 'dog': 1, 'ran': 4, 'and': 0, 'jumped': 3, 'into': 2,
 'water': 6]
```

The vectorized value are: [1 1 1 1 1 2 1]

Here `CountVectorizer` assigns a unique ID for each word in the sentence. It returns an encoded vector which contains the counts of all the words in the sentence. In the example sentence, the word “the” occurs twice. Thus, the encoded vector shows a count of 2 for the word “the” and count of 1 for all the other words.

Using `TfidfTransformer` we get:

```
['the': 5, 'dog': 1, 'ran': 4, 'and': 0, 'jumped': 3, 'into': 2,
 'water': 6]
```

The vectorized value are: [0.31622777 0.31622777 0.31622777 0.31622777
0.31622777 0.63245553 0.31622777]

The word counts generated by `CountVectorizer` are then passed to the `TfidfTransformer` to get an encoded vector that contains the normalized frequency scores. Thus, for the example sentence, the word “the” is assigned a lower score of 0.63245553 and all the other words are assigned a higher score of 0.31622777.

Using n-grams to get Uni-grams and Bi-grams we get:

```
['the': 10, 'dog': 2, 'ran': 8, 'and': 0, 'jumped': 6, 'into': 4,
 'water': 13, 'the dog': 11, 'dog ran': 3, 'ran and': 9, 'and
 jumped': 1, 'jumped into': 7, 'into the': 5, 'the water': 12]
```

Uni-grams: ['the', 'dog', 'ran', 'and', 'jumped', 'into', 'water']

Bi-grams: ['the dog', 'dog ran', 'ran and', 'and jumped', 'jumped into', 'into the', 'the water']

The vectorized value are: [0.24253563 0.24253563 0.24253563 0.24253563
0.24253563 0.24253563 0.24253563 0.24253563 0.24253563 0.24253563
0.48507125 0.24253563 0.24253563 0.24253563]

Here, the TfidfVectorizer generates a vocabulary that consists of Uni-gram and Bi-gram objects and assigns a unique ID for each n-gram object generated from the sentence. It then computes the counts of these objects as they appear in the input sentence and then generates the encoded vector that contains the normalized frequency scores. Thus, for the example sentence, the object “the” is the only object in the vocabulary that occurs twice in the whole sentence and thus, is assigned a lower score of 0.48507125. All the other objects occur only once in the sentence and thus, are assigned a higher score of 0.24253563.

This project uses a combination of CountVectorizer and TfidfTransformer to determine frequencies of word counts as well as uses TfidfVectorizer to compute Uni-grams and Bi-grams of the transcript, which is passed as input to the machine learning models. Experiments are conducted using the feature sets to determine which feature would provide the best classification results.

3.3.2 Models

Various machine learning, and deep learning models have been used to tackle the problem of hate speech detection, as given in the literature review. Based on the research conducted, we have implemented Naïve Bayes Classifier, Random Forrest Classifier, Linear Support Vector Machines (SVM) model and Recurrent Neural Network (RNN) model.

Naïve Bayes classifiers are used for classifying entities made up of discrete features such as word counts for text classification. For this project, we made use of a type of Naïve Bayes model known as Multinomial Naïve Bayes classifier, which is mostly used for text classification. Naïve Bayes works on the Bayes Theorem of conditional probability [32]. The formula is given by

$$P(A | B) = \frac{P(B | A) P(A)}{P(B)}$$

where $P(A | B)$ is the probability that an event A would occur given that event B has occurred. The classifier computes the probability for every outcome and determines the outcome with the highest probability.

Random Forrest Classifier is an ensemble learning method that makes use of multiple decision trees for classification and regression [4]. These classifiers aggregate the results of the decision trees to improve the overall accuracy. A basic architecture of a Random Forrest Classifier is shown below in figure 9. For a given data input, a decision tree is built for every sample in the dataset. The prediction from each of the trees is voted up on and the prediction with the highest votes is determined as the end prediction [33]. We have built a classifier with 1024 trees.

Figure 9. Random Forrest Classifier [33]

Support Vector Machine is a supervised machine learning model that can be used to solve classification and regression problems [4]. These models are discriminative in that they are able to distinguish between entities into their respective classes without explicit knowledge of these classes. These models make use of different kernels such as linear, rbf and so on, to transform the data before separating them based on the labels generated. For our project we have implemented

Linear SVM. A Linear SVM separates the data linearly by determining the best suitable hyperplane between the categories of data. A simple graph of an SVM model is shown below in figure 10.

Figure 10. A Simple Diagram of SVM

Recurrent Neural Networks (RNN) are a kind of artificial neural networks that utilize their internal memory to process sequences of input data [6]. A Simple RNN model is shown below in figure 11. Given an input, a RNN model remember significant points of the input and uses what it has learnt to make future predictions. In a RNN, the data is looped back as it considers the current data input as well what it has learnt from the previous data inputs which is stored in its memory. For our project, we used Keras with Tensorflow as backend to implement the RNN model [34]. We experimented with different layers and determined the accuracy and used ‘relu’ activation function and ‘adam’ optimizer with ‘bianry_crossentropy’ as the loss function.

Figure 11. A Simple Recurrent Neural Network [34]

3.4 Sentiment Analysis

Sentiment Analysis is the process of classifying entities into positive, negative or neutral categories using Natural Language Processing. In this project, we use the TextBlob python library [35] to perform Sentiment Analysis on the dataset. Given an input, TextBlob generates the polarity and subjectivity for that input. Polarity determines the sentiment of the input and it ranges from -1.0 to 1.0, with 1.0 being positive sentiment, -1.0 being negative sentiment and 0.0 being neutral sentiment.

TextBlob contains a lexicon that contains scores such as polarity, subjectivity, intensity etc., for certain words. TextBlob computes the polarity of an input text by computing the average polarity of individual words that are in its lexicon. Examples of sentiment analysis are given below in Table 3.

TABLE 3
EXAMPLE OF SENTIMENT ANALYSIS

Sentence	Polarity	Sentiment
holy sh*t that's great news	0.3	Positive
go back to your country you morons	-0.4	Negative

Consider the following example:

not very good

Polarity of good: 0.7

Polarity of not: 0.0

Polarity of very: 0.2

However, for negative words, TextBlob multiplies the polarity with -0.5 .

The word `very` is called as modifier which affects the polarity of the next word i.e., `good`. Thus, we multiply the overall polarity with inverse of the intensity score of `very` i.e., $\frac{1}{1.3}$

$$\text{Total polarity} = 0.7 * \frac{1}{1.3} * (-0.5) = -0.26923076923076916$$

Sentiment analysis was performed on our transcript dataset to obtain the polarity of each sentence and test if this analysis alone would be enough to classify the videos. However, we observed that some of the transcripts were being incorrectly classified.

4. EVALUATIONS AND RESULTS

In this section, we discuss the experiments conducted using the compiled dataset, evaluating the models and analyze the results obtained.

4.1 Experiments

Using the implemented crawler, we were able to compile a dataset which consisted of 300 videos, of which 150 were non-offensive videos, 85 racist videos and 65 sexist videos. The distribution of videos in the dataset is shown below in figure12 and figure 13. These videos were split to ensure a uniform length for easy processing. These videos are converted into FLAC format with mono channel as the Google Cloud Speech-to-Text API requires this format for processing the audios. Once the videos are converted into audio format, they are uploaded to the Google Storage Bucket. The Google Cloud Speech-to-Text API has separate functions to transcribe short files and long files respectively. Shorter files can be transcribed locally whereas files longer than 1 minute are required to be stored in the Google Storage Bucket. Once all files are uploaded, the Google Cloud Speech-to-Text API is executed to convert the audio files to get the resulting transcript.

Figure 12. Dataset Distribution for Two Labels

Figure 13. Dataset Distribution for Three Labels

Two different kinds of experiments were conducted. The first experiment dealt with classifying the videos into normal or hateful videos. The second experiment dealt with classifying the videos into normal, racist or sexist videos. The dataset was split into training and testing sets in the ratio 70:30, with 70% being training data and 30% being testing data before the feature extraction process. For feature extraction, the word counts, frequency of the word counts as well as n-grams are extracted from the data set. Experiments were conducted using these three different features using various models and the evaluation metrics were computed.

Sentiment Analysis is performed on the entire dataset and is classified into negative, positive or neutral categories. We observed that the normal videos were classified as positive where the some of the hate videos were classified as positive or neutral.

4.2 Evaluation Metrics

To evaluate the classifier models, we used metrics such as accuracy of the models, precision score, recall score and F1 score [36]. These metrics are commonly used to evaluate the performance of a model and can be determined using parameters obtained from a confusion matrix. A confusion matrix can be described as a table that illustrates the classification of actual data vs predicted data. An example of a confusion matrix for a binary classifier is shown below on Table 4.

TABLE 4
CONFUSION MATRIX OF A BINARY CLASSIFIER

	Predicted Data		
Actual Data		Offensive	Normal
	Offensive	True Positive	False Negative
	Normal	False Positive	True Negative

A confusion matrix consists of four parameters namely True Positives, True Negatives, False Positives and False Negatives [37]. True Positives specify the offensive videos that have been correctly classified as offensive. True Negatives specify the normal videos that have been correctly classified as normal. False Positives specify the normal videos that have been incorrectly classified as offensive. False Negatives specify the offensive videos that have been incorrectly classified as normal. For the performance of a model to be high, the False Positives and False Negatives need to be minimized.

Precision score can be defined as the percentage of number of correctly classified videos with respect to the total number of predicted videos. It is given by

$$\text{Precision Score} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}$$

Recall score can be defined as the percentage of correctly classified videos with respect to the total number of videos in that class. It is given by

$$\text{Recall Score} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}}$$

Accuracy can be defined as the percentage of number of correctly classified videos with respect to the actual number of videos. It is given by

$$\text{Accuracy} = \frac{\text{True Positive} + \text{True Negative}}{\text{Total Observations}}$$

F1 score is a combination of both precision score and recall score that is used to measure the overall accuracy of a model. The higher the F1 score value, the lower the False Positive and False Negative values. An F1 score of 1 means that the model is close to being ideal. It is given by

$$\text{F1 Score} = 2 \times \frac{\text{Precision score} \times \text{Recall score}}{\text{Precision score} + \text{Recall score}}$$

4.3 Results

The average results for each of the metrics for both experiments are shown below in Table 5 and Table 6. From the table, we can infer that Random Forrest Classifier model provides a comparatively better classification than the other models. Up on observation, we can infer that the usage of frequency of word counts as features provide better classification results compared to the other feature sets.

TABLE 5
AVERAGE RESULTS OF EXPERIMENT 1

	Accuracy	Precision Score	Recall Score	F1 Score
Multinomial Naïve Bayes	0.8512	0.8833	0.8533	0.8267
Linear SVM	0.8929	0.9133	0.8933	0.89
Random Forrest	0.9464	0.95	0.9467	0.9433
RNN	0.8036	0.65	0.80	0.72

TABLE 6
AVERAGE RESULTS OF EXPERIMENT 2

	Accuracy	Precision Score	Recall Score	F1 Score
Multinomial Naïve Bayes	0.7976	0.78	0.7967	0.7567
Linear SVM	0.8214	0.7933	0.8233	0.7967
Random Forrest	0.8571	0.7733	0.86	0.81
RNN	0.8036	0.65	0.80	0.72

The average values for each of the metrics is computed for all four models and is plotted in a bar graph using the Matplotlib Python library as shown in the figures below.

Figure 14. Average Accuracy of Models Given Two Labels

Figure 15. Average Accuracy of Models Given Three Labels

Figure 16. Average Precision Score of Models Given Two Labels

Figure 17. Average Precision Score of Models Given Three Labels

Figure 18. Average Recall Score of Models Given Two Labels

Figure 19. Average Recall Score of Models Given Three Labels

Figure 20. Average F1 Score of Models Given Two Labels

Figure 21. Average F1 Score of Models Given Three Labels

In addition, we also plotted the Receiver Operating Characteristics (ROC) curves as well computed the Area Under the Curve (AUC) for each of the models, generated when classifying for two labels, as shown below in figure 22, figure 23, figure 24 and figure 25. ROC curves are used to evaluate how a binary classifier is able to categorize entities into their respective classes. An AUC of 1 indicates that the model is able to completely differentiate between the two categories. When comparing our results, Multinomial Naïve Bayes and Linear SVM models generate an AUC of 1 whereas Random Forrest generates an AUC of 0.97 and RNN generates an AUC of 0.50. These results indicate that RNN model is not able to classify our data into their respective categories. While the models generate an AUC of 0.99 and 1, which is ideal, we also need to consider the possibility of overfitting due to the limited dataset.

Figure 22. ROC Curve for Multinomial Naïve Bayes

Figure 23. ROC Curve for Linear SVM

Figure 24. ROC Curve for Random Forrest Classifier

Figure 25. ROC Curve for RNN

In this project, three different feature sets were used to train the classifier models. The evaluation metrics for all the models are aggregated with respect to each feature set and plotted in a bar graph to provide a comparison of the feature sets as shown in figure 26 and figure 27. These graphs indicate that using the frequency of word counts of a text generates better classification as compared to the other two feature sets.

Figure 26. Feature set Comparison for Two Labels

Figure 27. Feature set Comparison for Three Labels

4.4 Comparison with Existing Approaches

The research conducted by M.S. Barakat et al., [14] focuses on detecting offensive videos based on the spoken content of videos. They implemented an approach to detect certain keywords in the spoken content with minimal training and language information using Dynamic Time Wrapping (DTW). We compared the average precision and recall scores obtained from their experiments with the average scores of our project. For simplicity of comparison, we have considered the average scores for classification of two labels. The comparison indicates that our approach provides better results in terms of average precision and recall scores as shown below in figure 28 and figure 29.

Figure 28. Comparison of Average Precision Scores

Figure 29. Comparison of Average Recall Scores

The research conducted by R. Kandakatla [15] deals using Naïve Bayes and SVM models to detect offensive videos based on the metadata content of the video such as description, likes, comments and so on. They made use of comment-based features and metadata-based features to conduct the experiments and computed the precision score, recall score and f1 scores for both models. We compared these scores obtained for Naïve Bayes and SVM models for their approach with our approach. For simplicity of comparison, we have considered the average scores for

classification of two labels. The comparison indicates that our approach provides better results in terms of precision, recall and f1 scores as shown below in figure 30, figure 31 and figure 32.

Figure 30. Comparison of Precision Scores

Figure 31. Comparison of Recall Scores

Figure 32. Comparison of F1 Scores

5. CONCLUSION

Hate speech detection has become an interesting domain for research. With social media platforms providing users with benefits such as anonymity, users are able to express their hateful opinions. Thus, there is a need for optimal hate speech detection system. As more people are turning towards video sharing sites, people tend to post opinionated videos which might not always be peaceful. The existing hate speech detection methods focus on text data. Hence, there is a need to find an optimal approach to detect hate speech in videos. The current methods employ various machine learning techniques to detect hate speech to provide fairly good results. By applying the same machine learning techniques, there is a possibility to detect hateful speech in videos.

The approach used in this project deals with converting the video into text format before passing it as input to machine learning models. Various machine learning models are trained and evaluated to compute the evaluation metrics such as accuracy, precision score, recall score and f1 score to determine the best working model for this data. The results indicate that Random Forrest Classifier model provided the best results with an accuracy of 96 %.

This project focuses on classifying videos into two or three labels. For future work, it can also be extended to classifying more than three categories as well as increasing the size of the dataset for better classification. This project makes use of Google Speech-to-Text API to generate transcripts of the spoken content. Alternatively, other APIs or speech-to-text generators could be considered as there may be a possibility of obtaining better translations. Another future work would be to include the tone of the speaker to understand the context in which the speech was expressed which might provide improved detection of hate speech.

REFERENCES

- [1] John T. Nockleby, "Hate Speech", In Encyclopedia of the American Constitution (Macmillan), 2nd edition, 2002.
- [2] K. Murphy, Machine Learning: A Probabilistic Perspective, MIT, 2012.
- [3] G. Ratsch, "A Brief Introduction into Machine Learning", 2004.
- [4] A. Heydarzadegan et al, "Evaluation of Machine Learning Algorithms in Artificial Intelligence", International Journal of Computer Science and Mobile Computing (IJCSMC), Vol.4 Issue.5, pg. 278-286, May 2015.
- [5] M. A. Nielsen, Neural Networks and Deep Learning, Determination Press, 2015.
- [6] A. Sherstinsky, "Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network", Nov 2018, arXiv:1808.03314v4
- [7] A. Razavi, D. Inkpen, S. Uritsky, S. Matwin, "Offensive Language Detection Using Multi-Level Classification", In Advances in Artificial Intelligence Springer, pp. 1627, 201
- [8] Z. Xu, S. Zhu, "Filtering Offensive Language in Online Communities using Grammatical Relations, in CEAS 2010 - Seventh Annual Collaboration, Electronic messaging, AntiAbuse and Spam Conference July 13-14, Redmond, Washington, US 201
- [9] T. Davidson, D. Warmley, M. W. Macy, I. Weber, "Automated hate speech detection and the problem of offensive language", Proc. 11th Int. Conf. Web Social Media, pp. 512-515, 2017.
- [10] N. D. Gitari, Z. Zuping, H. Damien and J. Long, "A lexicon-based approach for hate speech detection", in of International Journal of Multimedia and Ubiquitous Engineering, 10(4):215–230, 2015.
- [11] W. Warner and H. Julia, "Detecting Hate Speech on the World Wide Web", in Proceedings of the Workshop on Language in Social Media, Association for Computational Linguistics (ACL), pp. 19-26, 2012.
- [12] N. Djuric, J. Zhou, R. Morris, M. Grbovic, V. Radosavljevic, N. Bhamidipati, "Hate Speech Detection with Comment Embeddings", Proceedings of the 24th International Conference on World Wide Web, May 18-22, 2015, Florence, Italy doi>11145/2740908.274276

- [13] C. Nobata, J. Tetreault, A. Thomas, Y. Mehdad, Y. Chang, “Abusive language detection in online user content”, In WWW, 145–153, 2016.
- [14] F. D. Vigna, A. Cimino, F. Dell’Orletta, M. Petrocchi, M. Tesconi, “Hate me, hate me not: Hate speech detection on facebook”, In Proceedings of the First Italian Conference on Cybersecurity, pages 86 – 95, 2017.
- [15] P. Badjatiya, S. Gupta, M. Gupta, V. Varma, “Deep Learning for Hate Speech Detection in Tweets”, in Proceedings of the 26th International Conference on World Wide Web Companion, April 03-07, 2017, Perth, Australia [doi>11145/3041021.3054223]
- [16] L. Gao, R. Huang, “Detecting online hate speech using context aware models”, In Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2017, pages 260–266. INCOMA Ltd., 2017.
- [17] S. Biere, “Hate Speech Detection Using Natural Language Processing Techniques”, August 2018.
- [18] S. Malmasi, M. Zampieri, “Detecting Hate Speech in Social Media”, In Proceedings of the International Conference Recent Advances in Natural Language Processing (RANLP), pages 467–472, 2017.
- [19] M.S.Barakat, C.H.Ritz, and D.A.Stirling, “Detecting Offensive User Video Blogs: An Adaptive Keyword Spotting Approach”, in proc. of ICALIP, pp. 419-425, 2012.
- [20] R. Kandakatla, “Identifying Offensive Videos on YouTube”, 2016.
- [21] S. Parameswaran, Dr. S. Joseph, “A Review of Machine Learning Techniques used for Video Classification”, in International Journal of Current Engineering and Scientific Research (IJCESR), Vol. 4, issue 12, 2017.
- [22] Huang, J., Liu, Z., Wang, Y., Chen, Y., Wong, E. K., “Integration of multimodal features for video scene classification based on HMM”, Multimedia Signal Processing, 1999 IEEE 3rd Workshop on (pp. 53-58), 1999.
- [23] Roach, M. J., Mason, J. D., Pawlewski, M, “Video genre classification using dynamics”, In Acoustics, Speech, and Signal Processing, in Proceedings of IEEE International Conference (ICASSP’01), Vol. 3, pp. 1557-1560, 2001.
- [24] Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., Fei-Fei. L, “Large-scale video classification with convolutional neural networks”, In Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, pp. 1725-1732, 2014.

- [25] L. Kaushik, A. Sangwan and J.H.L. Hansen, “Automatic sentiment extraction from YouTube videos,” in IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU), pp.239-244, 2013.
- [26] Kale, A. and Wakde, D.G., “Video Retrieval Using Automatically Extracted Audio”, in proc. IEEE International Conference on Cloud & Ubiquitous Computing & Emerging Technologies (CUBE), DOI:11109/CUBE.2013.32, Page(s): 133 - 136, 2013
- [27] YouTube Data API, 2019. [Online]. Available at: <https://developers.google.com/youtube/v3>. [Accessed: 04 February 2019].
- [28] Pytube, 2019. [Online]. Available at: <https://python-pytube.readthedocs.io/en/latest>. [Accessed: 04 February 2019].
- [29] FFmpeg, 2019. [Online]. Available at: <https://ffmpeg.org>. [Accessed: 04 February 2019].
- [30] Google Cloud Speech-to-Text, 2019. [Online]. Available at: <https://cloud.google.com/speech-to-text>. [Accessed: 04 February 2019].
- [31] J. Brownlee, “How to Prepare Text Data for Machine Learning with scikit-learn”, 2017. [Online]. Available at: <https://machinelearningmastery.com/prepare-text-data-machine-learning-scikit-learn>. [Accessed: 10 April 2019].
- [32] R. Saxena, “How the Naïve Bayes Classifier Works in Machine Learning”, 2017. [Online]. Available at: <http://dataaspirant.com/2017/02/06/naive-bayes-classifier-machine-learning>. [Accessed: 10 April 2019].
- [33] A. Navlani, “Understanding Random Forests Classifiers in Python”, 2019. [Online]. Available at: <https://www.datacamp.com/community/tutorials/random-forests-classifier-python>. [Accessed: 10 April 2019].
- [34] “Recurrent Neural Networks”, 2019. [Online]. Available at: <https://machinelearning-blog.com/2018/02/21/recurrent-neural-networks>. [Accessed: 10 April 2019].
- [35] TextBlob, 2019. [Online]. Available at: <https://textblob.readthedocs.io/en/dev>. [Accessed: 04 February 2019].
- [36] C. Goutte, É. Gaussier. “A Probabilistic Interpretation of Precision Recall and F-Score, with Implication for Evaluation”, ECIR, 2015.
- [37] R. Joshi, “Accuracy, Precision, Recall & F1 Score: Interpretation of Performance Measures”, 2016. [Online]. Available at: <https://blog.exsilio.com/all/accuracy-precision-recall-f1-score-interpretation-of-performance-measures/>. [Accessed: 10 April 2019].

APPENDIX

Source Code

The source code for this project can be viewed at <https://github.com/unnathi10>.

Experimental Setup

This project makes use of Virtualenv to create a virtual Python environment with Anaconda distribution and Jupyter Notebook. The Virtualenv also contains Keras installation with Tensorflow backend to execute deep learning models.

A step by step guide to install Keras is available at:

<https://www.tensorflow.org/install/pip>

<https://keras.io/>

APIs

Since this project makes use of the YouTube Data API and the Google Cloud Speech-to-Text API, the code needs access credentials and authorization to use these APIs. A complete guide to using Google Cloud Platform is available at:

<https://cloud.google.com/docs/>

The project also requires access to Google Storage Bucket via the Google Cloud Platform.

A step by step guide to create and Google Storage Bucket is available at:

<https://cloud.google.com/storage/docs/creating-buckets>