2020 International Conference on Computational Science and Computational Intelligence (CSCI) | 978-1-7281-7624-6/20/$31.00 ©2020 IEEE | DOI: 10.1109/CSCI51800.2020.00104

Detection of Hate Speech in Videos Using Machine Learning

Ching Seh Wu
Department of Computer Science
San Jose State University
San Jose, CA USA
ching-seh.wu@sjsu.edu

Unnathi Bhandary
Department of Computer Science
San Jose State University
San Jose, CA USA
unnathi.bhandary@sjsu.edu

***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 research 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.**

***Index Terms*—Hate speech, machine learning, deep learning.**

I. Introduction

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. It can be expressed as speech, writing, gesture or display that attacks individuals because of the group they belong to.

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. 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. 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.

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. 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.

The goal of this research is to detect hate speech in videos using machine learning and deep learning algorithms based on the spoken content of the videos. We also compared the performance of different algorithms to obtain the optimal algorithm for our approach.

This paper is organized as follows: First, a brief overview of a few relevant, recent research done in the space of hate speech detection will be discussed. Second, we will present the methodology used to implement our approach. Third, we discuss the experiments and evaluation metrics used to evaluate our approach. Finally, we present the results obtained.

II. Overview

A. Related Work

Over the years, many solutions have been proposed to solve the problem of detecting hate speech. These solutions have been implemented using various machine learning and deep learning algorithms.

In [1] the authors implement an automatic flame detection model that makes use of multi-level classification to detect flames such as taunts, rants and squalid phrases in messages.

978-1-7281-7624-6/20/$31.00 ©2020 IEEE
DOI 10.1109/CSCI51800.2020.00104

The authors implement 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).

In [2] the authors implement 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. The authors generate lexicons related to hate speech using the semantic and subjective features to classify blog postings into not hateful, weakly hateful and strongly hateful classes.

In [3] the authors propose an approach that makes use of Natural Language Processing (NLP) techniques to detect hate speech in tweets. The authors have 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%.

In [4] the authors propose an approach to detect hate speech in online text using linear SVM classifier on Yahoo! News groups posts. The authors have used the Parts of Speech tagging for each sentence to obtain the features used to train the model.

B. *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.

In [5] the authors implement a Dynamic Time Wrapping (DTW) based approach to detect offensive words in video blogs using the audio as the feature. The authors propose a model that uses speaker independent keyword spotting that is applied to the audio content. The authors compare keyword templates with audio segments of the videos using the DTW algorithm to detect offensive words.

In [6] the author proposes a framework to determine offensive YouTube videos. The author has 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.

Based on the research we conducted, many of the approaches used machine learning algorithms such as SVMs, Naïve Bayes, neural networks and so on to detect hate speech.

III. METHODOLOGY

We propose a hate speech detection system that focuses on spoken content of the videos. This system is designed using the following steps:

1. Extract audio from video
2. Convert audio into text format
3. Train machine learning models over text-based features and classify videos as normal or hateful

A. *Dataset*

Since there were no suitable video datasets, the dataset was manually collected. We considered YouTube as the primary source since it is one of the most popular video sharing websites. 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 our dataset, we focused on videos with racist and sexist speech.

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 [7]. YouTube Data API contains 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 keywords 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 [8] 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/CAR2h5aSQO4	Racist
https://youtu.be/x9RhW-_QrmY	Sexist

B. *Dataset Processing*

To extract the audio content from the videos, we make use of the FFmpeg API [9]. 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 purpose of our approach, 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 [10]. The API requires 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.

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.

C. Sentiment Analysis

Sentiment Analysis is the process of classifying entities into positive, negative or neutral categories using Natural Language Processing. In this research, we use the TextBlob python library [11] 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.

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.

D. 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. For this purpose, the CountVectorizer and TfidfVectorizer libraries are used [12].

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.

Our approach 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.

E. 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. In this research, we made use of Multinomial Naïve Bayes classifier, which is mostly used for text classification. Naïve Bayes works on the Bayes Theorem of conditional probability [13]. Figure. 1 gives the formula for Bayes Theorem.

$$P(A \mid B) = \frac{P(B \mid A)\, P(A)}{P(B)}$$

Figure.1 Bayes Theorem of Conditional Probability

where $P(A \mid B)$ 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 [14]. These classifiers aggregate the results of the decision trees to improve the overall accuracy. 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 [15]. We have built a classifier with 1024 trees.

Support Vector Machine is a supervised machine learning model that can be used to solve classification and regression problems [14]. 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. In this paper, we have implemented Linear SVM. A Linear SVM separates the data linearly by determining the best suitable hyperplane between the categories of data.

Recurrent Neural Networks (RNN) are a kind of artificial neural networks that utilize their internal memory to process sequences of input data [16]. 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. We used Keras with Tensorflow as backend to implement the RNN model [17]. We experimented with different layers and determined the accuracy and used 'relu' activation function and 'adam' optimizer with 'bianry_crossentropy' as the loss function.

IV. EXPERIMENTS

In this section we describe the experiments conducted using the compiled dataset and the evaluation metrics used to compare the performance of our models.

A. Experiments

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. We compiled a dataset which consisted of 300 videos, of which 150 were non-offensive videos, 85 racist videos and 65 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.

B. Evaluation Metrics

To evaluate the classifier models, we used metrics such as accuracy of the models, precision score, recall score and F1 score [18]. These metrices 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.

A confusion matrix consists of four parameters namely True Positives, True Negatives, False Positives and False Negatives [19]. 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.

V. RESULTS

A. Evaluation Results

The average results for each of the metrics for both experiments are shown in Table 2 and Table 3. From the tables, 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 2
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 3
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 the accuracy are computed for all four models and is plotted in a bar graph using the Matplotlib Python library as shown in the Figure. 2 and Figure. 3.

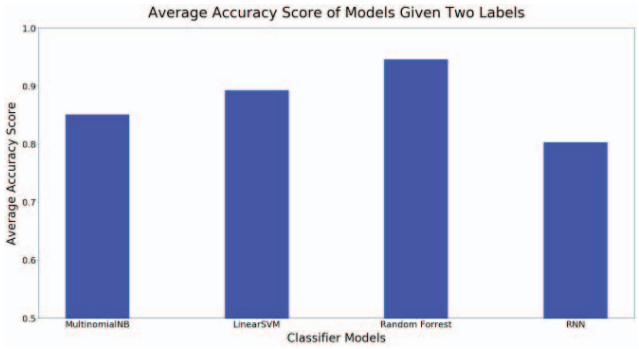


Figure.2 Average Accuracy of Models Given Two Labels

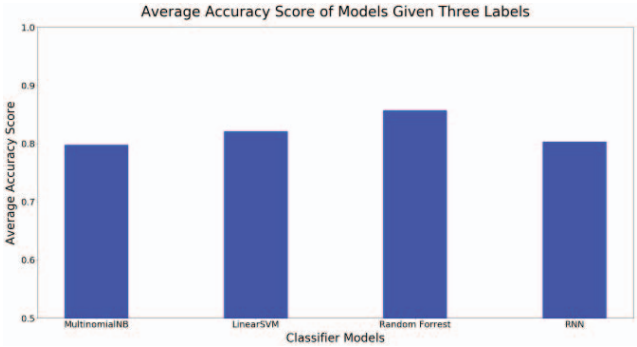
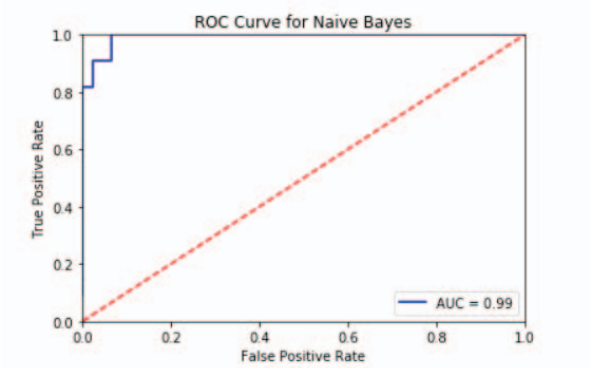
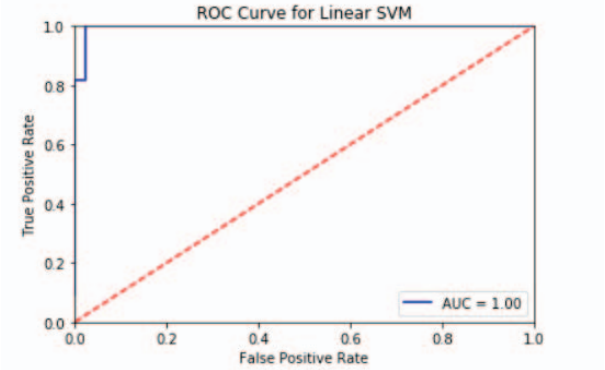


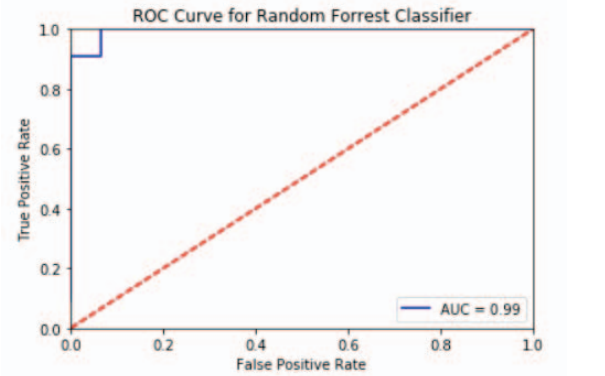
Figure.3 Average Accuracy of Models Given Three Labels

B. *ROC Curves*

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. 4. 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.







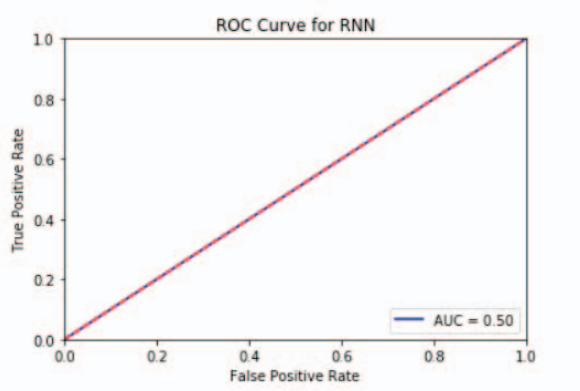


Figure.4 ROC Curves for Naïve Bayes, Linear SVM, Random Forrest and RNN models

C. *Comparison with Existing Approaches*

The research conducted by M.S. Barakat et al., [5] 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. 5.

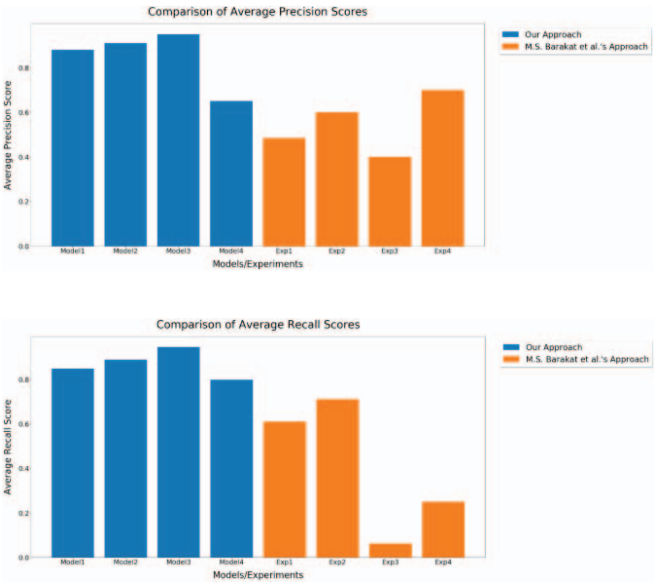


Figure.5 Comparison of Average Precision and Recall Scores

The research conducted by R. Kandakatla [6] 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. 6.

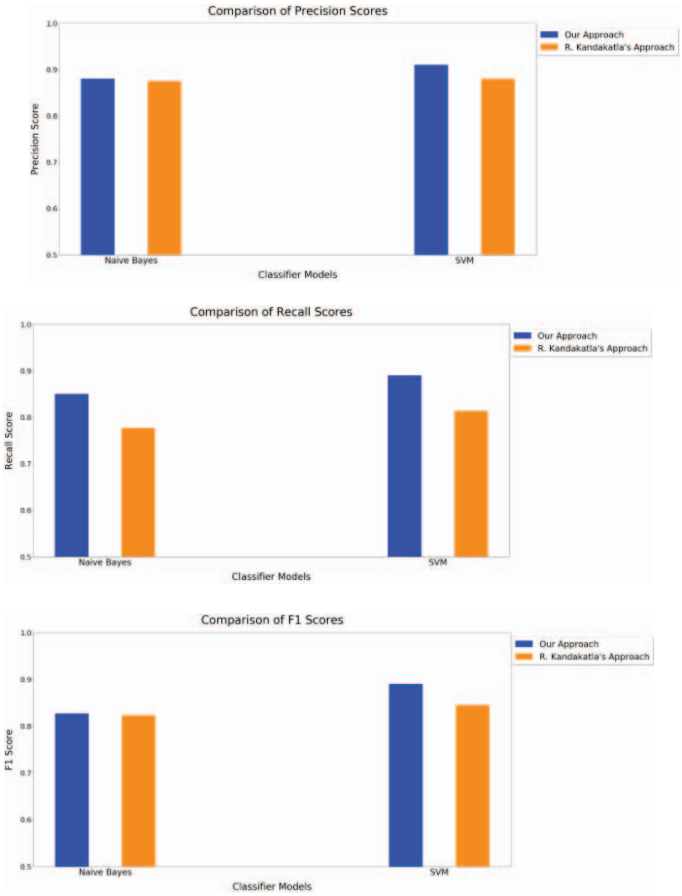


Figure.6 Comparison of Precision, Recall and F1 Scores

VI. Conclusion and Future Work

In this research, we implemented an approach to detect hate speech in videos which 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 %.

For future work, our approach can also be extended to classifying more than three categories as well as increasing the size of the dataset for better classification. This research 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] A. Razavi, D. Inkpen, S. Uritsky, S. Matwin, "Offensive Language Detection Using Multi-Level Classification", In Advances in Artificial Intelligence Springer, pp. 1627, 2019.

[2] 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.

[3] S. Biere, "Hate Speech Detection Using Natural Language Processing Techniques", August 2018.

[4] 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.

[5] 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.

[6] R. Kandakatla, "Identifying Offensive Videos on YouTube", 2016.

[7] YouTube Data API, 2019. [Online]. Available at: https://developers.google.com/youtube/v3. [Accessed: 04 February 2019].

[8] Pytube, 2019. [Online]. Available at: https://python-pytube.readthedocs.io/en/latest. [Accessed: 04 February 2019].

[9] FFmpeg, 2019. [Online]. Available at: https://ffmpeg.org. [Accessed: 04 February 2019].

[10] Google Cloud Speech-to-Text, 2019. [Online]. Available at: https://cloud.google.com/speech-to-text. [Accessed: 04 February 2019].

[11] TextBlob, 2019. [Online]. Available at: https://textblob.readthedocs.io/en/dev. [Accessed: 04 February 2019].

[12] 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].

[13] 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].

[14] 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.

[15] 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].

[16] A. Sherstinsky, "Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network", Nov 2018, arXiv:1808.03314v4

[17] "Recurrent Neural Networks", 2019. [Online]. Available at: https://machinelearning-blog.com/2018/02/21/recurrent-neural-networks. [Accessed: 10 April 2019].

[18] C. Goutte, É. Gaussier. "A Probabilistic Interpretation of Precision Recall and F-Score, with Implication for Evaluation", ECIR, 2015.

[19] 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].