A report on hate speech detection using natural language processing on twitter dataset

BY
Musharaf Maqbool
2020BCSE003
Computer Science and Technology
NIT SRINAGAR

Under the Supervision of
Dr. Nagendra Kumar
Assistant Professor
IIT Indore

NATIONAL INSTITUTE OF TECHNOLOGY SRINAGAR
(An autonomous Institute of National Importance under the aegis of Ministry of Education, Govt. of India)
DEPARTMENT OF TRAINING & PLACEMENT
Tel/Fax: +91-194-2424809 Extn: 2130/31 Email: placements@nitsri.net
Hazratbal , Srinagar Jammu and Kashmir, 190006, INDIA

NO: NIT/T&P/INTP/2022-23/CSE/004
Date: 04-11-2022

IIT INDORE

Subject: **Permission of Internship online/offline for student of NIT Srinagar.**

In – Plant/on–the –project internship/Practical Training is an important part of our engineering curriculum. This internship/training is regarded as a vital component of engineering education and is an indicator of extent of field experience, which is very essential for attaining excellence in the technical education. In this context, **Mr. /Ms. Musharaf Maqbool,** Enrolment No: **2020BCSE003** pursuing B. Tech in COMPUTER SCIENCE ENGINEERING DEPARTMENT (2020-2024) in this Institute has completed his/her 4th semester of the degree (pursuing in 5th semester) and is interested in 45 days internship in your esteemed organization.

It will be highly appreciated if your organization provides him/her a chance to get an exposure to some project related to him/her branch of engineering online/offline that is being carried out by your organization during winter vacation from 20th December 2022 to 15th February 2023.

We fervently hope that you will accede to our request and allow him/her to pursue him/her internship in your esteemed organization. The student has been advised to abide by the rules and regulation of your organization. Also, the student has to submit completion report and certificate in the training & placement department after completion of the internship, failing this his/her internship will be deemed incomplete.

Associate TPO (Internships)
Training and Placement
NIT Srinagar

Associate TPO (Internships)
Training & Placement Department
National Institute of Technology
Srinagar, J&K.

INDIAN INSTITUTE OF TECHNOLOGY INDORE

(An autonomous institute under the Ministry of Education, Government of India)

e-CERTIFICATE

This is to certify that Mr. Musharaf Maqbool has successfully completed the "Online Internship for the Undergraduate Students" with IIT Indore Faculty Mentor Dr. Nagendra Kumar, Department of Computer Science and Engineering, from December 15, 2022, to February 28, 2023. He has worked on the area entitled "Twitter Sentiment analysis".

Dr. Nagendra Kumar
IIT Indore Faculty Mentor

Professor Anand Parey
Dean, Alumni and Corporate Relations

ACKNOWLEDGEMENT

It is with a sense of gratitude; I acknowledge the effort of entire hosts of well-wishers who have in some way or other contributed in their own special ways to the success and completion of this internship project. I would like to express my sincerest gratitude to Dr. Nagendra Kumar for providing me with the opportunity to undertake this internship at Indian Institute of Technology Indore. His guidance and support during my time here were invaluable in helping me to gain a deeper understanding of Natural Language Processing. The knowledge and skills I have gained from working with a talented and dedicated team will be invaluable in my future endeavors. This internship has been a truly rewarding experience, and I am grateful for the opportunities it has provided me. I look forward to utilizing the knowledge and skills I have gained during my time here in the future.

DECLARATION

I hereby declare that the work contained in this report is a result of my own efforts and is submitted in partial fulfillment of the requirements for the Internship. The findings and conclusions of this report are based on the information collected and analyzed by me during my internship. This report has not been submitted in whole or in part, to any other university or institution for the award of any other degree or certification. The contents of this report are original and have not been plagiarized in any form. I am fully responsible for the accuracy and authenticity of the information contained in this report.

Musharaf Maqbool
B. Tech
Computer Science and Technology
NIT SRINAGAR

Under the Supervision of
Dr. Nagendra Kumar
Assistant Professor
IIT Indore

ABSTRACT

Hate speech is a prevalent and growing social problem due to the surge of social media technology usage. Despite the advancement of NLP technologies, the automated hate speech detection remains challenging. This report focuses on advancing the technology using state-of-the-art NLP techniques. We use a Twitter sentiment dataset for the detection of hate speech from tweets. Our best performing models are based on transformers and RNN.

INTRODUCTION

Hate speech is communication that disparages a person or group based on some characteristic such as race, color, ethnicity, gender, sexual orientation, nationality, religion, or other characteristics. It is important to detect hate speech because it can make people feel bad about themselves and can make them want to not use online services. We create machine learning and deep learning models for detection of hate-speech. Twitter's popularity has exploded in the previous few years, making it one of the most widely used social media sites. As a result of this development, the strategies described in this study are now more beneficial. Additionally, there has been an increase in the number of people who express their views in demeaning ways to others. As a result, hate speech has piqued interest in the subject of sentiment analysis, which has developed various algorithms for detecting emotions in social networks using intuitive means. Our goal in this work is to develop a system for automatically detecting hate speech through tweets using the various deep learning and machine learning models.
During the text analysis, word embedding is implemented Glove, Fast-Text, Bert, Mp-net etc which are then used with the combination of deep learning and machine learning models such as BiLSTM, LSTM, CNN, Logistic Regression, Naïve Bayes, and MLP.

PROBLEM STATEMENT

As mentioned above, our wok is to classify hate speech from tweets. We have obtained the tweet dataset from Kaggle which is labelled as 1 and 0 for hateful content and non-hateful content.

Dataset and data pre-processing

Dataset.
We used the Twitter Tweets dataset for the analysis. The dataset split into two parts, i.e., training dataset and the testing dataset. The training dataset is used for building the model, and the testing dataset is used for the validation of the model. We have pre-processed the text data using nltk and applied various text processing techniques and

then generated features and embeddings for the text and used these with various machine learning and deep learning models.

Pre-processing the dataset:
First, we check the dataset for missing or repeating values. If there is such data then we drop the row. Then we proceed to check the labels of the data. If the data is highly imbalanced, then we may proceed with various sampling techniques. After this we proceed with text pre-processing and feature extraction so that we can use various Machine Learning and Deep Learning models on the dataset.

Text Pre-processing
Text pre-processing is the process of cleaning the raw text data by removing the noise such as punctuations, emojis and common words to make it ready for our model to train. It is very important to remove unhelpful data or parts from our text. Text pre-processing improves the performance of an NLP system. For tasks such as sentiment analysis, document categorization, document retrieval based upon user queries, and more, adding a text pre-processing layer provides more accuracy. There are several types of text pre-processing techniques, one must use the ones suitable for their use case in an order that makes sense. Some common text pre-processing techniques include tokenization, stemming, lemmatization, stop word removal etc.

Tokenization
Tokenization is a common task in Natural Language Processing (NLP). It is a fundamental step in both traditional NLP methods like Count Vectorizer and Advanced Deep Learning-based architectures like Transformers. Tokens are the building blocks of Natural Language. Tokenization is a way of separating a piece of text into smaller units called tokens. By breaking text into tokens, it becomes easier to analyse and process the text programmatically. These tokens can be words, phrases, or even sentences.

Stemming
Stemming is the process of reducing a word to its base or root form, by removing any suffixes or prefixes that may be attached to it. The resulting word is called the stem, and it may not necessarily be a valid word in the language. Stemming is often used as a pre-processing step in natural language processing (NLP) tasks, such as information retrieval and text classification. By reducing words to their base form, it becomes easier to group related words together and perform analysis on them. For example, consider the following words: "walking", "walked", and "walks". These words all have the same root word "walk". By stemming these words, we can reduce them to their base form.

Lemmatization

Lemmatization is a text normalization technique used for Natural Language Processing (NLP). It can convert any word's inflections to the base root form. This allows us to categorize words with similar meanings, no matter how they are spelled. Lemmatization is similar to stemming but it brings context to the words. By reducing words to their base form, it becomes easier to perform analysis on them and to identify the meaning of the text. For example, consider the following words: "better", "best", and "good". These words have different forms, but they all have the same lemma "good".

Lemmatization is typically more accurate than stemming, but it is also computationally more expensive. There are various algorithms used for lemmatization in NLP, including the WordNet lemmatizer and the spaCy lemmatizer.

Stop words removal

Stop words are words that are commonly used in a language and are usually removed from texts because they do not carry important meaning. Examples of stop words in English include "the", "a", "an", "and", "or", "of", "in", "is", "that", "to", and "with". Stop words removal is a common technique used in natural language processing (NLP) to improve the efficiency and accuracy of text analysis. By removing stop words from a text, we can reduce the size of the data and focus on the more important words that carry meaning and contribute to the analysis.

Word/Sentence encoding techniques.

Count vectorizer and TF/IDF

Count vectorizer and TF-IDF are two common techniques used to convert raw text into numerical features that can be used as input to machine learning models in natural language processing (NLP) tasks.

Count vectorizer is a simple technique that counts the frequency of each word in a document and represents the document as a vector of word counts.

TF-IDF (Term Frequency-Inverse Document Frequency) is a more sophisticated technique that takes into account the frequency of each word in the entire corpus of documents, as well as the frequency of the word in the specific document being represented. TF-IDF assigns a weight to each word in the document that reflects both its frequency in the document and its rarity in the corpus. Words that are frequent in the document but rare in the corpus are given a high weight, while words that are common in both the document and the corpus are given a lower weight.

Bert model

BERT (Bidirectional Encoder Representations from Transformers) is a pre-trained language model developed by Google AI Language. The Transformer model consists of an encoder and a decoder. The BERT model is a variant of the Transformer encoder architecture that is trained on a large amount of text data using a language modelling objective.

The BERT encoder is a stack of transformer blocks, where each block consists of a multi-head self-attention mechanism and a feed-forward neural network. The self-attention mechanism allows the model to attend to different parts of the input sequence, while the feed-forward network provides non-linear transformations of the input. The encoder takes as input a sequence of tokens (usually words or sub-words), which are first converted into vector representations through an embedding layer. The resulting embeddings are then passed through a series of transformer blocks, where each block updates the embeddings by attending to the other tokens in the sequence.

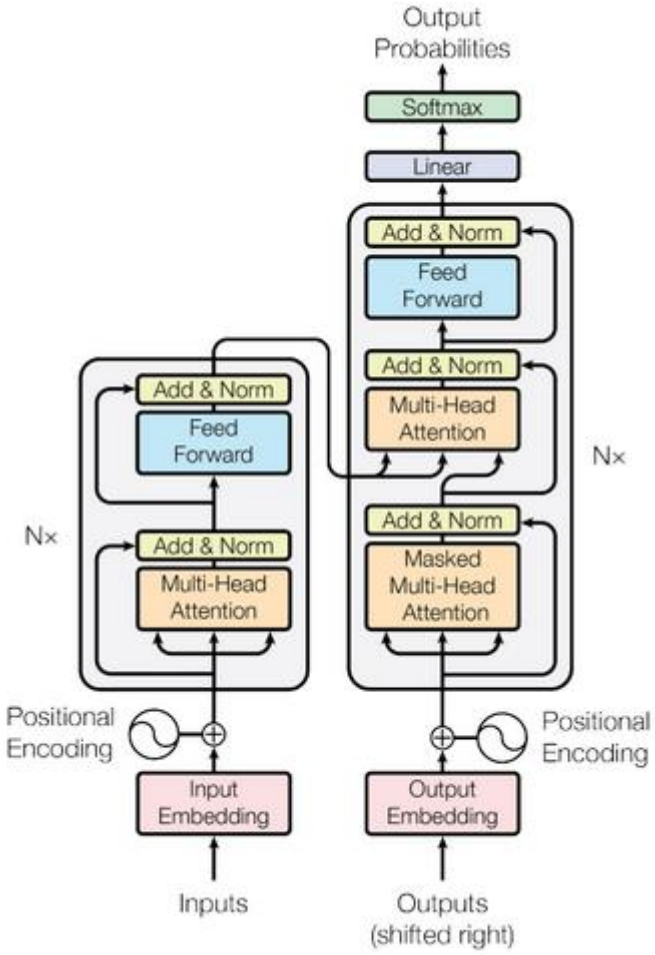
One of the key features of BERT is its bidirectional nature, which allows it to capture the context of each token from both its preceding and succeeding tokens. This is achieved through a technique called masked language modelling, where BERT randomly masks some tokens in the input sequence and predicts them based on the context of the other tokens in the sequence. After pre-training, the BERT model can be fine-tuned on specific downstream natural language processing tasks, such as text classification, question answering, and named entity recognition, by adding a task-specific output layer and training the entire model end-to-end. This allows the model to adapt to the specific nuances of the task and achieve state-of-the-art performance on a wide range of benchmarks.

Roberta

RoBERTa (Robustly Optimized BERT Pretraining Approach) is a transformer-based language model that was introduced in 2019 by Facebook AI Research. RoBERTa is a variation of the BERT (Bidirectional Encoder Representations from Transformers) model, which was pre-trained on a large corpus of text data using a modified training procedure to achieve state-of-the-art performance on a wide range of natural language processing tasks.

RoBERTa uses the same architecture as BERT, consisting of a stack of transformer blocks, but with modifications to the training process. Unlike BERT, RoBERTa is trained on a much larger corpus of text, with longer sequences and no masking of tokens during pre-training. The training data is also pre-processed using a more thorough cleaning procedure, which removes noise and improves the quality of the input. Unlike BERT, RoBERTa does not use masked language modelling during pre-training. Instead, RoBERTa uses a dynamic masking strategy where each training example is randomly

sampled from a large corpus of text and no special tokens are used to mask or replace the input.



General Transformer architecture

Mp-net Sentence transformer

Mp-net is a type of pre-trained neural network model for natural language processing (NLP) tasks, particularly for sentence and text embeddings. It is an extension of the popular BERT (Bidirectional Encoder Representations from Transformers) model, which is trained on large amounts of text data to produce high-quality language representations.

The Mp-net model, like BERT, is based on the transformer architecture, which allows the model to process long sequences of text and capture both local and global dependencies within the text. However, while BERT is designed for bidirectional language modelling, Mp-net is specifically optimized for sentence embeddings.

Google Sentence Transformer T5: T5 stands for "Text-to-Text Transfer Transformer". It was released by Google on 2020. It is a pre-trained neural network model for natural

language processing (NLP) tasks that is developed by Google. It is based on the transformer architecture, which allows the model to process long sequences of text and capture both local and global dependencies within the text. The Google Sentence Transformer model is pre-trained on a large corpus of text data and is designed to generate high-quality sentence embeddings, which are fixed-length vectors that capture the semantic meaning of sentences. The model is trained using a supervised learning approach, where the goal is to predict the similarity between pairs of sentences.

ELECTRA: ELECTRA (Efficiently Learning an Encoder that Classifies Token Replacements Accurately) is a transformer-based neural network model for pre-training natural language processing (NLP) tasks. It is designed to improve the efficiency and effectiveness of pre-training compared to traditional models like BERT. ELECTRA is trained using a novel pre-training approach called the "discriminative loss" method, which involves training a generator model to replace tokens in the input text with plausible alternatives, and training a smaller, discriminator model to distinguish between the original and replaced tokens. This approach is more efficient than traditional pre-training methods because it allows the generator to focus on learning to generate plausible tokens, while the discriminator focuses on learning to distinguish between the original and replaced tokens. ELECTRA can also be used as a sentence transformer, which is a type of model that can generate high-quality sentence embeddings. Sentence embeddings are fixed-length vectors that capture the semantic meaning of sentences and can be used for a wide range of NLP tasks, such as text classification, clustering, and retrieval.

Brief description of different machine and deep learning models.

Multinomial Naive Bayes

The Multinomial Naive Bayes algorithm is a Bayesian learning approach that is popular in Natural Language Processing (NLP). It is used to guess the tag of a text, such as an email or a newspaper story, using the Bayes theorem. The algorithm calculates each tag’s likelihood for a given sample and outputs the tag with the greatest chance. The Multinomial Naive Bayes algorithm is suitable for classification with discrete features (e.g., word counts for text classification). The multinomial distribution normally requires integer feature counts. However, in practice, fractional counts such as tf-idf may also work.

Logistic Regression

Logistic Regression is a supervised method of learning used for predicting the probability of a dependent or a target variable. Using Logistic Regression, you can

predict and establish relationships between dependent and one or more independent variables. Logistic regression is one of the most popular Machine Learning algorithms, which comes under the Supervised Learning technique.

Multilayer perceptron (MLP)

A multilayer perceptron (MLP) is a fully connected class of feedforward artificial neural network (ANN) that consists of three types of layers—the input layer, output layer, and hidden layer. The input layer receives the input signal to be processed, and the output layer produces the result of the processing. The hidden layer is where the processing takes place. MLPs have the same input and output layers but may have multiple hidden layers in between the aforementioned layers.

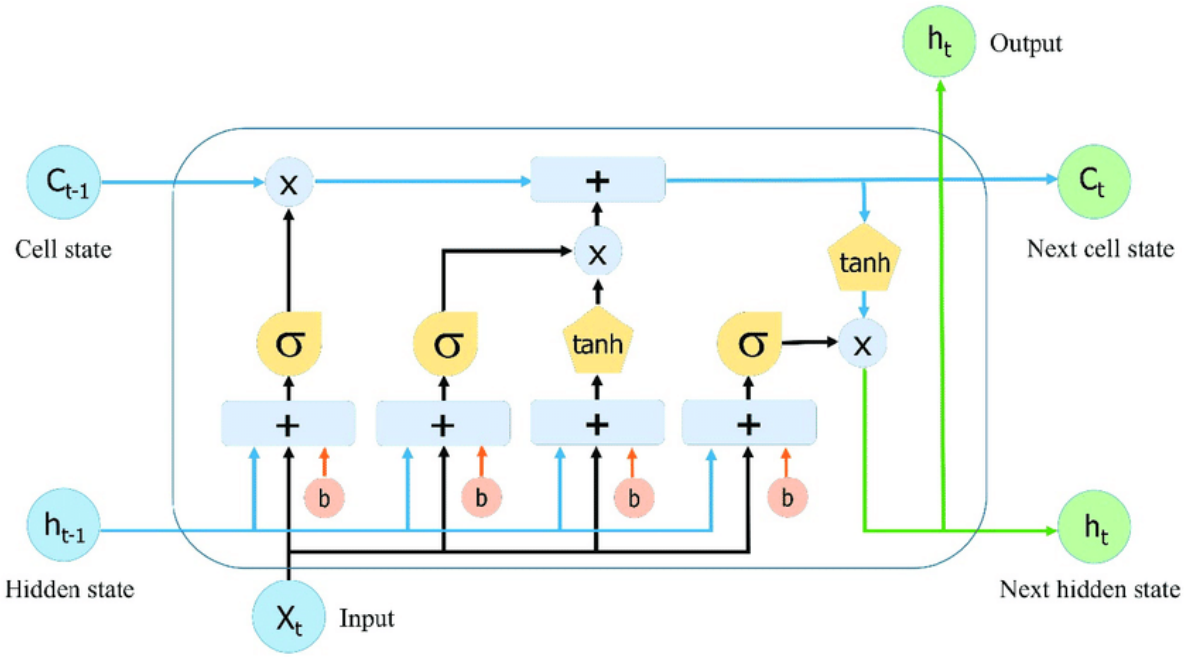
LSTM and BILSTM

LSTM (Long Short-Term Memory) and BILSTM (Bidirectional Long Short-Term Memory) are recurrent neural network (RNN) architectures that are commonly used in natural language processing (NLP) tasks.

LSTM is a type of RNN that is designed to handle the problem of vanishing gradients that often occurs in traditional RNNs. LSTM cells contain a memory cell, which is used to store information for long periods of time, and a set of gates that control the flow of information in and out of the cell. The gates are used to selectively forget, update, or output information from the cell, allowing the model to capture long-term dependencies in the input sequence.

BILSTM is a variation of the LSTM architecture that processes the input sequence in both forward and backward directions. By processing the input sequence in both directions, BILSTM is able to capture information from the past and future context of each input token, which can improve the model's ability to capture dependencies in the input sequence.

LSTM and BILSTM are powerful architectures for modelling sequential data in natural language processing tasks. LSTM is designed to capture long-term dependencies in the input sequence, while BILSTM processes the input sequence in both directions to capture information from both past and future context. Both LSTM and BILSTM have been shown to be effective in a wide range of NLP tasks, making them popular choices for many applications in this field.



Inputs:

X_t Current input

C_{t-1} Memory from last LSTM unit

h_{t-1} Output of last LSTM unit

Outputs:

C_t New updated memory

h_t Current output

Nonlinearities:

σ Sigmoid layer

tanh Tanh layer

b Bias

Vector operations:

X Scaling of information

\+ Adding information

Proposed Methodology

The deep learning models are integrated with the word embedding model such as inverse glove (global vector), document frequency (TF-IDF), and transformer-based embedding, the implemented models are the combination of the convolutional neural network, long short-term memory, and multilayer perceptron. The description of the models are as follows:

Model1: Count-vectorizer/TF-IDF with Naive Bayes, Logistic Regression and Multilayer perceptron.

We used NLTK to tokenize the tweets and to remove English stop-words. We also use a lemmatization technique which switches any kind of a word to its base root mode. We turned the pre-processed tweets into BOW by using Count-vectorization/TF-IDF technique which returns a vector with the length of the entire vocabulary (All Words) and count for the number of times each word occurs in the sentence.

We used Multinomial Naive Bayes model which is based on bayes theorem of probability, logistic regression model and multilayer perceptron model.

Results:

Multinomial Naive Bayes:

	precision	recall	f1-score	support
0	0.97	0.96	0.97	5985
1	0.53	0.62	0.57	408
accuracy			0.94	6393
macro avg	0.75	0.79	0.77	6393
weighted avg	0.95	0.94	0.94	6393

Using this model, we get precision and recall of 97% and 96% respectively for group 0 And we get 53% and 62% precision and recall respectively for group 1.

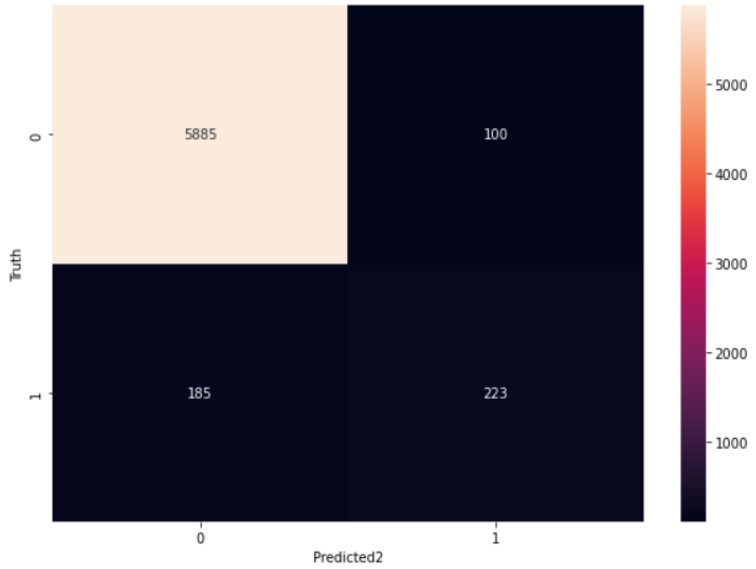
Logistic Regression:

	precision	recall	f1-score	support
0	0.98	0.91	0.95	5985
1	0.38	0.76	0.51	408
accuracy			0.91	6393
macro avg	0.68	0.84	0.73	6393
weighted avg	0.94	0.91	0.92	6393

Using this model, we get precision and recall of 98% and 91% respectively for group 0 And we get 38% and 76% precision and recall respectively for group 1.

Multilayer perceptron:
The MLP network has three layers:
(i) An input layer with 6 neurons with a uniform kernel initializer and ReLU as activation function.
(ii) A hidden layer with 6 neurons activated by ReLU activation function.
(iii) An output layer with the rest of the neurons (a single neuron with sigmoid activation function)
(iv) All of our models have been built using the Keras library

Using this model, we get precision and recall of 96.9% and 98.3% respectively for group 0 and we get 69.04% and 54.6% precision and recall respectively for group 1.



Model2: Using Multilayer perceptron with Bert encodings.

Text Pre-processing: We used NLTK to tokenize the tweets and to remove English stop-words. We also use a lemmatization technique which switches any kind of a word to its base root mode.

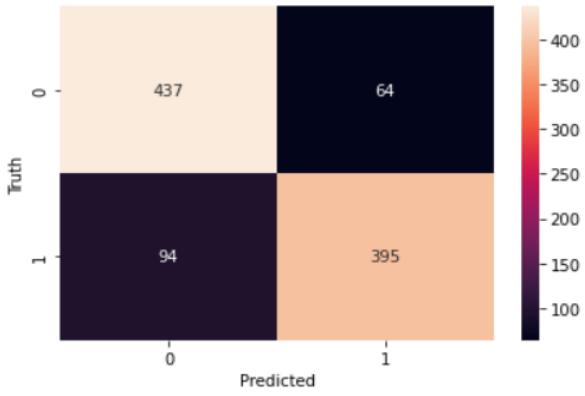
Sentence embeddings: We use the pre-trained BERT pre-processor and encoder to create the sentence embeddings.

We apply under sampling technique on the data-set. Under sampling is a technique used in machine learning to address class imbalance problems. Class imbalance occurs when the number of instances of one class is much smaller than the number of instances of another class in the dataset.
In under sampling, some of the instances from the majority class are removed from the training dataset.
Model: We use artificial neural network model with Bert encoding.

The MLP network has three layers:
(i) An input layer with 6 neurons with a uniform kernel initializer and ReLU as activation function.
(ii) A hidden layer with 6 neurons activated by ReLU activation function.
(iii) An output layer with the rest of the neurons (a single neuron with sigmoid activation function)
(iv) All of our models have been built using the Keras library

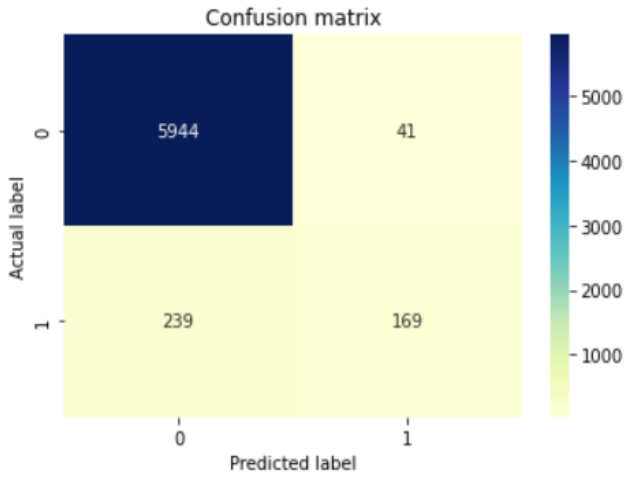
We use adam as optimizer and binary cross entropy as the loss function.
Results:

	precision	recall	f1-score	support
0	0.82	0.87	0.85	501
1	0.86	0.81	0.83	489
accuracy			0.84	990
macro avg	0.84	0.84	0.84	990
weighted avg	0.84	0.84	0.84	990



Using this model, we get precision and recall of 82% and 87% respectively for group 0 and we get 86% and 81% precision and recall respectively for group 1.

If we don't use any under sampling/oversampling technique we get this result.




```
200/200 [==============================] - 0s 2ms/step
              precision    recall  f1-score   support

           0       0.96      0.99      0.98      5985
           1       0.80      0.41      0.55       408

    accuracy                           0.96      6393
   macro avg       0.88      0.70      0.76      6393
weighted avg       0.95      0.96      0.95      6393
```

We get precision and recall of 96% and 99% respectively for group 0 and we get 80% and 41% precision and recall respectively for group 1.

Model 3: Using Multilayer perceptron with Bert encodings and over sampling the minority class.

Text Pre-processing: We used NLTK to tokenize the tweets and to remove English stop-words. We also use a lemmatization technique which switches any kind of a word to its base root mode.
We separate some data for testing and use smote on training data.
SMOTE (Synthetic Minority Over-sampling Technique) is a popular oversampling technique used in machine learning to address class imbalance problems. SMOTE generates synthetic examples of the minority class by creating new instances along the line segments that connect k nearest neighbours of each instance in the minority class.
Sentence embeddings are created using Bert transformer.

Model: We directly use artificial neural network model which consists of
(i) An input layer with 12 neurons with a uniform kernel initializer and GeLU as activation function.
(ii) A hidden layer with 8 neurons activated by GeLU activation function.
(iii) An output layer with the rest of the neurons (a single neuron with sigmoid activation function)
with Bert encoding and smote.

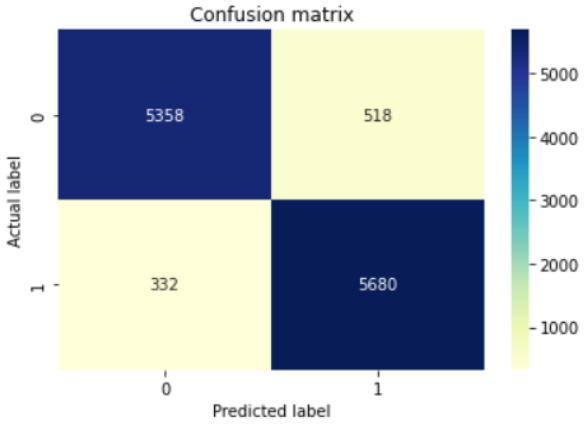
Results**:** The results depend upon the number of elements separated for testing.

```
df_test0=df_test0.iloc[0:1000,]
df_test1=df_test1.iloc[0:1000,]
```

```
372/372 [==============================] - 1s 2ms/step
              precision    recall  f1-score   support

           0       0.94      0.91      0.93      5876
           1       0.92      0.94      0.93      6012

    accuracy                           0.93     11888
   macro avg       0.93      0.93      0.93     11888
weighted avg       0.93      0.93      0.93     11888
```



We have separated 1000 entries from group 0 and 1 for testing set.
We get precision and recall of 94% and 91% respectively for group 0 and we get 92% and 94% precision and recall respectively for group 1.

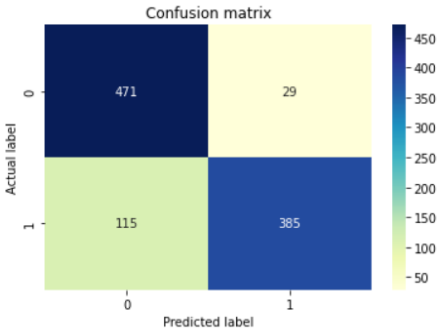
We have separated 500 elements from group 0 and group 1 for testing.

```
df_test0=df_test0.iloc[0:500,]
df_test1=df_test1.iloc[0:500,]
```

```
32/32 [==============================] - 0s 1ms/step
              precision    recall  f1-score   support

           0       0.80      0.94      0.87       500
           1       0.93      0.77      0.84       500

    accuracy                           0.86      1000
   macro avg       0.87      0.86      0.85      1000
weighted avg       0.87      0.86      0.85      1000
```



We get precision and recall of 80% and 94% respectively for group 0 and we get 93% and 77% precision and recall respectively for group 1.

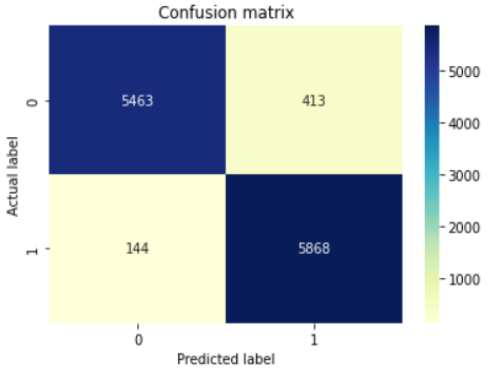
Model:4 Using LSTM and Smote with Bert encodings.

Text Pre-processing: We used NLTK to tokenize the tweets and to remove English stop-words. We also use a lemmatization technique which switches any kind of a word to its base root mode.
We use smote on the data-set to increase minority elements.

Sentence embeddings: We use the pre-trained BERT pre-processor and to create embeddings for the sentences.

Model**:** We use LSTM (Long Short-Term Memory) is a type of recurrent neural network (RNN).

Results:



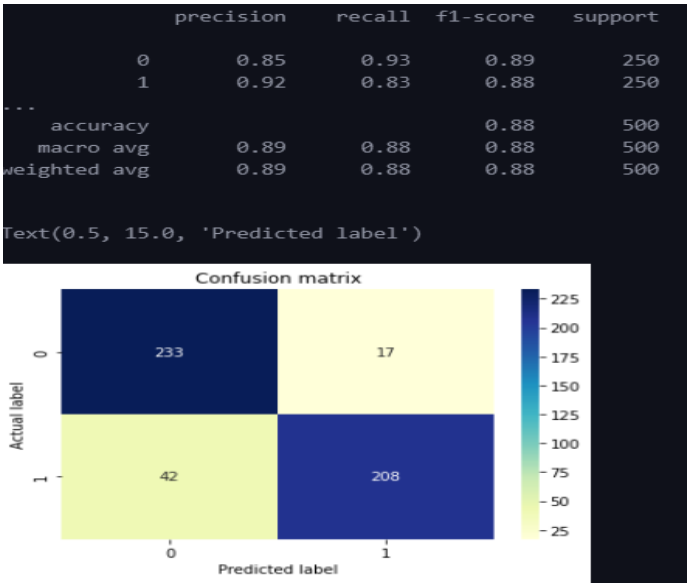
	precision	recall	f1-score	support
0	0.97	0.93	0.95	5876
1	0.93	0.98	0.95	6012
accuracy			0.95	11888
macro avg	0.95	0.95	0.95	11888
weighted avg	0.95	0.95	0.95	11888

We get precision and recall of 97% and 93% respectively for group 0 and we get 93% and 98% precision and recall respectively for group 1.

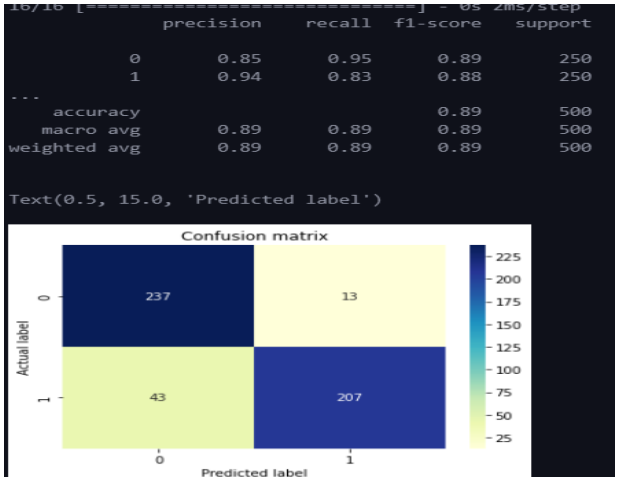
Model 5: Using LSTM/BILSTM with Mp-net encodings.

Text Pre-processing: We used NLTK to tokenize the tweets and to remove English stop-words. We also use a lemmatization technique which switches any kind of a word to its base root mode.
We use smote on the data-set to increase minority elements.

Model: We have used LSTM and BiLSTM with Mp-net sentence embeddings.
We have an input layer with 12 LSTM/BILSTM units and 6 dense layers with ReLU activation function and one output layer with sigmoid activation function.



We get precision and recall of 85% and 93% respectively for group 0 and we get 92% and 83% precision and recall respectively for group 1.



We get precision and recall of 85% and 95% respectively for group 0 and we get 94% and 83% precision and recall respectively for group 1.

Model:6

Text Pre-processing: We used NLTK to tokenize the tweets and to remove English stop-words. We also use a lemmatization technique which switches any kind of a word to its base root mode.

We use smote on the data-set to increase minority elements.

Sentence embeddings: We have used different transformer-based models to generate the sentence embeddings.

(i) Google Sentence Transformer T5

(ii) RoBERTa:

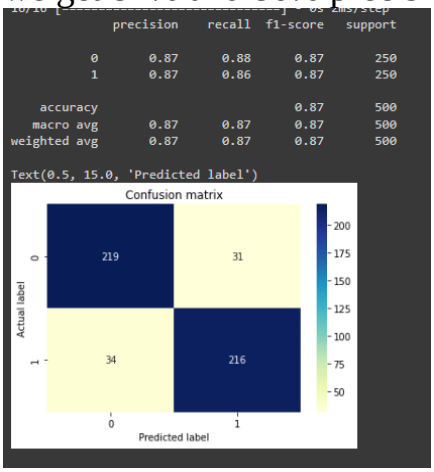
(iii) ELECTRA:

Model: We have used LSTM and BILSTM with different sentence embeddings. We have an input layer with 6 LSTM/BILSTM units and 6 dense layers with ReLU activation function and one output layer with sigmoid activation function.

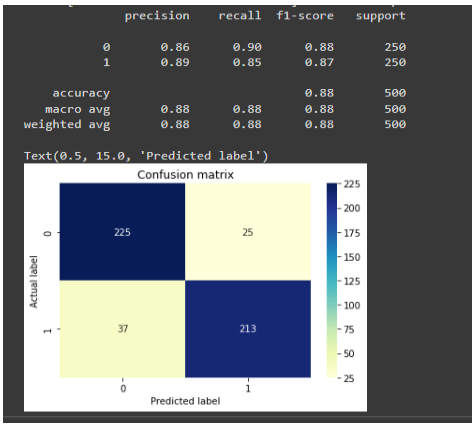
Results:

Using embeddings from google sentence transformer:

Using lstm: We get precision and recall of 87% and 88% respectively for group 0 and we get 87% and 86% precision and recall respectively for group 1.

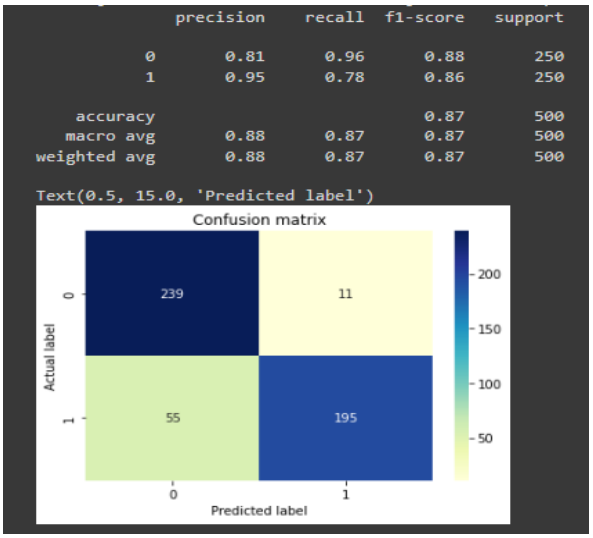


Using bilstm: We get precision and recall of 86% and 90% respectively for group 0 and we get 89% and 85% precision and recall respectively for group 1.

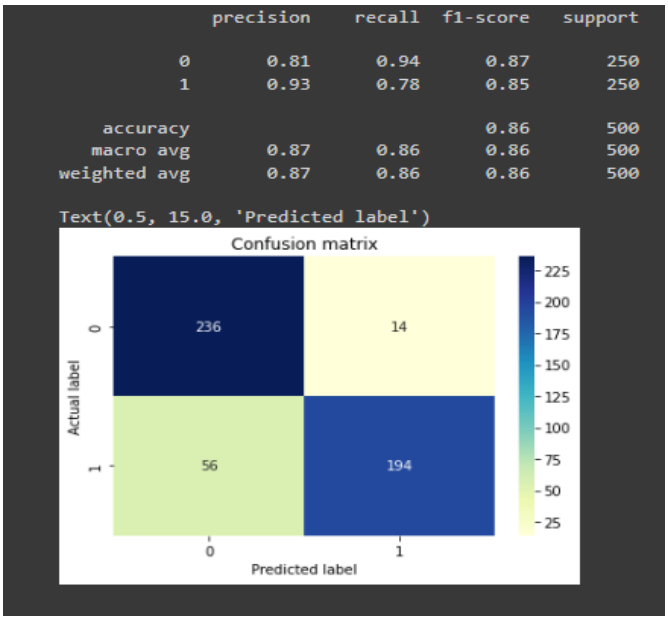


Using embeddings from Roberta model:

Using lstm: We get precision and recall of 81% and 96% respectively for group 0 and we get 95% and 78% precision and recall respectively for group 1.



Using bilstm: We get precision and recall of 81% and 94% respectively for group 0 and we get 93% and 78% precision and recall respectively for group 1.

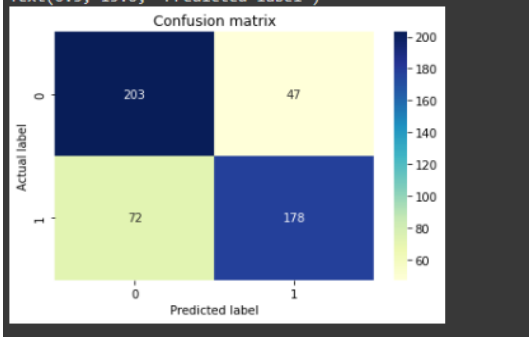


Using embeddings from Electra model:

Using lstm: We get precision and recall of 74% and 81% respectively for group 0 and we get 79% and 71% precision and recall respectively for group 1.

	precision	recall	f1-score	support
0	0.74	0.81	0.77	250
1	0.79	0.71	0.75	250
accuracy			0.76	500
macro avg	0.76	0.76	0.76	500
weighted avg	0.76	0.76	0.76	500

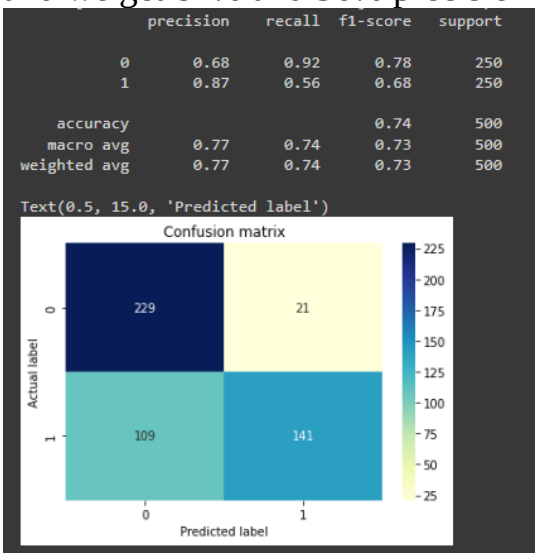
```
Text(0.5, 15.0, 'Predicted label')
```

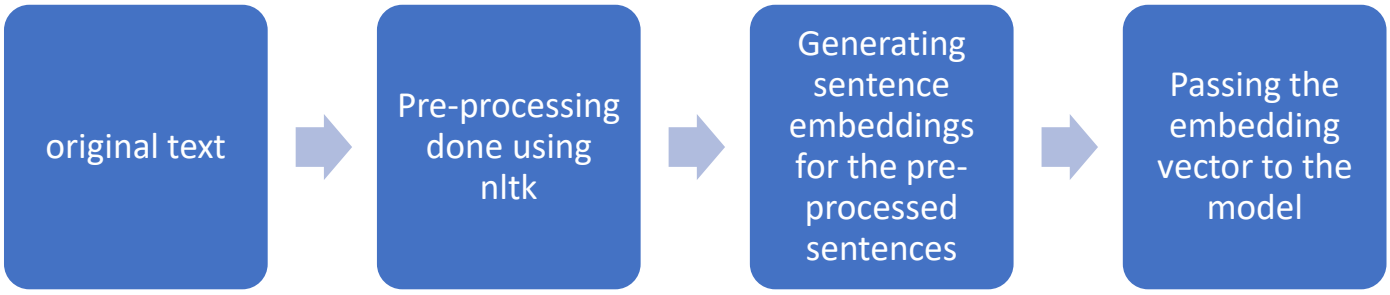


Using bilstm: We get precision and recall of 68% and 92% respectively for group 0 and we get 87% and 56% precision and recall respectively for group 1.

	precision	recall	f1-score	support
0	0.68	0.92	0.78	250
1	0.87	0.56	0.68	250
accuracy			0.74	500
macro avg	0.77	0.74	0.73	500
weighted avg	0.77	0.74	0.73	500

```
Text(0.5, 15.0, 'Predicted label')
```





S No	Model	Precision	Recall	F1 score
1	Count vectorizer/Tf-Idf + Naïve Bayes	.75	.79	.77
2	Count vectorizer/Tf-Idf + Logistic Regression	.68	.84	.73
3	BERT + Multilayer Perceptron (undersampling)	.84	.84	.84
4	BERT + Multilayer Perceptron	.88	.70	.76
5	BERT + Multilayer Perceptron + SMOTE	.93	.93	.93
6	BERT + LSTM +SMOTE	.95	.95	.95
7	MP_NET + BILSTM +SMOTE	.89	.89	.89
8	MP_NET + LSTM +SMOTE	.89	.88	.89
9	Sentence T5 + LSTM +SMOTE	.87	.87	.87
10	Sentence T5 + BILSTM + SMOTE	.88	.88	.88
11	Roberta + LSTM +SMOTE	.88	.87	.87
12	Roberta + BILSTM + SMOTE	.87	.86	.86
13	Electra + LSTM +SMOTE	.76	.76	.76
14	Electra + BILSTM + SMOTE	.77	.74	.73

Only Model 4 and 5 are tested on synthetic samples. Rest all test only tested on non-synthetic samples.

CONCLUSION

This paper focuses on text categorization in which the datasets is based on English language. We have used different types of machine learning and deep learning models based on ANN and RNN. Thus far, the success of our deep learning models has been the best. The parameters used for evaluating the model are precision, Recall and the F1 score. The model implemented with the transformer word embedding and MLP and RNN performs the best among the different models.