AN AI BASED OFFENSIVE LANGUAGE DETECTION SYSTEM

[1]K. Prabhakar, [2]D. Anvita, [3]K. Medha, [4]M. Chandini

[1]Professor, [2,3,4] B. Tech

[1]Computer Science and Engineering,

[1]Dhanekula Institute of Engineering and Technology, Ganguru, India

Abstract: Offensive language is the misdeed of using any natural language which could cause offence to a reasonable person or group of people in view of a social platform. Netizens write and post abusive language on social media in view of addressing a problem, criticize some person or thing for some reason, etc. This leads to frustration, depression and large change in their behaviour. An automatic discriminative software system, taking into account some sensitivity parameters for offensive language detection would be a useful tool. There are some systems which could address this problem but could not perform in an efficient way. The proposed system is based on machine learning techniques identify the offensive language used with improved accuracy.

***IndexTerms* – Offensive language.**

I. Introduction

Language is a form of communication that allows us to speak, read and write. Now a days we have large access to social media, where people are free to post their feelings on the internet. People have more freedom to express themselves openly and anonymously thanks to the widespread usage of the internet in many forms such as blogs, business networks, forums, and social media such as Twitter and Facebook. Social media is generating a lot of information through which individuals can exchange ideas in a positive way as well as contribute to the spread of hate speech in a negative way, as it has opened up a new galaxy of opportunities for people to express themselves. As a result, tools and methods that can automatically recognize hate speech and objectionable content on the internet and stop it from spreading are desperately needed.

II. Literature Survey

Existing System

- To cope with the problem of offensive language, researchers have presented numerous machine learning algorithms and their variants in the pat. Many of the proposed projects make use of text feature extraction techniques like BOW (Bag of words) and dictionaries. The majority of research in this area is focused on extracting features from text.
- Flames are described as demonstrating extreme subjectivity depending on the context in the literature on offensive language detection and specifically on natural language analysis. Subjectivity of this type is either speculative or evaluative.
- Evaluative expressions contain emotions (such as hate, anger), judgments, and opinions, whereas speculative expressions include any questionable terms.
- As a result, any hint of extremeness in such subjectivities could be used as a useful trait for evaluating and possibly detecting flames.

The existing system fail to meet the efficiency requirements, thus call for a system with much improved accuracy.

Proposed System

- The system connects to the Twitter API, extracts the tweets, and saves them for training purposes.
- Now, data pre-processing such as cleaning of data is done on the extracted data.
- If the data is not present in a good dimension, then techniques like overfitting and under-fitting are applied.
- A suitable Deep Learning Approach (DistilBERT) is used to train, test and to process the data.
- After fedding the data into the algorithm, a tweet is supplied for prediction to identify whether it is offensive language tweet or hate speech tweet or neither.
- Later analysis is done on the processed data and represented using pie charts or bar graphs.
- Overall efficiency of the system is calculated and evaluated.

III. Offensive Language Detection System

The goal of this study is to anticipate inappropriate language based on a tweet from a Twitter user. We make our own dataset by scraping tweets from a variety of Twitter domains and labelling them with polarity ratings given by the Textblob Python module. Then, to make predictions on this dataset, we build several deep learning models (RNN, GRU, and CNN). We investigate the effects of character-based models in these models.

The Offensive Language Detection System is built on the DistilBERT model using Python programming language.

STEP-1: Firstly, essential libraries are loaded into the model. If any of the required library is not available on the system they are to be installed on the system and then be loaded into the model.

STEP-2: Load the training data set and the test data set. It is the time to set up GPU for training. There is no need to perform data pre-processing as the BERT model is capable of cleaning the data. Print the sample data of the training data and test data to know the dimension of the data. Perform any modifications if necessary and modify the data according to the required format.

STEP-3: Tokenization and Input Formatting is done on the data. We must utilize the tokenizer provided by the library to apply the pre-trained BERT.

STEP-4: Train the model. The training data and the test data are fed into the model. The model is trained by creating a Bert Classifier. Now define the optimizer and the learning rate scheduler. The training loop continues by the number of epochs specified in the BERT model. The accuracy of each epoch is provided by the model.

STEP-5: The data predictor is defined. New tweets are tested using the predictor function. The predictor determines if the information is offensive, hateful, or neither.

IV. Workflow

Fig-1: Flow Chart

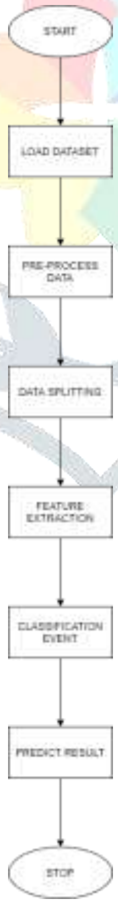
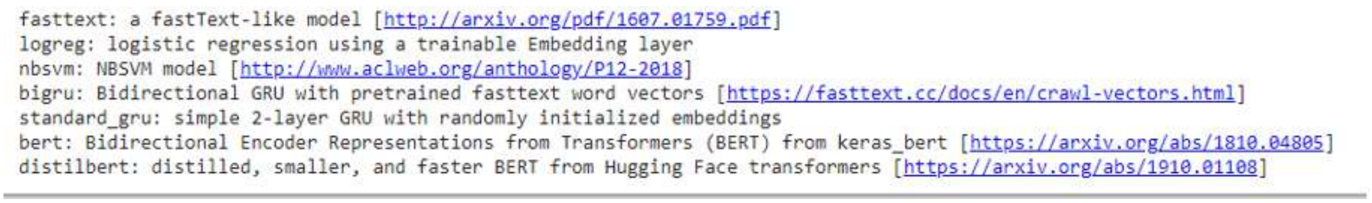


Figure 1 shows the workflow of the system. It represents the flow of the system. The system is defined in the way represented in the flowchart. It follows the path specified in the flowchart. The execution of the system is done based on the flowchart specified. The events are carried out in a systematic manner to obtain the desired output.

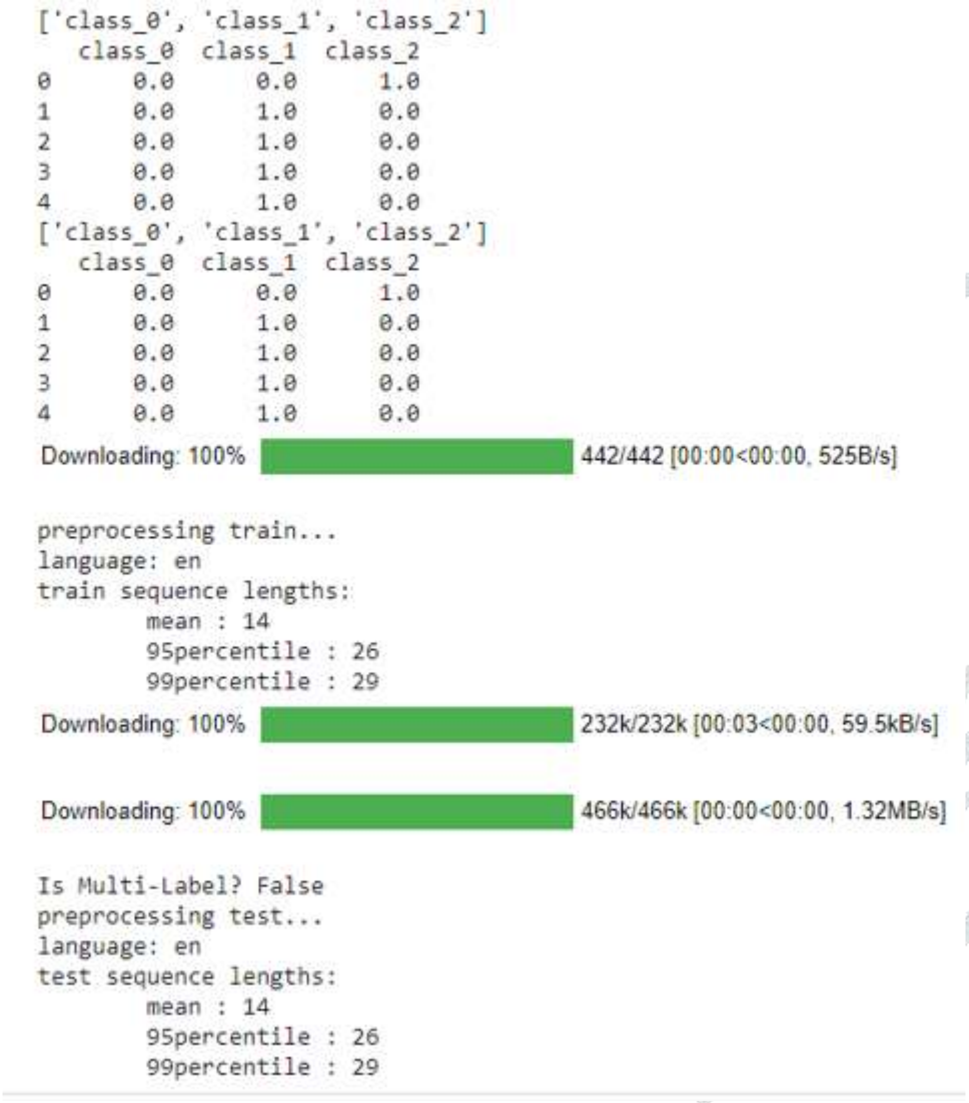
V. Result

Tokenizing and input formatting



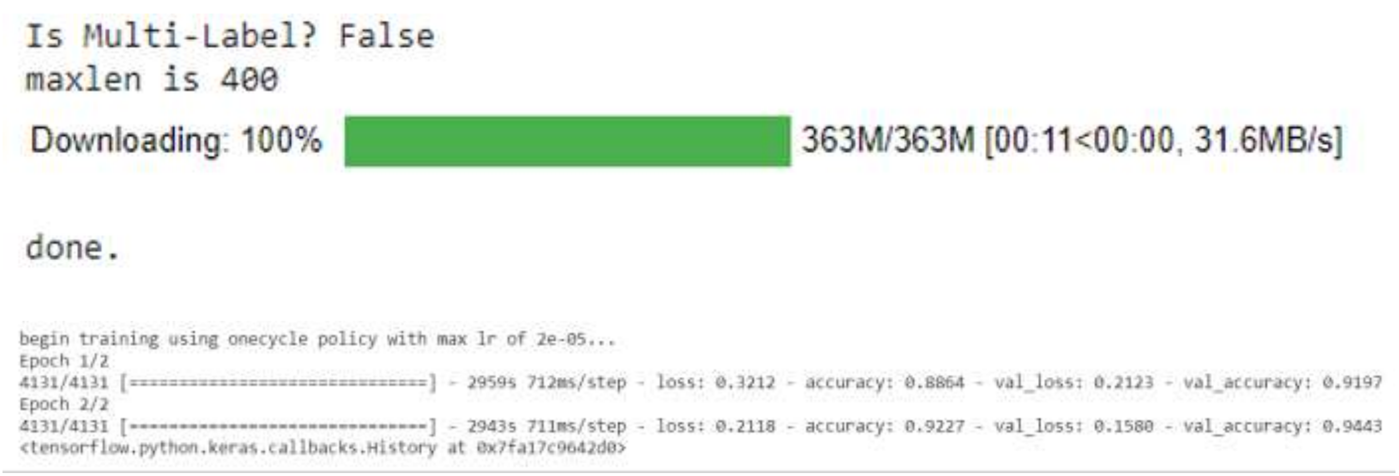
```
fasttext: a fastText-like model [http://arxiv.org/pdf/1607.01759.pdf]
logreg: logistic regression using a trainable Embedding layer
nbsvm: NBSVM model [http://www.aclweb.org/anthology/P12-2018]
bigru: Bidirectional GRU with pretrained fasttext word vectors [https://fasttext.cc/docs/en/crawl-vectors.html]
standard_gru: simple 2-layer GRU with randomly initialized embeddings
bert: Bidirectional Encoder Representations from Transformers (BERT) from keras_bert [https://arxiv.org/abs/1810.04805]
distilbert: distilled, smaller, and faster BERT from Hugging Face transformers [https://arxiv.org/abs/1910.01108]
```

Pre-trained model



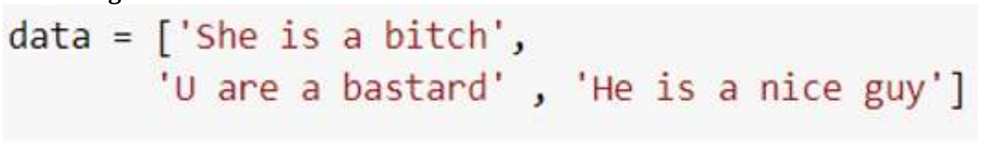
```
['class_0', 'class_1', 'class_2']
   class_0  class_1  class_2
0      0.0      0.0      1.0
1      0.0      1.0      0.0
2      0.0      1.0      0.0
3      0.0      1.0      0.0
4      0.0      1.0      0.0
['class_0', 'class_1', 'class_2']
   class_0  class_1  class_2
0      0.0      0.0      1.0
1      0.0      1.0      0.0
2      0.0      1.0      0.0
3      0.0      1.0      0.0
4      0.0      1.0      0.0
Downloading: 100%  442/442 [00:00<00:00, 525B/s]

preprocessing train...
language: en
train sequence lengths:
        mean : 14
        95percentile : 26
        99percentile : 29
Downloading: 100%  232k/232k [00:03<00:00, 59.5kB/s]

Downloading: 100%  466k/466k [00:00<00:00, 1.32MB/s]

Is Multi-Label? False
preprocessing test...
language: en
test sequence lengths:
        mean : 14
        95percentile : 26
        99percentile : 29
```

Training the model

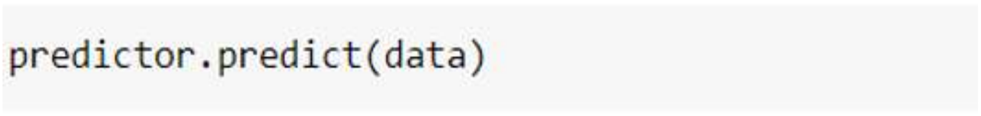


```
Is Multi-Label? False
maxlen is 400
Downloading: 100%  363M/363M [00:11<00:00, 31.6MB/s]

done.

begin training using onecycle policy with max lr of 2e-05...
Epoch 1/2
4131/4131 [==============================] - 2959s 712ms/step - loss: 0.3212 - accuracy: 0.8864 - val_loss: 0.2123 - val_accuracy: 0.9197
Epoch 2/2
4131/4131 [==============================] - 2943s 711ms/step - loss: 0.2118 - accuracy: 0.9227 - val_loss: 0.1580 - val_accuracy: 0.9443
<tensorflow.python.keras.callbacks.History at 0x7fa17c9642d0>
```

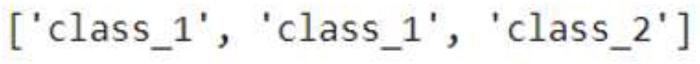
Predicting the result



```
data = ['She is a bitch',
        'U are a bastard' , 'He is a nice guy']
```



```
predictor.predict(data)
```



```
['class_1', 'class_1', 'class_2']
```

Where “class_1” refers “offensive” and “class_2” refers “not-offensive”.

VI. Conclusion

Finally, we introduced a novel technique to word embedding optimization for classification applications. On a user level, we conducted a comparative evaluation of some of the most extensively used deep learning models for depression identification from tweets. Based on recognizing offensive users, we constructed a predictive algorithm to predict whether a user's tweet is offensive or not.

Using a dataset acquired from Twitter and manually built, we evaluated the performance of the four classifiers. We discovered that those who are offensive are more socially isolated, as indicated by how they interacted with popular emojis and hashtags in their tweets.

REFERENCES

[1] Davidson, T., Warmsley, D., Macy, M. and Weber, I. (2017) Automated Hate Speech Detection and the Problem of Offensive Language. Proceedings of ICWSM.

[2] Kumar, R., Ojha, A.K., Malmasi, S. and Zampieri, M. (2018) Benchmarking Aggression Identification in Social Media. In Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC). pp. 1-11.

[3] Malmasi, S., Zampieri, M. (2018) Challenges in Discriminating Profanity from Hate Speech. Journal of Experimental & Theoretical Artificial Intelligence. Volume 30, Issue 2, pp. 187-202. Taylor & Francis.

[4] Waseem, Z., Davidson, T., Warmsley, D. and Weber, I. (2017) Understanding Abuse: A Typology of Abusive Language Detection Subtasks. Proceedings of the Abusive Language Online Workshop.