HATE AND OFFENSIVE TEXT DETECTION USING DEEP LEARNING

Kundhi Kiranmai

M. Tech
Department of Computer Science and System Engineering
Andhra University College of Engineering
Visakhapatnam, 530003

Abstract- **In recent years, people write and post abusive language on online social media platforms such as Twitter, Facebook, etc which is easily spread on internet. due to enormous volume of such posts the problem of detecting hateful and offensive text in social media is very difficult to solve manually. Hence systems that can automatically detect hate and offensive text in social media has lot of significance in modern world. In this project, Bidirectional Encoder Representation from Transformers (BERT+CNN), Convolutional Neural Network (CNN), and Linear Short Term Memory (LSTM) are used to identify hateful text. A benchmark dataset of approximately 25 thousand annotated tweets or used to construct models based on deep learning methods. The effectiveness of BERT+CNN , CNN and LSTM models are experimentally analyzed and compared all these models classification performance. The overall aim of these project is to develop an efficient Deep Learning model for detection of hateful and offensive text automatically.**

Keywords: **Enormous, Significance, CNN, LSTM, BERT, Approximately.**

1.Introduction

Social media has been widely used for a variety of reasons, including news, business, and advertising. the concept of enabling users to publish anything on social media at any moment for global visibility. Online, some users willfully spew hostile, threatening, or violent statements. Any public communication that disparages a person or a group on the basis of some traits, such as race, color, ethnicity, gender, sexual orientation, nationality, religion, or other qualities, is usually referred to as hate speech.

Social networks promote more covert and anonymous interactions between users, giving certain users the ability to remain anonymous while still feeling comfortable to spew hate speech. If it continues to be unregulated and produces disruptive antisocial effects, it

The first step in removing accounts with objectionable information from a social media platform is to identify the polarity of the text. This is crucial for government agencies, social security organizations, law enforcement, and social media firms. Automatic identification of hostile text will enable the platform to recognize the hateful content and eliminate them much more quickly and efficiently than manual filtering, which takes a lot of time. Both the scientific community and the commercial sector are now more interested in the issue of detecting online hate speech. Numerous studies have attempted to automate the procedure, which is typically treated as a supervised classification issue. Recently, a machine learning approach has been developed that can learn the many correlations between distinct textual passages.

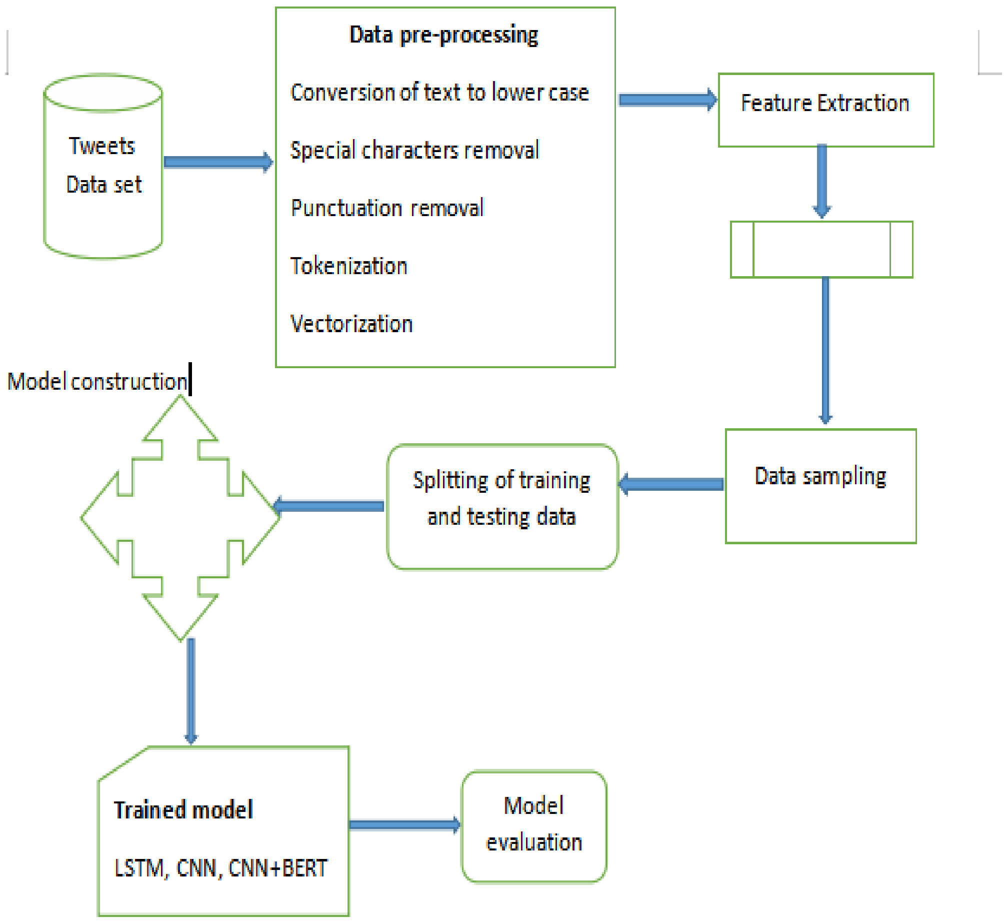


Fig 1. Architecture of Hate and Offensive text detection method

In this paper, we focused on several famous deep learning methods such as LSTM , CNN and BERT+CNN first we classified LSTM and CNN models individually then classify the BERT+CNN pipeline model, compare these three models.

2. Literature review

The hate and offensive text detection is a crucial part in social media as it helps to idetintifying hate text in the social medial and preventing them from causing problems of negative response from people. Several techniques have been proposed in the literature for hate speech detection ranging from machine learning algorithms to deep learning algorithms. In this literature survey we will review some of the significant works done in hate speech detection.

The hate speech is often found on social media, such as malicious comments intended to insult individuals or groups. One of the most used social media platforms in Indonesia is Twitter, with 63.6% of users. According to the Indonesia National Police, hate speech cases were more dominant during the period from April 2020 to July 2021. Combating cybercriminals is also difficult, therefore infrastructure and personnel are required. Therefore, efforts are needed to identify hate speech on the Twitter platform in Indonesian so that law enforcement can detect the spread of hate speech. Deep learning is one method for detecting hate speech. In this research, we use a deep learning model of Long Short-Term Memory (LSTM) with word embedding. Fast Text and Global Vector that we use as input for word representation and classification. Fast Text embedding s make use of sub-word information to create word embedding s and GloVe embedding s using an unsupervised learning method trained on a corpus to generate distributional feature vectors. From the evaluation results on the experimental model, LSAT-Fast Text using random oversampling has an advantage with an F1-score of 89.91% compared to LSAT-GloVe to obtain an F1-score of 82.14%.

The textual hate speech detection systems and highlights their primary datasets, textual features, and machine learning models. The results of this literature review are integrated with content analysis, resulting in several themes for 138 relevant papers. This study shows several approaches that do not provide consistent results in various hate speech categories. The most dominant sets of methods combine more than one deep learning model. Moreover, the analysis of several hate speech datasets shows that many datasets are small in size and are not reliable for various tasks of hate speech detection. Therefore, this study provides the research community with insights and empirical evidence on the intrinsic properties of hate speech and helps communities identify topics for future work.

3. Related Work

This section gives an terminology of hate speech detection in the field and it provides information about the used methodologies for both of the features and classifiers.

3.1 TERMINOLOGY

Hate speech harms equality by encouraging prejudice against specific groups of people. Typically, women and immigrants are the major targets. Due to the refugee crisis and political changes over the past few decades, there has been a sharp increase in anti-immigrant sentiment. The issue is currently being addressed by several governments and policy officials, who are particularly focusing on identifying and monitoring hate speech directed at immigrants. Hatred for women is a well-known form of prejudice that has persisted for a long time and typically takes the form of abuse, denigration, and discrimination against women in the workplace, in society, and in the home. The following concepts are part of the conceptual deconstruction process that applies to hate speech:
(1) it singles out specific traits in order to target particular groups or classes of people
(2) It exhibits hateful attitudes, thoughts, or behaviors.

Hate speech detection falls under the category of sentiment and emotion analysis, which can be either explicit or implicit .It is often expressed in the form of negative opinions, abusive, stereotypical, humorous, sarcastic, and sarcastic messages. For example, “let's go home” and “women's opinions don't matter”, these words are considered hate speech.However, in some cases, certain words can have a negative connotation but may not be hate speech in the full context.
For example, "I hate that they waste their time," although the word "hate" is used here, the statement is not hate speech.

3.2 Word Embedding

Word embedding is an important natural language processing (NLP) technique that seeks to convey the semantic meaning of a word.It provides a useful numerical description of the term based on its context.Words are represented by a dense vector that can be used to estimate the similarity between words.The word is represented by an N-dimensional vector appropriate to represent the word's meaning in a particular language.Word integration has been widely used in many recent NLP tasks due to its effectiveness, such as document encapsulation, text classification, speech part tagging, named entity recognition, parsing, etc.sentimentality and many other problems.The most popular pre-trained word embedding models are Google Word2Vec, Stanford GloVe and they are described as follows:

3.3 Word2vec

Word2Vec is one of the most recently used word embedding templates.It is provided by the Google Research team.Word2Vec associates each word with a vector based on the context around it from a large corpus.The training process for extracting word vectors is of two types, the Continuous Bag of Words (CBOW) model, which predicts the target word from its context, and the Skip-Gram (SG) model, which predicts the target context from a single word.gave.The word feature vector is processed and updated based on the context in which the word appears in the corpus.If word integration is well formed, similar words will appear close together in multidimensional space.The similarity between words is measured by the cosine distance between their vectors.Google has released a vector model called Google Word2Vec, which is trained on a huge corpus of over 100 billion words.

3.4 GloVe

This is a popular word embedding pattern called GloVe (Global Vector for Word Representation). GloVe learns to integrate using an unsupervised learning algorithm trained on a data warehouse to generate distributed feature vectors.During the learning process, a statistical-based matrix is constructed to represent the word-by-word occurrences of the corpus.This matrix representing word vectors.The learning process requires time and space to build the matrix, which is a very expensive process.The difference between GloVe and Word2Vec is that in the learning process, Word2Vec is a prediction-based model and GloVe is a quantity-based model.GloVe is taken from Wikipedia, web data, Twitter, and each sample is available in a variety of vector sizes.

3.5 RECURRENT NEURAL NETWORK

An important concept in RNNs is the concept of hidden states.The hidden state is a vector representing the internal state of the network at a given time step.The hidden state is updated every step based on the current input and the previous hidden state.The network output at each time step is then generated based on the current hidden state.Updating the hidden state in an RNN can be described mathematically using a set of regression equations.These equations determine how to generate the current hidden state based on the current input and the previous hidden state.The specific form of these equations depends on the architecture of the RNN, but in general they involve a combination of matrix multiplication and element-by-element nonlinear activation functions.One problem with RNNs is the gradient vanishing problem, which can arise when training the network over long sequences.To solve this problem, variants of RNNs have been developed, such as the Long-Term Short-Term Memory (LSTM) network and the Controlled Recurrence Unit (GRU), which use special control mechanisms.to selectively update and forget information in memory.These variants have shown better performance in many applications, especially when dealing with long data strings.RNNs have been successfully applied to a variety of tasks, including natural language processing, speech recognition, image annotation, and time series prediction.

LSTM (Long Short Term Memory)

LSTM is an enhanced version of Recurrent Neural Network, which is one of the deep learning models designed to collect information from a sequence of information.It differs from a feed forward neural network in that it has reverse connections.RNNs experience a gradient vanishing problem that occurs when the weights are no longer updated due to the small value of the received error function compared to the current weights in the iteration.The value disappears in very long sequences and is close to zero.This problem prevents the RNN from training.LSTM solves this problem by adding an additional interactive cell to maintain long chain dependencies.

3.6 CNN (Convolutional Neural Network)

A convolutional neural network (CNN) is a layer of an artificial deep neural network that forwards and uses a variant of multilayer perceptron designed to require minimal reprocessing.CNN can be thought of as a feature mining architecture that basically consists of only several convolutional layers with nonlinear activation functions, but is still a core component of a larger network.It must be trained with a classifier to produce useful results.Initially, CNN is most often applied for image classification.In 2014, Kim proposed the CNN model to classify sentences.Through verification tests and industry consensus, it is generally accepted that the CNN model is the ideal model that delivers both efficiency and quality in text classification tasks.

The whole model consists of four parts:

input layer, convolution layer, pooled layer and fully connected layer.The input layer is a sentence consisting of words concatenated with s, followed by a convolutional layer with some filters.During the training phase, the CNN automatically learns the filter's values based on the specific task the user wants to perform.Each filter combines the sentence matrix and produces variable-length feature maps.Then, a clustering layer is performed on each map and the maximum or average count of each feature map is recorded.Thus, a univariate feature vector is generated from all the maps and these features are concatenated to form a feature vector for the penultimate layer.Then, the final soft max layer will classify the text based on the feature vector received from the final layer.

3.7 BERT (Bidirectional encoder Representation From Transformers)

The key technical innovation of BERT is the application of Transformer two-way training, a popular model of attention, to language modeling.This contrasts with previous attempts to test a string of left-to-right text, or a combination of left-to-right and right-to-left.The results of the paper show that a two-way trained language model can have a deeper perception of language context and flow than a one-way language model.

4. Methodology

This section describes the used methodologies to handle the detection of hate speech.Mainly there are three approaches used in this study seeking to find the best classification performance.

Data Processioning

We collected a dataset from Kaggle, an open source platform. The labeled data set contained three classes namely Hate speech, Offensive language and neither. Hate speech is denoted as 0, Offensive language denoted as 1 and neither denoted as 2. We removed the special symbols from the texts. Then we converted the texts in lower case. We also used stemming to convert the words into their basic words. We checked the data set for number of data for hate speech, offensive language and neither.

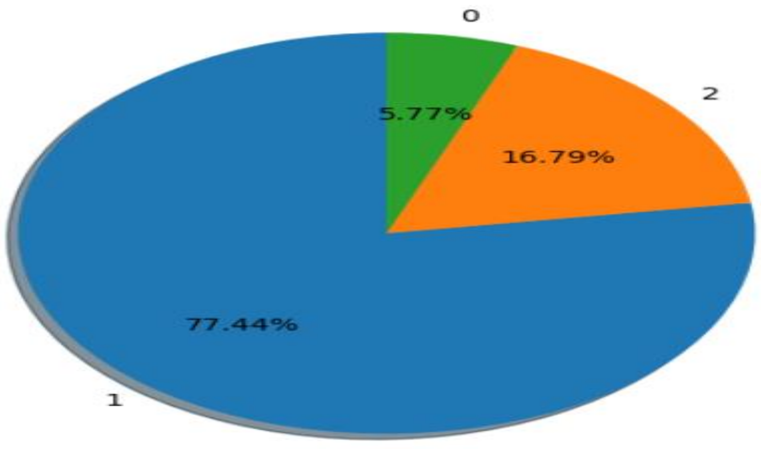


Fig. 2: Bar diagram for three classes

We divided the dataset into training and testing.We keep 80 and 20 for training and testing.With the training dataset, we trained an LSTM, CNN and BERT+CNN path. We applied encryption to prepare the data for the algorithms. Hot encoding is a process of converting text data into numeric data.Each word is given a unique digital representation in a single hot encoding. Next, we apply padding. Padding is a process of adding zeros to the beginning or end of a sentence to make all sentences the same length.Next, we apply word integration. Integration is a process of representing each word in a higher dimensional space. It is useful for finding similarities and differences between words efficiently.

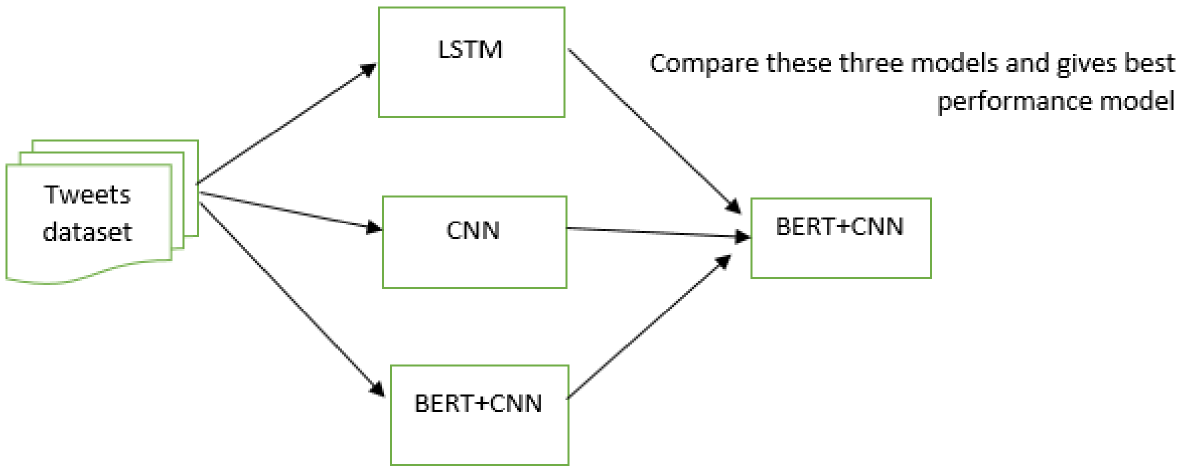


Fig 3: Block diagram of proposed model

In this paper train the dataset using LSTM, CNN and BERT+CNN pipe-lining models individually classified and then compare this three models performance.

LSTM

The LSTM model is used as the classification method. The LSTM architecture model used is Adam optimization, binary cross entropy is used as loss function, relu activation in hidden gate and sigmoid function activation for output gate. Adam's optimization can solve the problem of irregular gradients. Then, using the dropout coefficient values suggested by [23], the researcher selected the dropout coefficient values from the middle range of the model [0.2, 0.35], specifically could be 0.2 and , for this experiment. Without skip embedding, a higher probability of skipping a repeating layer will result in an overfitting, unlike regularizing an embedding layer with a dropout probability in the optimal range, the probability of skipping a repeating layer higher will lead to increased resistance to over-equipment. The binary cross entropy calculates the cross entropy loss between the actual and predicted labels. The binary classifier is selected if the two labels are in the target set of values. The sigmoid activation function takes a value from 0 to 1. The value of 0.5 is set to 1 and the model evaluation is done by comparing it with the test data set. The researchers conducted 64 training batches of word integration and LSTM

CNN

CNN text Classification consists of

- Embedding Layer: Convert word indices into dense ectors using the pre-trained embeddings.
- Convolution Layer: Apply convoutional filters of varying sizes to capture local patterns in the text.
- Max-Pooling Layer: Reduce the dimensionality of the output from convolutional layers by taking the maximum value across each feature map.
- Fully Connected Layer: Flatten the pooled features and pass them through one or more fully connected layers to learn high-level representations.
- Output Layer: Use a softmax activation function to predict thr probability of each class

BERT + CNN

BERT for text encryption

Use a per-trained BERT model to encode your text data into contextual embedding. You can tune BERT on a specific dataset for hate speech detection, or use a per-trained and pre-tuned BERT model for text classification tasks.

CNN for local feature extraction

After encoding your text with BERT, consider integrating BERT as input to the CNN component. BERT integration usually has some form of [sequence_length, embedding_dimension]. reshape your BERT embedding into a form compatible with 2D convolutional layers, you may want to add a channel dimension to make it [sequence_length, embedding_dimension, 1]. Train the association model using a labeled hate speech detection dataset. Use loss function and optimize. Fine-tune the CNN and BERT components during training. Evaluate model performance on validation dataset using appropriate metrics such as accuracy. Finally, test the model on a separate test dataset to evaluate its generalization performance in hate speech detection.Model Evaluation.

Accuracy

Accuracy is one of the most widely used performance measures and it is the ratio of total number of entries classified accurately to the total number of observations.

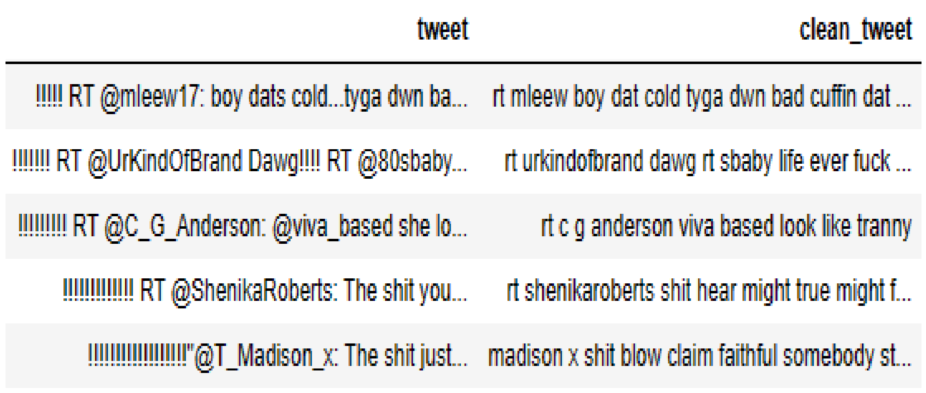
$$Accuracy = \frac{True\ Negative + True\ Positive}{True\ Negative + False\ Positive + True\ Positive + False\ Negative}$$

5. Experimental Results

5.1 Prepossessing results

Lowercase and remove special characters, punctuation, URL in lowercase stage i.e. change text to lowercase. It is very important to put the text in lowercase because the computer may evaluate the same word twice if it is written in uppercase or otherwise. When text is vectorized for feature extraction, for example, the words "boikot" and "Boikot" can be evaluated differently and assigned to different vectors. After successfully converting the text to lowercase, it will remove special characters, punctuation, and URLs. The raw data contains many examples of punctuation or special characters (@, $, *,(),[],!,:,',#,\,/) is not very important and is not understood by the computer. Therefore, its presence in the data contributes to noise and must be eliminated. This is done by removing all punctuation and special characters using regular expressions. Regular expressions are also used to strip URLs from normally unimportant content.

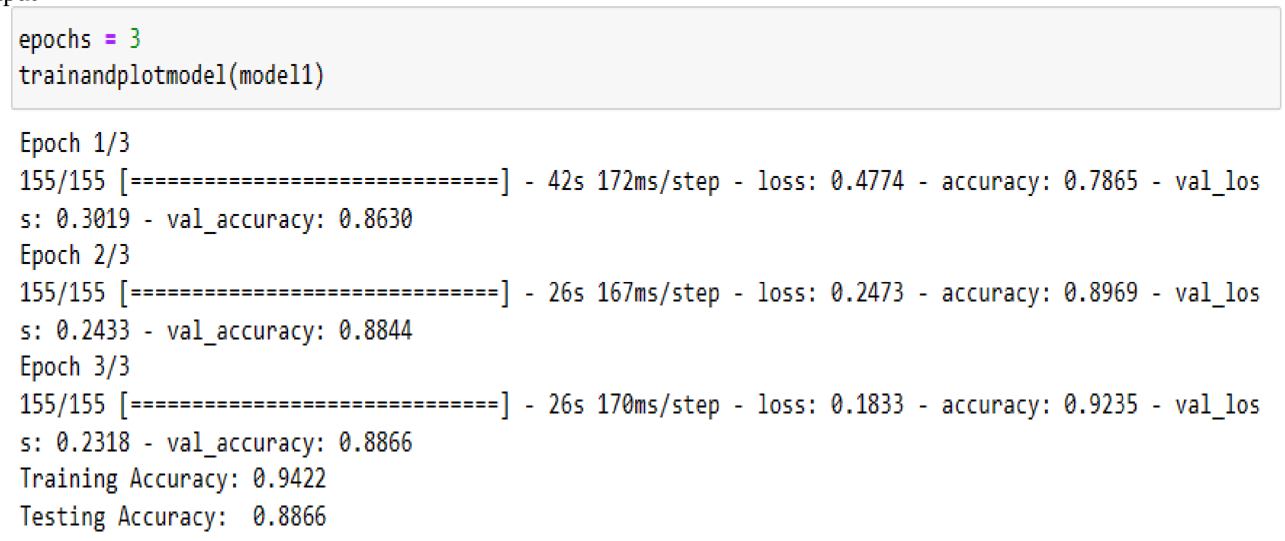
Table 1. Lowercase and removing special character, punctuation, URLs



tweet	clean_tweet
!!!!! RT @mleew17: boy dats cold...tyga dwn ba...	rt mleew boy dat cold tyga dwn bad cuffin dat ...
!!!!!!! RT @UrKindOfBrand Dawg!!!! RT @80sbaby...	rt urkindofbrand dawg rt sbaby life ever fuck ...
!!!!!!!!! RT @C_G_Anderson: @viva_based she lo...	rt c g anderson viva based look like tranny
!!!!!!!!!!!!! RT @ShenikaRoberts: The shit you...	rt shenikaroberts shit hear might true might f...
!!!!!!!!!!!!!!!!!!"@T_Madison_x: The shit just...	madison x shit blow claim faithful somebody st...

5.2 Classification Results

Output



```
epochs = 3
trainandplotmodel(model1)
```

```
Epoch 1/3
155/155 [==============================] - 42s 172ms/step - loss: 0.4774 - accuracy: 0.7865 - val_los
s: 0.3019 - val_accuracy: 0.8630
Epoch 2/3
155/155 [==============================] - 26s 167ms/step - loss: 0.2473 - accuracy: 0.8969 - val_los
s: 0.2433 - val_accuracy: 0.8844
Epoch 3/3
155/155 [==============================] - 26s 170ms/step - loss: 0.1833 - accuracy: 0.9235 - val_los
s: 0.2318 - val_accuracy: 0.8866
Training Accuracy: 0.9422
Testing Accuracy:  0.8866
```

Fig 4 : LSTM output

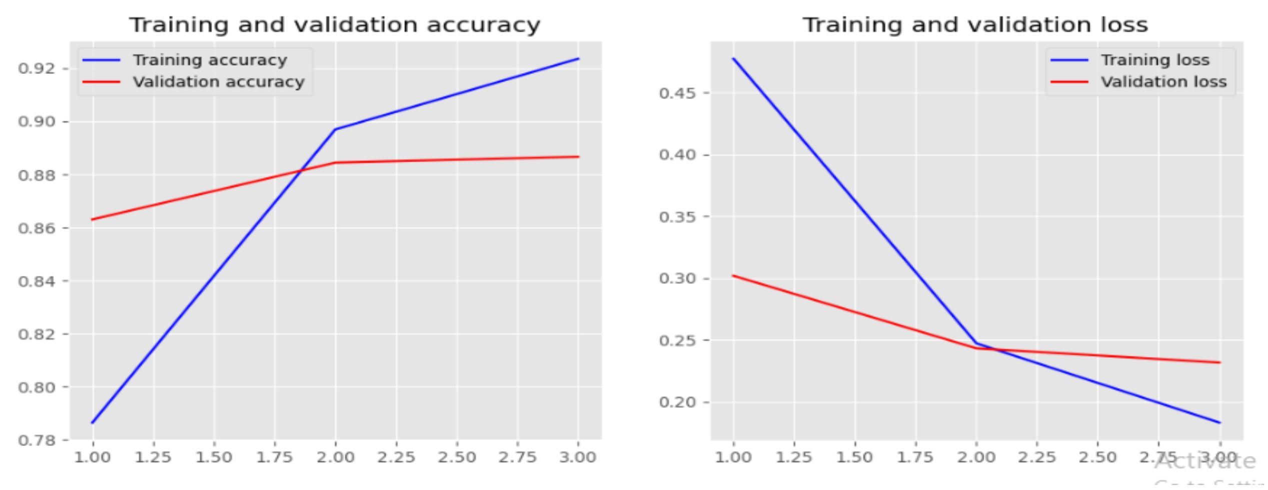
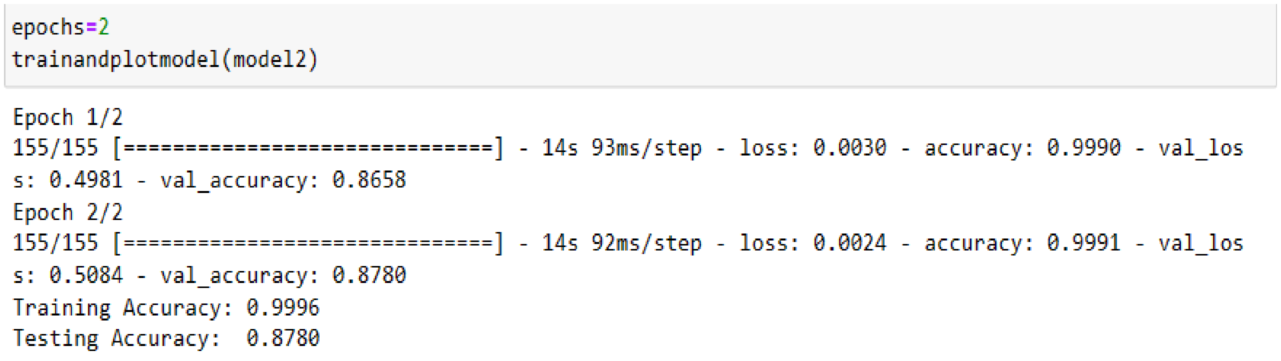


Fig 5 : LSTM graph

Output:



```
epochs=2
trainandplotmodel(model2)
```

```
Epoch 1/2
155/155 [==============================] - 14s 93ms/step - loss: 0.0030 - accuracy: 0.9990 - val_los
s: 0.4981 - val_accuracy: 0.8658
Epoch 2/2
155/155 [==============================] - 14s 92ms/step - loss: 0.0024 - accuracy: 0.9991 - val_los
s: 0.5084 - val_accuracy: 0.8780
Training Accuracy: 0.9996
Testing Accuracy:  0.8780
```

Fig 6: CNN output

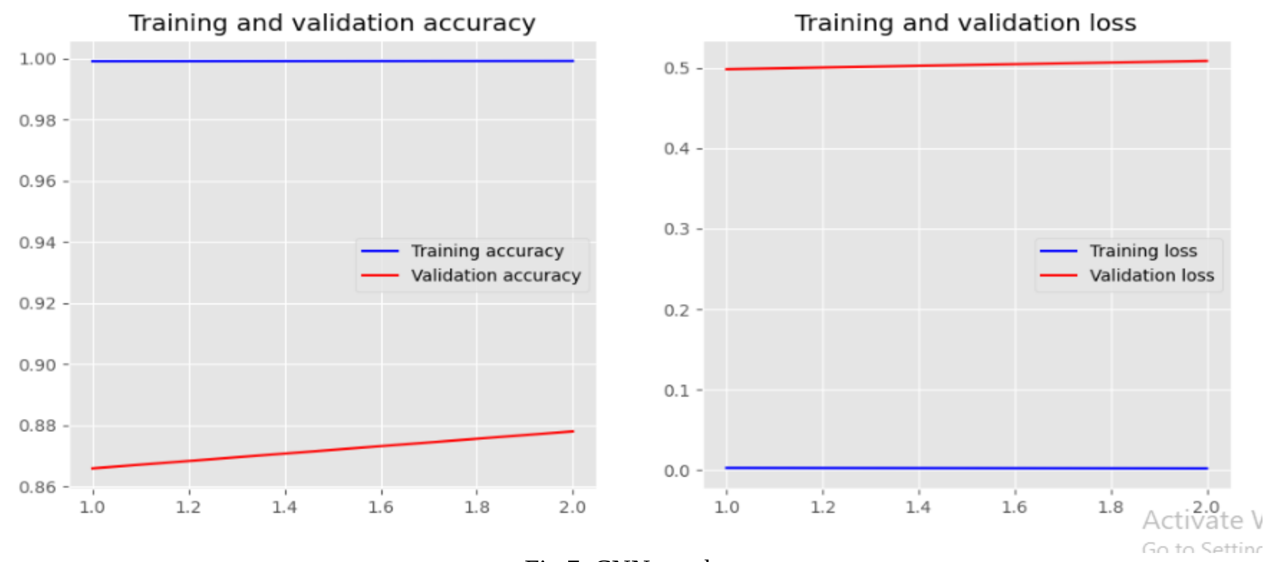
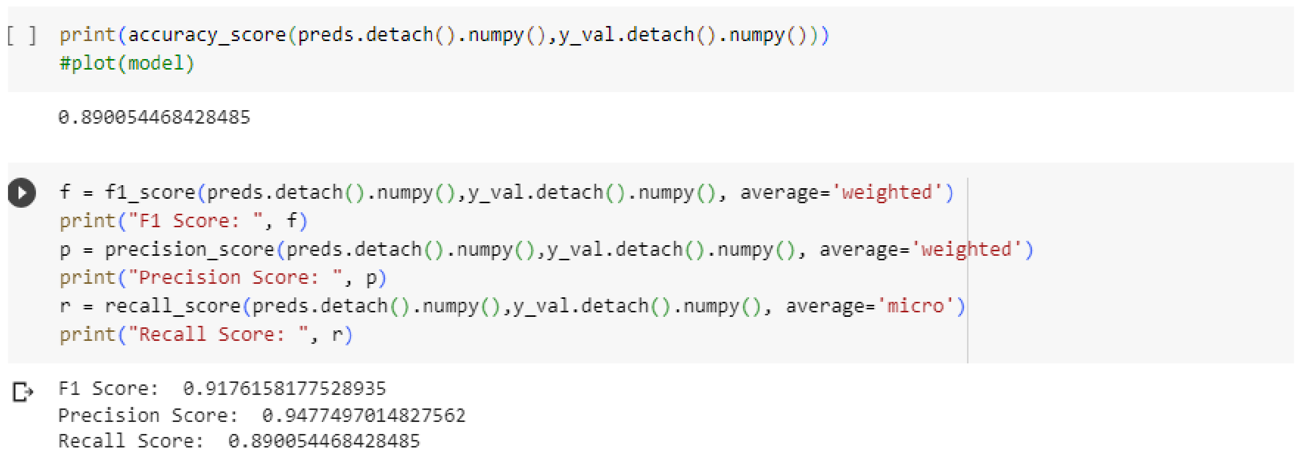


Fig 7: CNN graph



```
[ ] print(accuracy_score(preds.detach().numpy(),y_val.detach().numpy()))
    #plot(model)

    0.890054468428485

    f = f1_score(preds.detach().numpy(),y_val.detach().numpy(), average='weighted')
    print("F1 Score: ", f)
    p = precision_score(preds.detach().numpy(),y_val.detach().numpy(), average='weighted')
    print("Precision Score: ", p)
    r = recall_score(preds.detach().numpy(),y_val.detach().numpy(), average='micro')
    print("Recall Score: ", r)

    F1 Score:  0.9176158177528935
    Precision Score:  0.9477497014827562
    Recall Score:  0.890054468428485
```

Fig 8: BERT+ CNN output

Comparative Results

Model	Training Accuracy	Testing Accuracy
LSTM	0.9914	0.8866
CNN	0.9993	0.8780
BERT+CNN	0.9945	0.8954

6. Conclusion and Future Work

In this study, we took a public date set of 25,229 tweets that were labeled as hateful, offensive, and content-free. We evaluated and compared three different models LSTM, CNN and BERT+CNN The results obtained from our experiments are very promising, showing the effectiveness of the proposed models in the detection task. The results show that the BERT+CNN pipe-lining model outperforms then LSTM and CNN with an accuracy of 0.8954. We also believe that the experiments performed can be improved in various ways. First, we aim to extend our out-of-domain tests and evaluate suggested models on other hate/offensive language datasets, such as those found in (kaggle). This can help us understand how to distinguish between hate speech and abusive/offensive language.

In future studies, we look forward to expanding our approach to languages other than English, such as Bengali, Oria, Marathi, Tamil and Telugu. We also plan to develop a hybrid model that includes advanced ML, evolutionary computation, and swarm intelligence methods to improve the accuracy of hate speech detection for multilingual tweets. language.

REFERENCES:

1. Sajjad, M., Zulifqar, F., Khan, M. U. G., & Azeem, M. (2019, August). Hate speech detection using fusion approach. In *2019 International Conference on Applied and Engineering Mathematics (ICAEM)* (pp. 251-255). IEEE.
2. Arbaatun, C. N., Nurjanah, D., & Nurrahmi, H. (2022). Hate Speech Detection on Twitter through Natural Language Processing using LSTM Model. *Building of Informatics, Technology and Science (BITS)*, *4*(3), 1548-1557.
3. Paul, C., & Bora, P. (2021). Detecting hate speech using deep learning techniques. *International Journal of Advanced Computer Science and Applications (IJACSA)*, *12*(2), 619-623.
4. Alatawi, H. S., Alhothali, A., & Moria, K. (2021). Detection of hate speech using bert and hate speech word embedding with deep model. *ArXiv, abs/2111.01515*.
5. Zhou, Y., Yang, Y., Liu, H., Liu, X., & Savage, N. (2020). Deep learning based fusion approach for hate speech detection. *IEEE Access*, *8*, 128923-128929.