A PROJECT REPORT

on

Toxic Comment Classification

In Partial Fulfilment of the Requirement for the Award of

BACHELOR’S DEGREE IN

INFORMATION TECHNOLOGY

BY

Pragyesh Kumar Sinha	1606034
Shubham Gupta	1606059
Soumyajeet Addya	1606066
Srijan Mehrotra	1606229
Uditya Kumar	1606233

UNDER THE GUIDANCE OF

PROF. Ajay Kumar Jena

SCHOOL OF COMPUTER ENGINEERING
KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY
BHUBANESWAR, ODISHA -751024

CERTIFICATE

This is certify that the project entitled

Toxic Comment Classification

submitted by

Pragyesh Kumar Sinha	1606034
Shubham Gupta	1606059
Soumyajeet Addya	1606066
Srijan Mehrotra	1606229
Uditya Kumar	1606233

is a record of bonafide work carried out by them, in the partial fulfilment of the requirement for the award of Degree of Bachelor of Engineering (Computer Sci-ence & Engineering OR Information Technology) at KIIT Deemed to be university, Bhubaneswar. This work is done during the year 2019-2020, under my guidance.

Date: 31 / 03 / 2020

(Prof. Ajay Kumar Jena)
Project Guide

Acknowledgements

We are profoundly grateful to Prof. Ajay Kumar Jena for his expert guidance and continuous encouragement throughout to see that this project rights its target since its commencement to its completion.

Pragyesh Kumar Sinha
Shubham Gupta
Soumyajeet Addya
Srijan Mehrotra
Uditya Kumar

ABSTRACT

The Internet is used globally all around the world and it offers a lot of anonymity to anyone who wants to express their views on any particular subject. The anonymity that the internet offers can be harmful because it allows people to say whatever they want without any one being accountable for it. Personal attacks, bullying are really common in the comment area of YouTube, Twitter, etc.

As a result Toxic Comment Analysis has become a really vast area for people, industry and scientists to perform research on. In this project, we used Wikipedia's talk page edits to train a different multi-level classifier that detects different levels of toxicity. We will first clean the data to remove links, IP-addresses, image-links, CSS code, wiki templates, and also process the internet's shorthand. We will then use Tf-idf Vectorizer and Fasttext to vectorize the cleaned corpus to produce the vector of the text. Which we will pass through different deep learning models: Artificial Neural Networks (ANN), Convolutional Neural Networks (CNN), and Recurrent Neural Networks (RNN) with LSTM to predict the six different types of toxicity: toxic, severe toxic, obscene, threat (eg threat to kill someone) insult, identity hate (hate against a specific group of people). Finally we will use Bi-directional Gated Recurrent Unit with a Convolutional layer, to get the final accuracy and F1-score.

Keywords: NLP, Neural Networks, CNN, RNN, LSTM, Tfidf, Bi-GRU.

Contents

List of Figures

Chapter 1

Introduction

Toxic Comments are those comments that have the capability to spark some kind of controversy between - two people, two groups of people, or a person and a group of people. It generally consists of rude things about a specific person or a group of person in general. It can also be a radical idea that can't be supported by a lawful society or that doesn't bring about any positive changes in society. The Internet has become a common grounds for preaching such toxic comments, simply because of the anonymity that the internet grants.

People on the Internet can say anything they want and get away with it without having any kind of accountability for what they say. And this kind of toxic comments is not only harmful to the proper usage of the internet but also harmful for the people on the internet. People on the internet might get bullied, threatened or might face racism and bigotry for their identity, color, race, etc. This comment not only are a nuisance but also can be mentally taxing for people who face it.Thus for properly using the internet without facing any kind of toxic comments, those comments should be recognized and removed. It can also be a daunting task if proper techniques and tools are not used. In this project, we will use classification tools to recognize toxic comments.

Classification is the process or action of classifying something, in which things are put into categories. In this project, there are six types of toxic categories in which we can put a comment in toxic, severe_toxic, threat, identity_hate, obscene, insults. A comment can fall under one or many of these categories. If a comment doesn't fall under any of these categories then the comment must be free of any kind of toxicity and is clean. There are many different kinds of classification machine learning models that can be used to classify this: SVM, Naive Bayes, KNN, Decision Tree, Random forest, etc. In this project, we will use different kinds of Neural Networks for the classification of the comments.

Neural Networks are inspired by the networks of neurons present in our brain. A neural network is a connection of neurons or nodes. The connections between two neurons are modeled as weights which can be positive or negative, based on if we need excitatory or inhibitory connections. There are many different types of Neural Networks that are used for classification: Neural Networks, Convolutional Neural Networks, Recurrent Neural Networks with LSTM or GRU. We can also combine different models to make models such as GRU with Convolutional Networks, LSTM with Convolutional Networks, etc

Chapter 2

Literature Survey

Many previous research works have been done on Toxic Comment Classification. Many studies have used SLM like Linear Regression, SVM, NB to classify the toxicity of the comments, also many research papers try to see if there is any difference between classifying the data before preprocessing and after preprocessing. Many even try to clean the data, like remove links, ip addresses, etc, many don't.

In this project, learning from all those who researched it previously, we will be using NN to build the classifiers, we will also clean the models of not useful stuff like Links, IP addresses. And also pre-process it using three different pre-processor.

Besides classic binary classification tasks, related work examines different viewpoints of toxic languages, such as "racism" (Greevy and Smeaton, 2004; Waseem, 2016; Kwok and Wang, 2013) and "sexism" (Waseem and Hovy, 2016; Jha and Mamidi, 2017), or the hardness of toxicity (David-son et al., 2017; Sharma et al., 2018).

From this we got the inspiration to classify the data using Deep Learning Models instead of simple Binary Classification or Shallow Learning Models

Chapter 3

Software Requirements Specification

3.1 Tools

3.1.1 Jupyter notebook

The Jupyter Notebook is an open source application that can run python code and display the output immediately. It also uses RELP which makes it flexible to use and really easy to make ML models and rum ML projects.

3.2 Framework

3.2.1 NumPy

NumPy is a library that supports arrays and many other numerical functions which are required during the pre-processing of the data. It can be used to manipulate large matrices which is really helpful for performing many different numerical functions

3.2.2 Pandas

Pandas is used to manipulate data in the data frame. It is really helpful in the data frame manipulation. It imports and exports the data in a data frame.

3.2.3 Matplotlib

Matplotlib is used to plot graphs. It is really helpful to analyse the data to get insight on the data with the help of graphs. We can plot a variety of graphs using this library, we can plot histogram, bar graph, box plot, violin plot, etc.

3.2.4 Keras

Keras is an open-source neural-network library written in Python. It uses the back- end framework of TensorFLow. Thus it is helpful in making Deep Learning models like NN, CNN, RNN, LSTM, GRU, etc. It makes the work of making the architecture of the model relatively easy.

3.2.5 Tf-Idf Vectorizer

TF-IDF is a process which assigns a number to the word in a text depending on the frequency of times the text appears on the comment and also how many times it appears on the documents. It vectoriss the data based on the frequency of times the word appears on the model.

Chapter 4

Requirement Analysis

4.1 Vectorization

In Machine Learning the features are numerical data on which mathematical operations can be performed to predict the results. But in our project, the main data is text data. Thus the text data has to be converted to numerical data to predict toxicity. The task of converting the text data to numerical data is done by a vectorizer. Vectorizers are of many types: Tfidf Vectorizer, Countvectorizer, Google Word2Vec, FastText, etc.

4.2 Neural Networks

Neural Networks work with the base concept that a layer of neurons gets numerical data from features which are added by multiplying and adding different weights. The result is then passed through a formula, eg: Sigmoid or Relu, etc. The result is passed to another layer of Neurons. Thus the process continues. In this project, we will use Neural Networks and variations of Neural Networks like:

1. CNN: Which firsts embeds the numerical data to make it more compact and then runs the similar concept of Basic Neural Networks.

2. RNN with LSTM: LSTM has a special property with the help of which it can remember the data that came before the current data, thus making the processing of the data more accurate. RNN is a sequence of neural network blocks that are linked to each other like a chain. Thus RNN is better at dealing with sequential data and since our data is textual data - and textual data is sequential - we can predict it will perform better than the Neural Networks or CNN.

Chapter 5

Project Planning

5.1 DATA ANALYSIS

The data contains 150 K rows with 8 columns: ID, comment_text, toxic, severer_toxic, obscene, threat, insult, identity_hate. The last six columns have values either 0 or 1 which denote if the comment text is toxic or not. The data is going to be explored to find correlations between unique words and toxicity and not on sentences and toxicity.

5.2 DATA CLEANING

The comment text is going to be explored to find un-important text in the comments and this un-important text will be cleaned

5.3 DATA PREPROCESSING

We can preprocess the data in three different ways: Vectorizing the data using Tfidf Vectorizer. Using fast text word embedding of Wikipedia. Using badwords.csv predefined by Google.

5.4 MODEL TRAINING AND TESTING

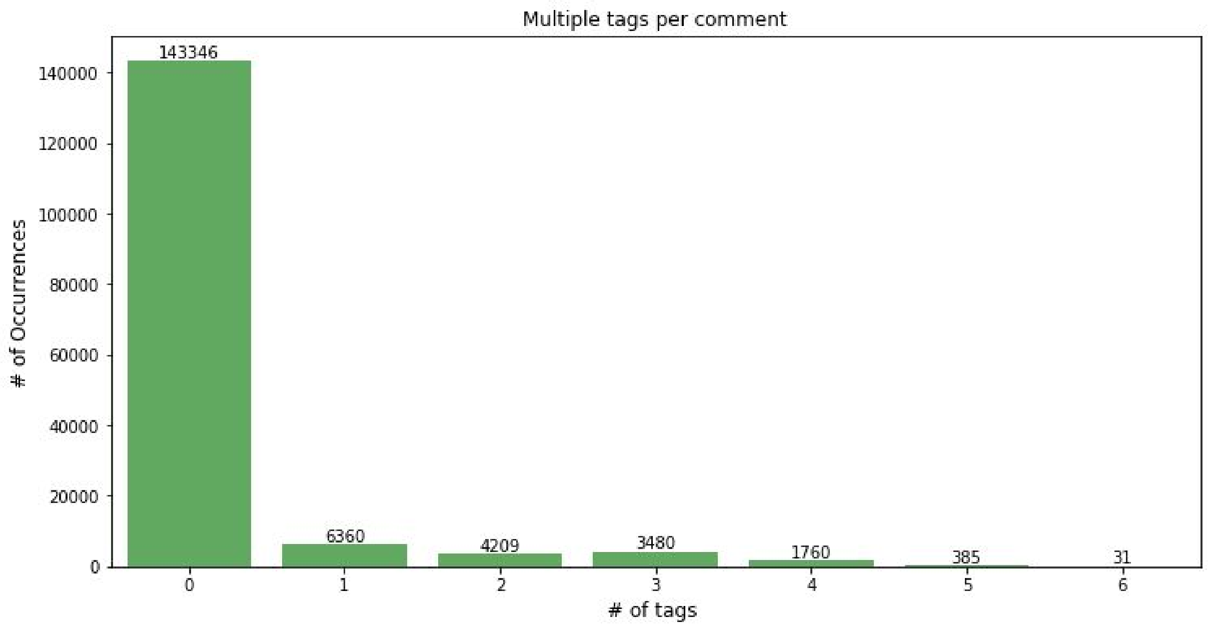
The four types of preprocessed data will be used to train and test three different deep learning models: Neural Networks, Convolutional Neural Networks and RNN under Long Short Term Memory, and Bi-GRU (using CNN).

Chapter 6

Implementation

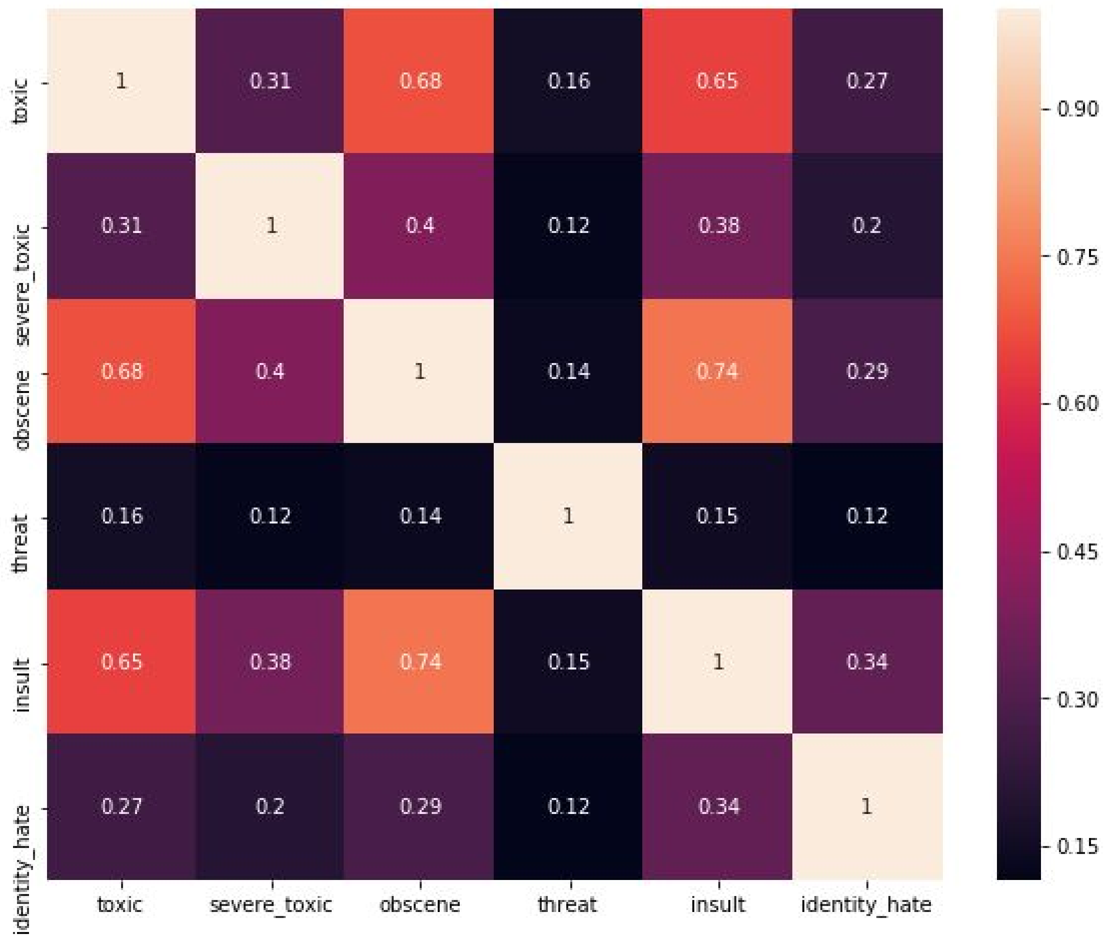
6.1 ANALYSIS

The data was explored thoroughly to see if there are any correlations between no of words per comment and toxicity and it was plotted using matplotlib. The relationship between different types of toxicity was also found out and it was seen that every comment in severe_toxic class was also toxic.



Multiple toxicity Bar Graph

We also found out the correlations between the different type of toxicity. Correlations between toxic-obscene, toxic-threat, and obscene-threat is high compared to other combinations. But since correlation is not causation, we can't conclude that if a comment is toxic it will also be obscene and vice-versa.



Correlation Heat Map

6.2 CLEANING

The comment text contains both lower and upper case letters, links, IP addresses, wiki tokens, CSS templates and short-hands. In this step we removed the IP addresses, wiki tokens, links and CSS templates, and then converted the whole text to lowercase and converted the short-hands to long-hands. This text cleaning is done using 'RE' library

Code of cleaning:

```
def clean_text(cmnt_text, clean_wiki_tokens = True):
    cmnt_text = cmnt_text.lower()
    #removing links
    cmnt_text = re.sub(r"https?:\/\/(www\.)?[-a-zA-Z0-9@:%._\+~#=]{2,256}\.[a-z]{2,6}\b([-a-zA-Z0-9@:%_\+.~#?&//=]*)", "", cmnt_text)
    #removing IP addresses
                                                                              cmnt_text                    =
re.sub(r"(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)(\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)){3}",
"", cmnt_text)

    if clean_wiki_tokens:
        #removing images
        cmnt_text = re.sub(r"image:[a-zA-Z0-9]*\.jpg", " ", cmnt_text)
        cmnt_text = re.sub(r"image:[a-zA-Z0-9]*\.png", " ", cmnt_text)
        cmnt_text = re.sub(r"image:[a-zA-Z0-9]*\.gif", " ", cmnt_text)
        cmnt_text = re.sub(r"image:[a-zA-Z0-9]*\.bmp", " ", cmnt_text)

        #removing CSS
        cmnt_text = re.sub(r"#([A-Fa-f0-9]{6}|[A-Fa-f0-9]{3})", " ",cmnt_text)
        cmnt_text = re.sub(r"\{\|[^\}]*\|\}", " ", cmnt_text)

        #removing templates
        cmnt_text = re.sub(r"\[?\[user:.*\]", " ", cmnt_text)
        cmnt_text = re.sub(r"\[?\[wikipedia:.*\]", " ", cmnt_text)
        cmnt_text = re.sub(r"\[?\[special:.*\]", " ", cmnt_text)
        cmnt_text = re.sub(r"\[?\[category:.*\]", " ", cmnt_text)

    cmnt_text = re.sub(r"what's", "what is ", cmnt_text)
    cmnt_text = re.sub(r"\'s", " ", cmnt_text)
    cmnt_text = re.sub(r"\'ve", " have ", cmnt_text)
    cmnt_text = re.sub(r"can't", " cannot ", cmnt_text)
    cmnt_text = re.sub(r"n't", " not ", cmnt_text)
    cmnt_text = re.sub(r"i'm", " i am ", cmnt_text)
    cmnt_text = re.sub(r"\'m", " i am ", cmnt_text)
    cmnt_text = re.sub(r"\'re", " are ", cmnt_text)
    cmnt_text = re.sub(r"\'d", " would ", cmnt_text)
    cmnt_text = re.sub(r"\'ll", " will ", cmnt_text)
    cmnt_text = re.sub(r",", " ", cmnt_text)
    cmnt_text = re.sub(r"\.", " ", cmnt_text)
    cmnt_text = re.sub(r"!", " ! ", cmnt_text)
    cmnt_text = re.sub(r"\/", " ", cmnt_text)
    cmnt_text = re.sub(r"\?", " ? ", cmnt_text)
    cmnt_text = re.sub(r"\!", " ! ", cmnt_text)
    cmnt_text = re.sub(r"\"", " ", cmnt_text)
    cmnt_text = re.sub(r"\^", " ^ ", cmnt_text)
    cmnt_text = re.sub(r"\+", " + ", cmnt_text)
    cmnt_text = re.sub(r"\-", " - ", cmnt_text)
    cmnt_text = re.sub(r"\=", " = ", cmnt_text)
    cmnt_text = re.sub(r"'", " ", cmnt_text)
    cmnt_text = re.sub(r"(\d+)(k)", r"\g<1>000", cmnt_text)
    cmnt_text = re.sub(r":", " : ", cmnt_text)
    cmnt_text = re.sub(r" e g ", " eg ", cmnt_text)
    cmnt_text = re.sub(r" b g ", " bg ", cmnt_text)
    cmnt_text = re.sub(r" u s ", " american ", cmnt_text)
    cmnt_text = re.sub(r"\0s", "0", cmnt_text)
    cmnt_text = re.sub(r" 9 11 ", "911", cmnt_text)
    cmnt_text = re.sub(r"e - mail", "email", cmnt_text)
    cmnt_text = re.sub(r"j k", "jk", cmnt_text)
    cmnt_text = re.sub(r"\s{2,}", " ", cmnt_text)
    cmnt_text = re.sub(r"\n", " ", cmnt_text)


    return(cmnt_text)
```

6.3 Preprocessing

The comment text needs to be converted in vectors which can be used to train the models. We primarily used three different types of pre-processing:

1. Vectorizing the data using Tfidf Vectorizer. It depends on the number of times a 'term' or word is present in a comment and how many times the 'term' occurs in the whole corpus of comments.

2. Fast-Text embedding, which is a predefined file which contains the vector for many different words. It is made by Facebook AI or FBAI.

3. Bad words. It is a pre-defined csv file with some of the most commonly used bad words. This csv file was used to Tokenize the words of the corpus.

6.4 Training and Testing

1. Neutral Networks

The cleaned, Tf Idf Vectorizer data was used to train Neural Networks with 4 layers and 2 drop- outs.

Code of NN:

```
model = Sequential()
model.add(Dense(1000, input_dim = 500, activation='relu'))
model.add(Dropout(0.3))
model.add(Dense(250, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(32, activation='relu'))
model.add(Dense(num_class, activation='sigmoid'))
adam = optimizers.Adam(lr=0.001, beta_1=0.9, beta_2=0.999, epsilon=1e-08, decay=0.0)
model.compile(loss='binary_crossentropy', optimizer=adam, metrics=['accuracy'])
model.summary()
```

2. Convolutional Neural Networks

The cleaned data pre- processed using Fast- Text was used to train CNN with 1 Embedding, 2 Convolution, 1 drop-out and 2 layers.

Code of CNN:

```
print("training CNN ...")
model = Sequential()
model.add(Embedding(nb_words, embed_dim,
        weights=[embedding_matrix], input_length=max_seq_len, trainable=False))
model.add(Conv1D(num_filters, 7, activation='relu', padding='same'))
model.add(MaxPooling1D(2))
model.add(Conv1D(num_filters, 7, activation='relu', padding='same'))
model.add(GlobalMaxPooling1D())
model.add(Dropout(0.5))
model.add(Dense(32, activation='relu', kernel_regularizer=regularizers.l2(weight_decay)))
model.add(Dense(num_classes, activation='sigmoid'))  #multi-label (k-hot encoding)

adam = optimizers.Adam(lr=0.001, beta_1=0.9, beta_2=0.999, epsilon=1e-08, decay=0.0)
model.compile(loss='binary_crossentropy', optimizer=adam, metrics=['accuracy'])
model.summary()
```

3. Recurrent Neural Networks

The cleaned data pre- processed using BadWords.csv is used to train RNN with LSTM (Long Short Term Memory) with 1 Embedding and 4 dense layers.

Code of RNN:

```
embedding_vecor_length = 300
model = Sequential()
model.add(Embedding(5000, embedding_vecor_length, input_length=X_train.shape[1]))
model.add(LSTM(256))
model.add(Dense(128, activation='relu'))
model.add(Dense(64, activation='relu'))
model.add(Dense(64, activation='relu'))
model.add(Dense(6, activation='sigmoid'))

# Compiling Model using optimizer
opt = Adam(lr=1e-3)
model.compile(loss='binary_crossentropy',optimizer=opt,metrics=['accuracy'])
model.summary()
```

4. Bi-GRU-CNN

The cleaned data is preprocessed using Fast Text and a bidirectional GRU using CNN layer is used to classify the data.

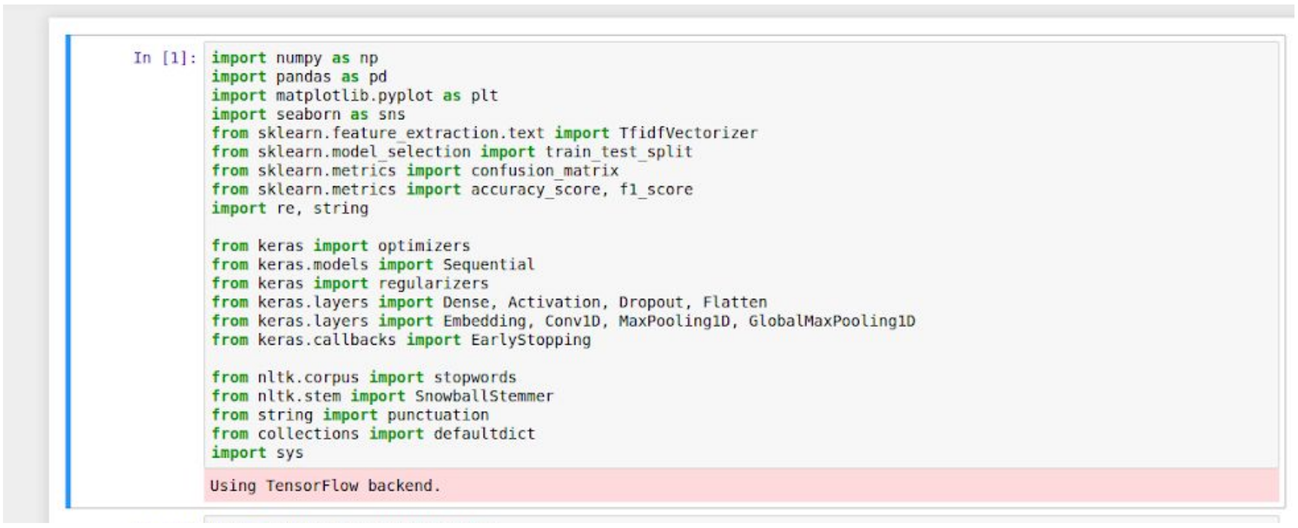
Code of Bi-GRU_CNN:

```
from keras.optimizers import Adam, RMSprop
from keras.callbacks import EarlyStopping, ModelCheckpoint, LearningRateScheduler
from keras.layers import GRU, BatchNormalization, Conv1D, MaxPooling1D

file_path = "bi_gru_cnn.hdf5"
check_point = ModelCheckpoint(file_path, monitor = "val_loss", verbose = 1,
                              save_best_only = True, mode = "min")
ra_val = RocAucEvaluation(validation_data=(X_val, y_val), interval = 1)
early_stop = EarlyStopping(monitor = "val_loss", mode = "min", patience = 5)

def build_model(lr = 0.0, lr_d = 0.0, units = 0, dr = 0.0):
    inp = Input(shape = (max_len,))
    x = Embedding(max_features, embed_size, weights = [embedding_matrix], trainable =
False)(inp)
    x = SpatialDropout1D(dr)(x)

    x = Bidirectional(GRU(units, return_sequences = True))(x)
    x = Conv1D(64, kernel_size = 2, padding = "valid", kernel_initializer = "he_uniform")(x)
    avg_pool = GlobalAveragePooling1D()(x)
    max_pool = GlobalMaxPooling1D()(x)
    x = concatenate([avg_pool, max_pool])

    x = Dense(6, activation = "sigmoid")(x)
    model = Model(inputs = inp, outputs = x)
    model.compile(loss = "binary_crossentropy", optimizer = Adam(lr = lr, decay = lr_d),
metrics = ["accuracy"])
    model.summary()
    history = model.fit(X_train, y_train, batch_size = 128, epochs = 4, validation_data =
(X_val, y_val),
                        verbose = 1, callbacks = [check_point, early_stop])
    model = load_model(file_path)
    return model
```

Chapter 7

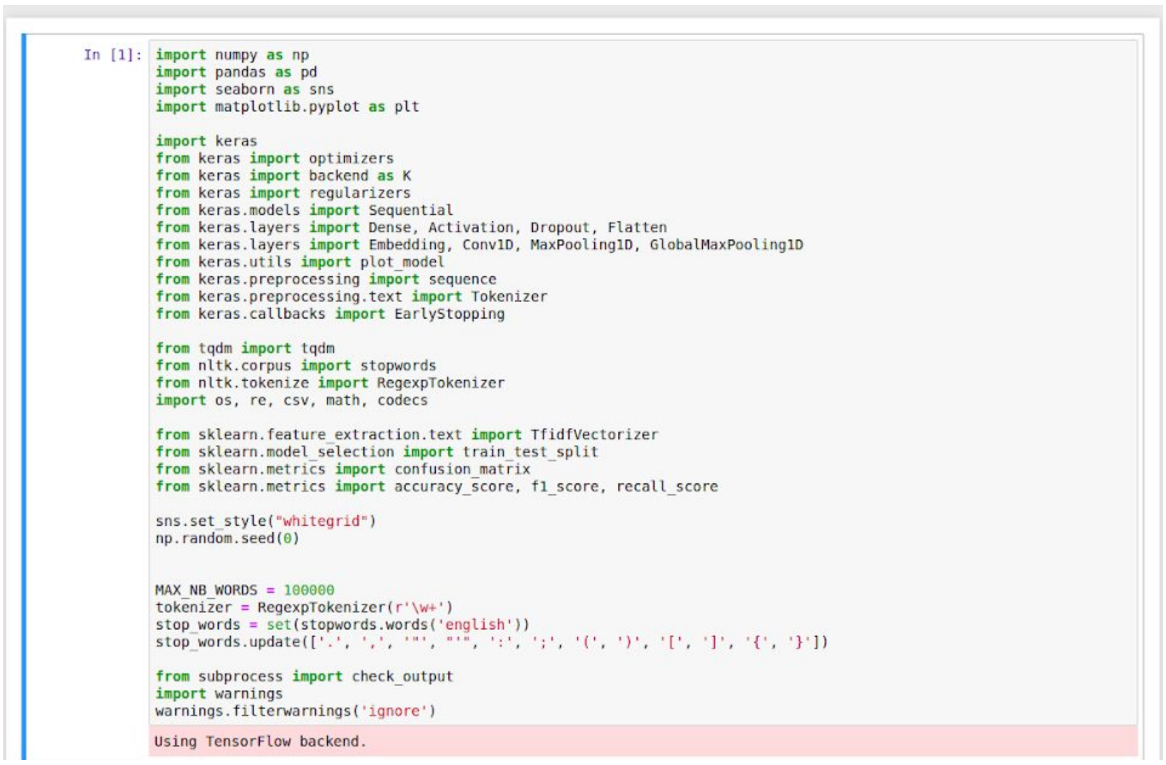
ScreenShots

Basic Imports of NN



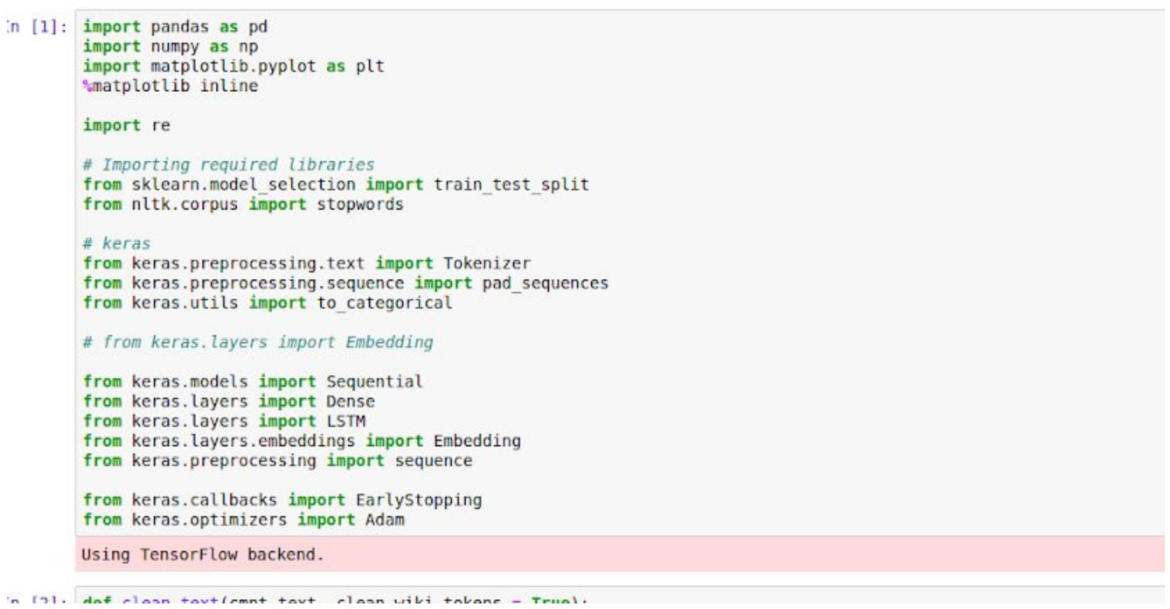
```
In [1]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        import seaborn as sns
        from sklearn.feature_extraction.text import TfidfVectorizer
        from sklearn.model_selection import train_test_split
        from sklearn.metrics import confusion_matrix
        from sklearn.metrics import accuracy_score, f1_score
        import re, string

        from keras import optimizers
        from keras.models import Sequential
        from keras import regularizers
        from keras.layers import Dense, Activation, Dropout, Flatten
        from keras.layers import Embedding, Conv1D, MaxPooling1D, GlobalMaxPooling1D
        from keras.callbacks import EarlyStopping

        from nltk.corpus import stopwords
        from nltk.stem import SnowballStemmer
        from string import punctuation
        from collections import defaultdict
        import sys

        Using TensorFlow backend.
```

Basic Imports of CNN



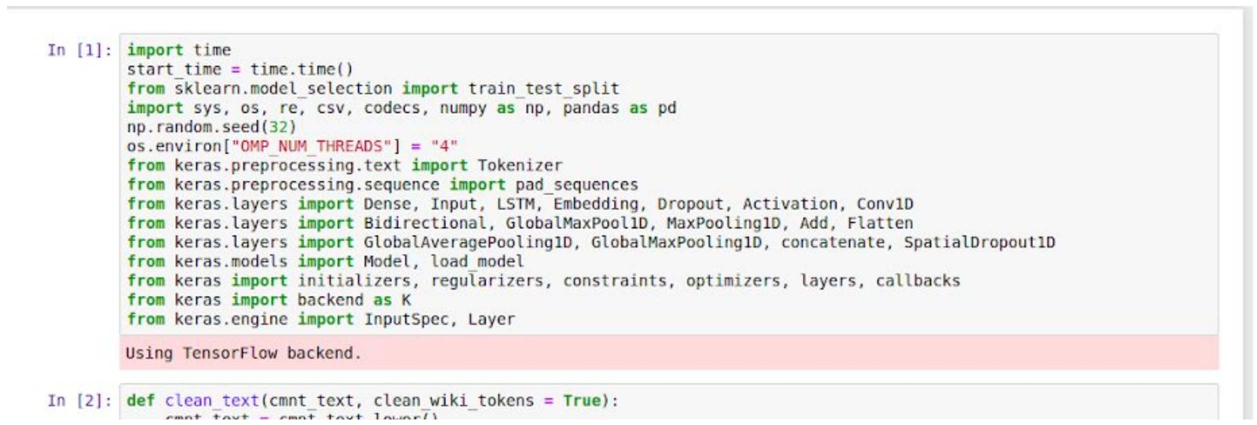
```
In [1]: import numpy as np
        import pandas as pd
        import seaborn as sns
        import matplotlib.pyplot as plt

        import keras
        from keras import optimizers
        from keras import backend as K
        from keras import regularizers
        from keras.models import Sequential
        from keras.layers import Dense, Activation, Dropout, Flatten
        from keras.layers import Embedding, Conv1D, MaxPooling1D, GlobalMaxPooling1D
        from keras.utils import plot_model
        from keras.preprocessing import sequence
        from keras.preprocessing.text import Tokenizer
        from keras.callbacks import EarlyStopping

        from tqdm import tqdm
        from nltk.corpus import stopwords
        from nltk.tokenize import RegexpTokenizer
        import os, re, csv, math, codecs

        from sklearn.feature_extraction.text import TfidfVectorizer
        from sklearn.model_selection import train_test_split
        from sklearn.metrics import confusion_matrix
        from sklearn.metrics import accuracy_score, f1_score, recall_score

        sns.set_style("whitegrid")
        np.random.seed(0)


        MAX_NB_WORDS = 100000
        tokenizer = RegexpTokenizer(r'\w+')
        stop_words = set(stopwords.words('english'))
        stop_words.update(['.', ',', '"', "'", ':', ';', '(', ')', '[', ']', '{', '}'])

        from subprocess import check_output
        import warnings
        warnings.filterwarnings('ignore')

        Using TensorFlow backend.
```

Basic Imports of RNN



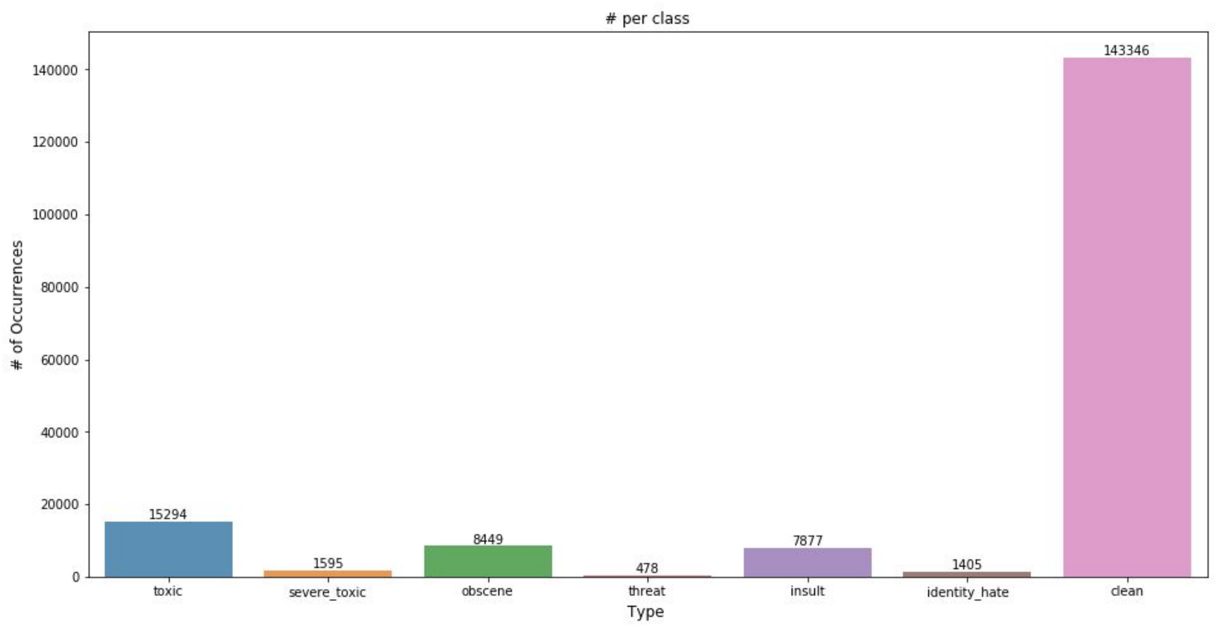
```
In [1]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        %matplotlib inline

        import re

        # Importing required libraries
        from sklearn.model_selection import train_test_split
        from nltk.corpus import stopwords

        # keras
        from keras.preprocessing.text import Tokenizer
        from keras.preprocessing.sequence import pad_sequences
        from keras.utils import to_categorical

        # from keras.layers import Embedding

        from keras.models import Sequential
        from keras.layers import Dense
        from keras.layers import LSTM
        from keras.layers.embeddings import Embedding
        from keras.preprocessing import sequence

        from keras.callbacks import EarlyStopping
        from keras.optimizers import Adam

        Using TensorFlow backend.

In [2]: def clean_text(cmnt_text, clean_wiki_tokens = True):
```

Basic Imports of BI_GRU_CNN

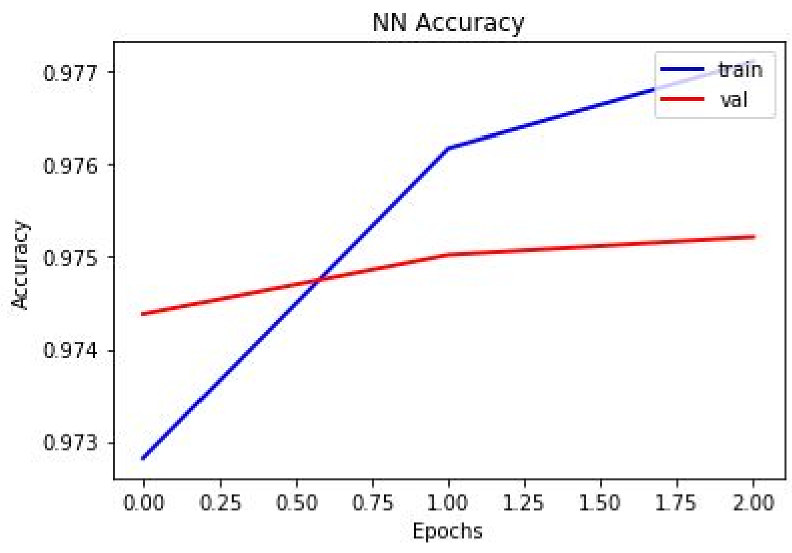


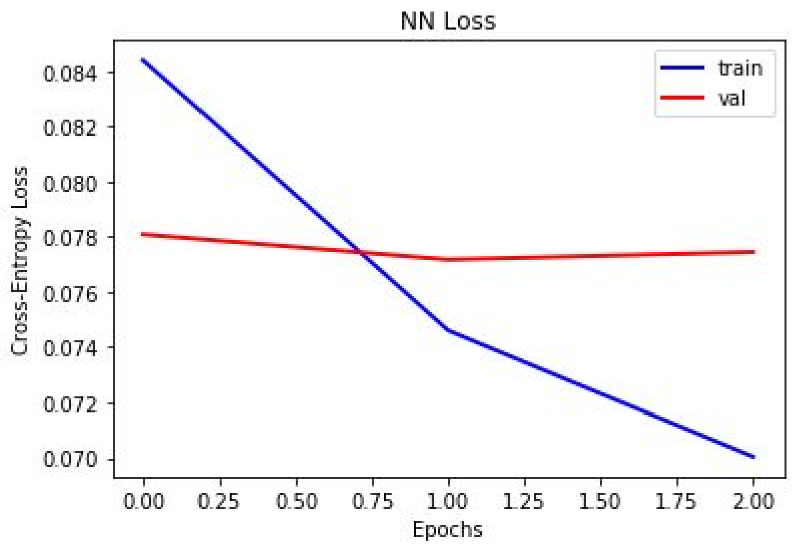
```
In [1]: import time
        start_time = time.time()
        from sklearn.model_selection import train_test_split
        import sys, os, re, csv, codecs, numpy as np, pandas as pd
        np.random.seed(32)
        os.environ["OMP_NUM_THREADS"] = "4"
        from keras.preprocessing.text import Tokenizer
        from keras.preprocessing.sequence import pad_sequences
        from keras.layers import Dense, Input, LSTM, Embedding, Dropout, Activation, Conv1D
        from keras.layers import Bidirectional, GlobalMaxPool1D, MaxPooling1D, Add, Flatten
        from keras.layers import GlobalAveragePooling1D, GlobalMaxPooling1D, concatenate, SpatialDropout1D
        from keras.models import Model, load_model
        from keras import initializers, regularizers, constraints, optimizers, layers, callbacks
        from keras import backend as K
        from keras.engine import InputSpec, Layer

        Using TensorFlow backend.

In [2]: def clean_text(cmnt_text, clean_wiki_tokens = True):
```

Bar Plot for Clean and Toxic

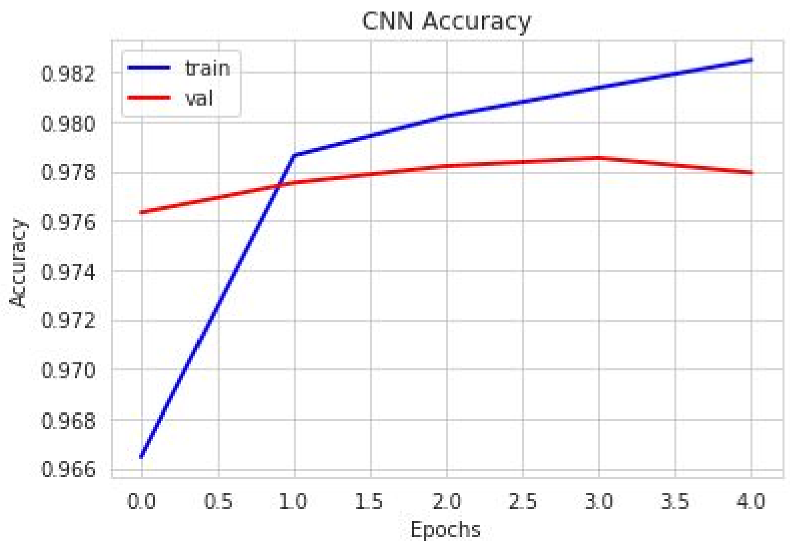


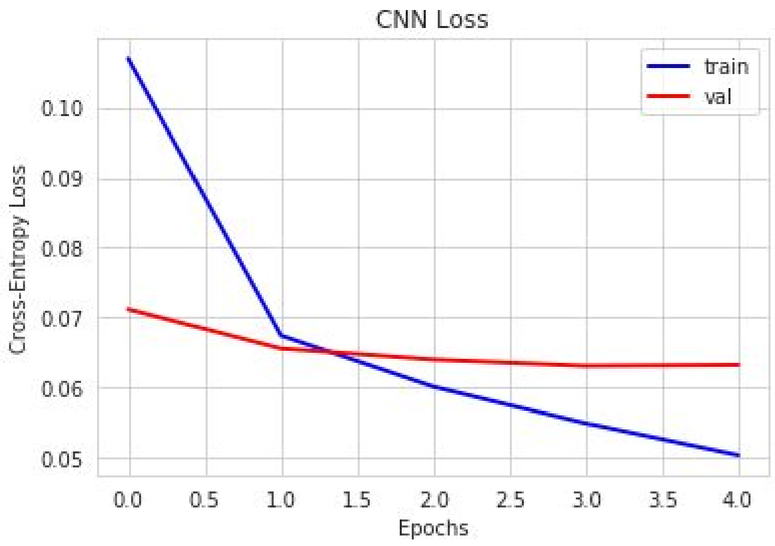
NN accuracy and loss



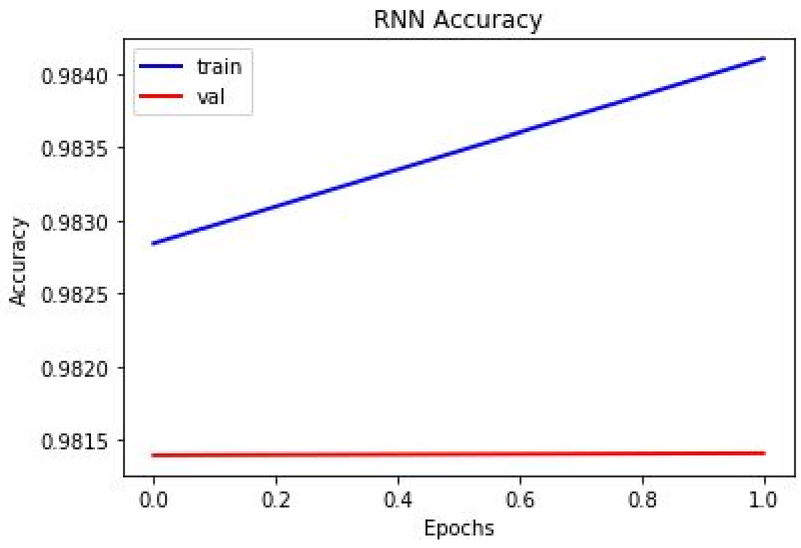


CNN accuracy and loss

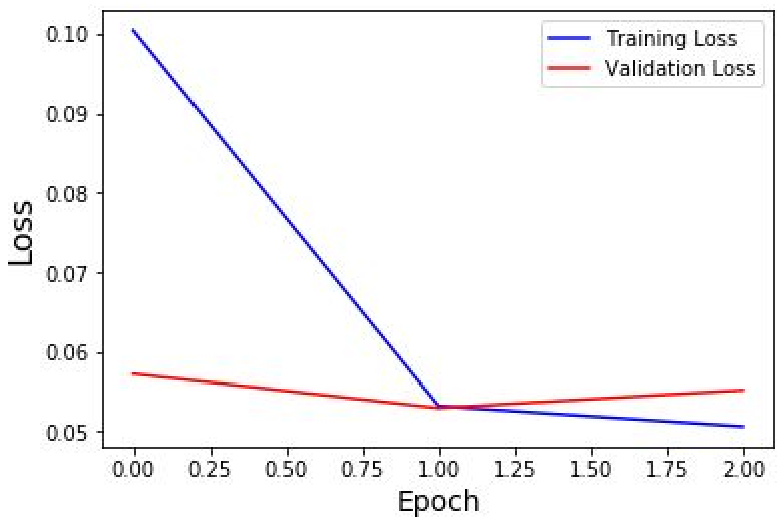




RNN Accuracy and loss

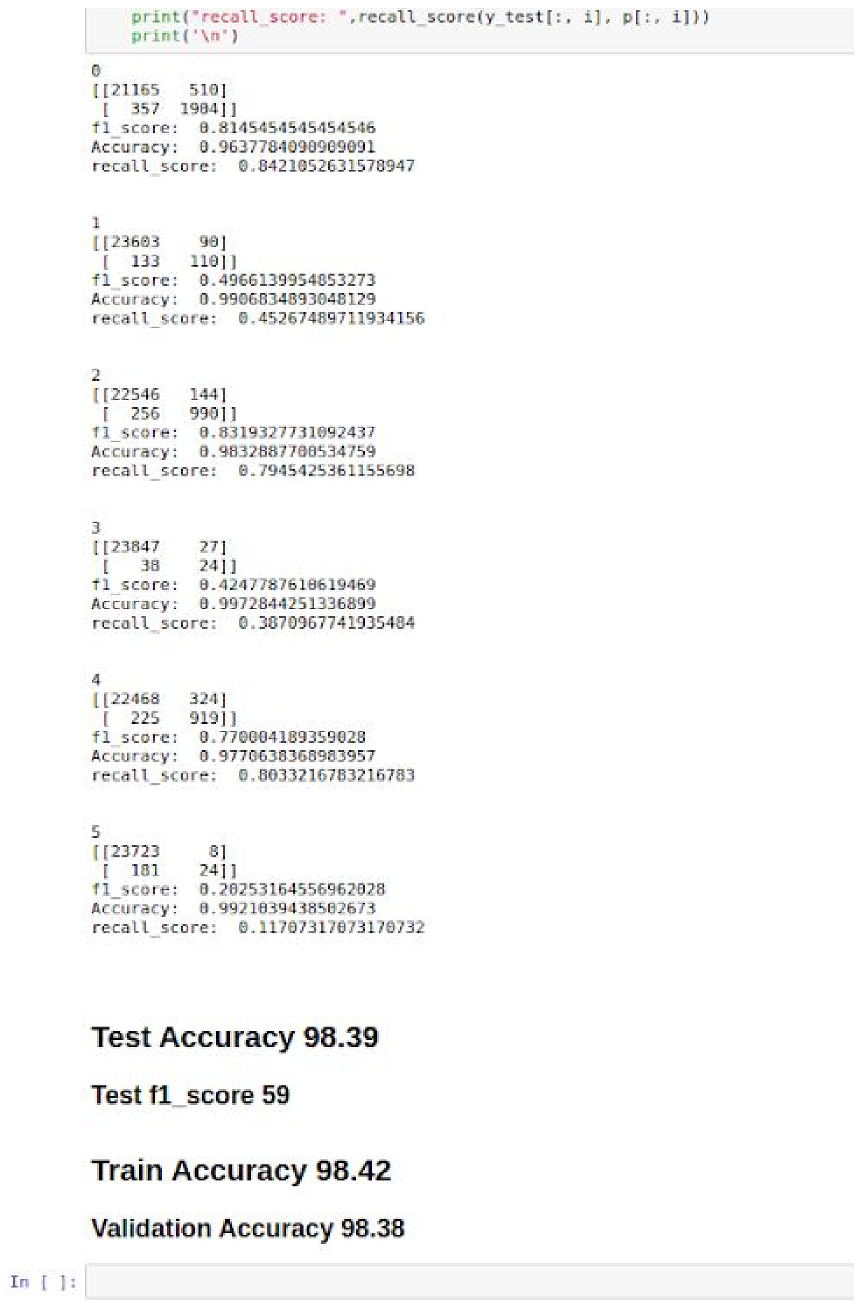


RNN Loss



Training Loss
Validation Loss
Loss
Epoch
0.10
0.09
0.08
0.07
0.06
0.05
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00

Bi-GRU-CNN Accuracy and F1-Score



```
    print("recall_score: ",recall_score(y_test[:, i], p[:, i]))
    print('\n')
```

```
0
[[21165   510]
 [  357  1984]]
f1_score:  0.8145454545454546
Accuracy:  0.9637784090909091
recall_score:  0.8421052631578947


1
[[23603    90]
 [  133   110]]
f1_score:  0.4966139954853273
Accuracy:  0.9906834893048129
recall_score:  0.45267489711934156


2
[[22546   144]
 [  256   990]]
f1_score:  0.8319327731092437
Accuracy:  0.9832887700534759
recall_score:  0.7945425361155698


3
[[23847    27]
 [   38    24]]
f1_score:  0.4247787610619469
Accuracy:  0.9972844251336899
recall_score:  0.3870967741935484


4
[[22468   324]
 [  225   919]]
f1_score:  0.770004189359028
Accuracy:  0.9770638368983957
recall_score:  0.8033216783216783


5
[[23723     8]
 [  181    24]]
f1_score:  0.20253164556962028
Accuracy:  0.9921039438502673
recall_score:  0.11707317073170732
```

Test Accuracy 98.39

Test f1_score 59

Train Accuracy 98.42

Validation Accuracy 98.38

```
In [ ]:
```

Chapter 8

Conclusion

In this project we saw that the deep learning models gave good results on a cleaned text data. Bi-GRU using a convolutional layer gave the best results with 98.4% accuracy. The second best result was given by RNN with 97% accuracy. CNN gave 96.6% accuracy. The Neural Network gave only 94.4% accuracy. The CNN and RNN gave 0 F1 score in threat, identity_hate, whereas this wasn't the case in Bi-GRU and NN. From the Neural Network accuracy graph we can say that Neural Network overfit into the training data, that did not happen in Bi-GRU. Thus we can say Bi-GRU is the best model till now.

Future Work

There is a lot of scope with future research/project work in this domain:

- We can use this model in a web extension which can be implemented on websites like YouTube and Twitter to detect and self-delete toxic comments.
- We can use Data Augmentation to make the data less imbalanced and get better accuracy, and can check what effect that has in the F1-score of CNN and RNN and if it increases the score of Bi-GRU.

Reference

[1] Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character- level convolutional networks for text classification. In Advances in neural information processing systems.

[2] Charu C Aggarwal and ChengXiang Zhai. 2012. A survey of text classification algorithms. In Mining text data. Springer, 163–222.

[3] Hossein Hosseini, Sreeram Kannan, Baosen Zhang, and Radha Poovendran. 2017. Deceiving Google's Perspective API Built for Detecting Toxic Comments. arXiv preprint arXiv:1702.08138 (2017).

[4] "Kaggle Toxic Classification Challenge." [Online]. Available: www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge

[5] CrowdFlower, CrowdFlower Data Science Report, 8–9, 2016.

[6] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning Word Vectors for Sentiment Analysis," Proceedings of the 49th Annual Meeting of the Association for Computational Linguis- tics: Human Language Technologies, 142–150 (2011).

[7] Fabio Del Vigna, Andrea Cimino, Felice Dell'Orletta, Marinella Petrocchi, and Maurizio Tesconi, Hate me, hate me not: Hate speech detection on facebook (2017).

[8] Sanjana Sharma, Saksham Agrawal, and Manish Shrivastava, Degree based classification of harmful speech using twitter data (2018).

[9] Steven Zimmerman, Udo Kruschwitz, and Chris Fox, Improving hate speech detection with deep learning ensembles (2018).

[10] Mohammad, F., Is preprocessing of text really worth your time for online comment classification? (2018)

[11] Conversational AI Research, https://www.github.com/conversationai/unintended-ml-bias-analysis (2018).