Name: Gaurav Taneja
ID: 1001955801

Report For Toxic Comment Classification

Introduction:

There is freedom of expression in today's world. People today want to share their ideas and opinions with their peers and are not afraid to do so in public. As a result, many platforms are available on the internet where anyone can share their opinions and ideas anonymously or claim credit for them. This platform has provided a plethora of opportunities and assistance to those in need. These platforms have many benefits, but there is one drawback: people share irrelevant or inappropriate information. These comments can undermine people's self-esteem, leading to a variety of mental health issues. Many large corporations, as well as the government, have taken steps to identify and neutralize these comments or the source of these comments. The government refers to the comments as cyberbullying and has taken harsh action against them. The large corporation has taken the step of asking the user to confirm or restrict the viewer of content at the time of posting. All of this is highly dependent on the classification of the comments.

Motivation:

- Today, most businesses are going online. As businesses move online, a massive quantity of data is created every minute. Companies can utilize this information as feedback to enhance their services.
- Machine learning is a gift that allows these firms to examine, analyze, and apply prediction to make business decisions.
- As the data is in multiple forms, it will be difficult to interpret or forecast directly. To obtain consistent results, we must divide the dataset into test and train datasets. The train data set is then fed into the deep learning model, which is a subset of machine learning.
- There are numerous unpleasant things on the internet today that impair the mental health of many famous people and offend viewers who see negative remarks on social media. Many large corporations have taken an effort to combat this type of conduct by blocking or removing comments.
- This project was chosen since it was presented as a challenge on the Kaggle website when I was browsing. In this research, I would investigate deep learning models that analyze text.
- I'll be utilizing the tokenizer from TensorFlow Keras to parse text. For text data, the tokenized model provides a lexical scanner. As a result, tokenization is utilized in natural language processing. Tokenization is well known for its use in cybersecurity and the creation of NFTs, tokenization is also an important part of the NLP process. Tokenization is used in natural language processing to split paragraphs and sentences into smaller units that can be more easily assigned meaning.
- A tokenizer oversees preparing model inputs. Tokenizers for all models are included in the library. Most tokenizers come in two flavors: full Python implementations and "Fast" implementations based on the Rust library Tokenizers. "Fast" implementations enable:
- a significant speedup, especially when performing batch tokenization and
- more mapping methods between the original string (characters and words) and the token space (e.g., getting the index of the token comprising a given character or the span of characters corresponding to a given token).

Research Questions:

1. How the data with all inconsistencies are used?
2. How is the preposition which has no meaning separated from words that have meaning?
3. How are words turned into language that is understandable by machines?
4. What should be done to visualize the dataset?
5. How to decide the classification classes?
6. How new data enter are classified using the training dataset that is with us?

Clean the data:

1. **How the data with all inconsistencies are used?**

The solution to this problem is to remove the row when the data's class column is null. It can be identified in a variety of ways. One method is to manually look through the file in Excel, identify the rows, and remove them. Another option is to use a program. Is to open the file and convert it to a data frame. After converting the file to a data frame, we can drop the rows using pandas' built-in function 'dropna.' If there is no NA in the file, replace the blank data with NA and then proceed as described above.

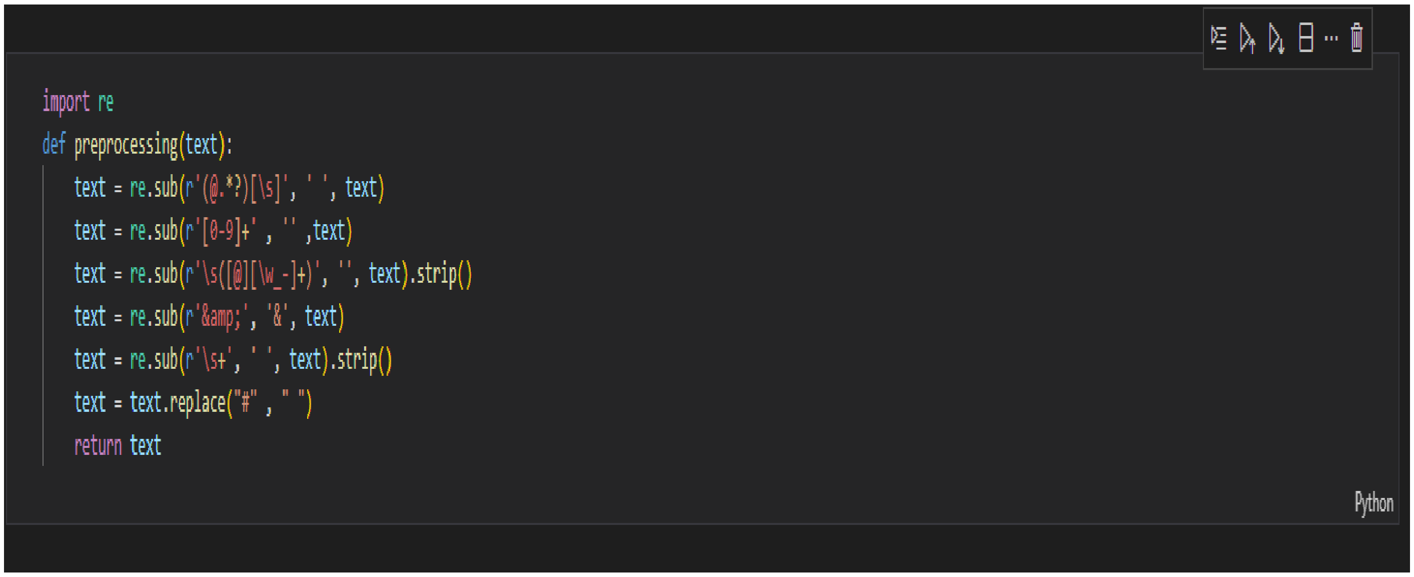
In some cases, we may need to consider two-column combinations for the class column. In this case, we cannot remove the entire row because other data may affect the results. In this case, it is preferable to take the mean of the column values and fill in the data that is missing from the column.

The cleaning with text can be done by removing garbage values from the string. The garbage values can be special characters or any unrecognizable symbols.

In this project, we are using two models to compare the results of the model:

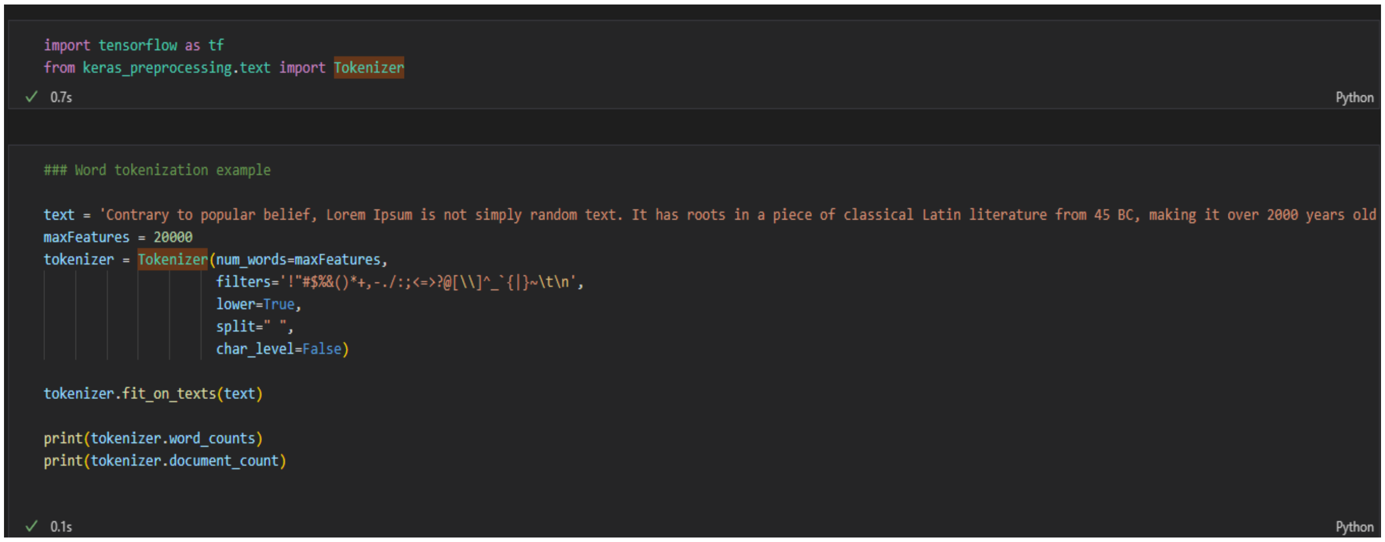
We have used two types of cleaning the data.

For the model, we have added a function that will remove the unnecessary character from the comment text:



```python
import re
def preprocessing(text):
    text = re.sub(r'(@.*?)[\s]', ' ', text)
    text = re.sub(r'[0-9]+' , '' ,text)
    text = re.sub(r'\s([@][\w_-]+)', '', text).strip()
    text = re.sub(r'&amp;', '&', text)
    text = re.sub(r'\s+', ' ', text).strip()
    text = text.replace("#" , " ")
    return text
```

In the Keras model, we are using an inbuilt library Tokenize to clear the data:



```
import tensorflow as tf
from keras_preprocessing.text import Tokenizer
```

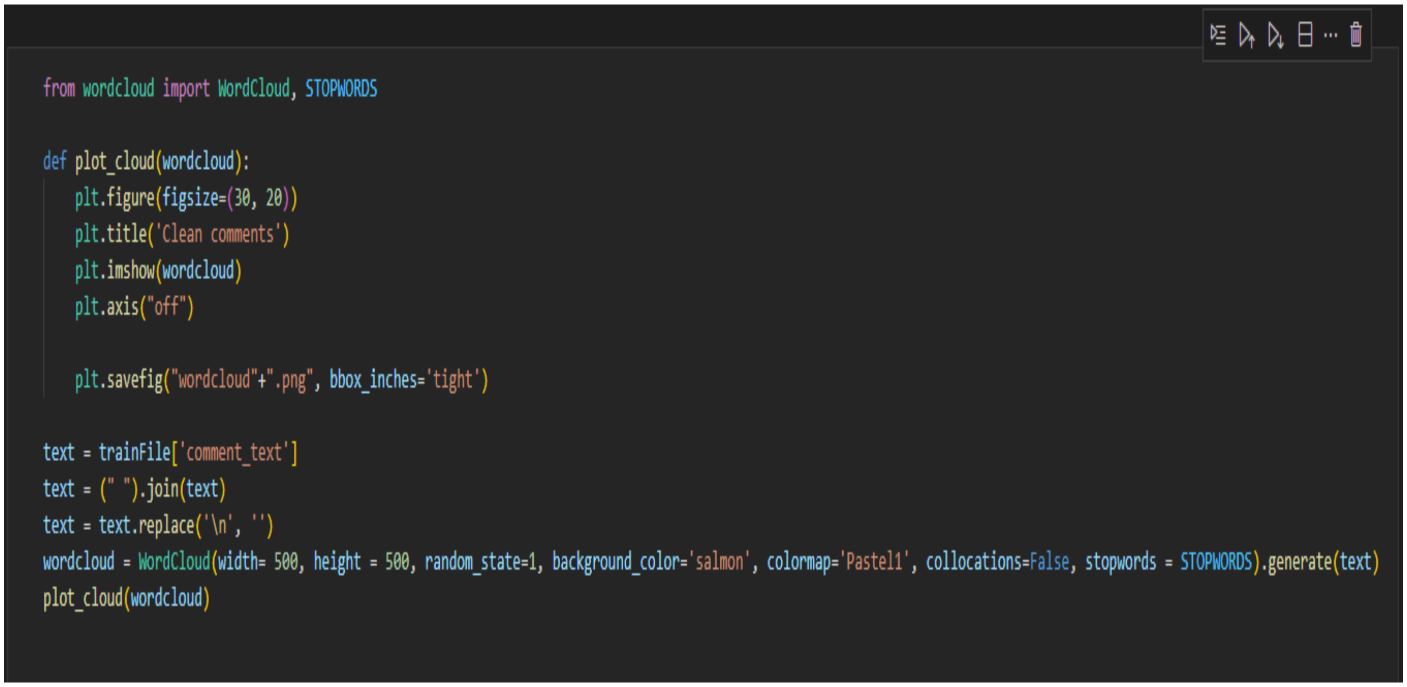
✓ 0.7s Python

```
### Word tokenization example

text = 'Contrary to popular belief, Lorem Ipsum is not simply random text. It has roots in a piece of classical Latin literature from 45 BC, making it over 2000 years old
maxFeatures = 20000
tokenizer = Tokenizer(num_words=maxFeatures,
                      filters='!"#$%&()*+,-./:;<=>?@[\\]^_`{|}~\t\n',
                      lower=True,
                      split=" ",
                      char_level=False)

tokenizer.fit_on_texts(text)

print(tokenizer.word_counts)
print(tokenizer.document_count)
```

✓ 0.1s Python

2. **How is the preposition which has no meaning separated from words that have meaning?**

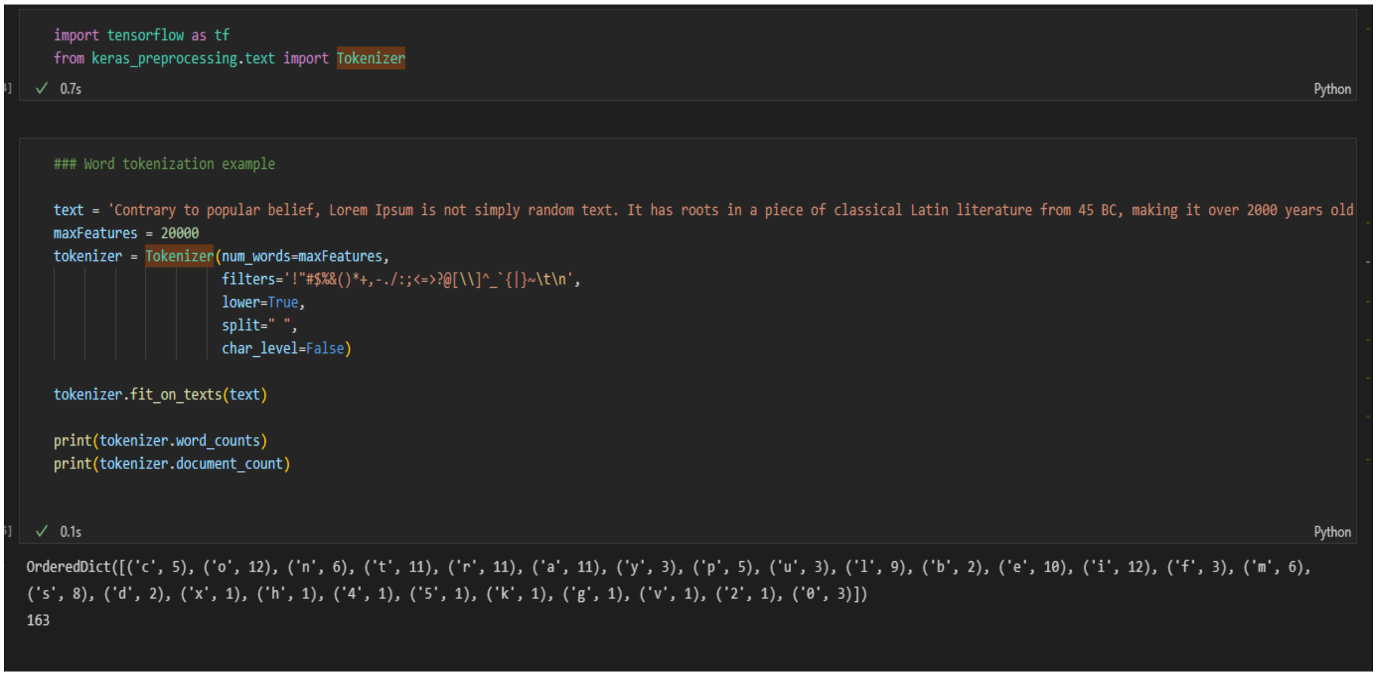
If we are working with Natural Language Processing the preposition has no meaning in the sentence. Preposition basically depends on other words to make meaning. So, training the model with the preposition will confuse the model and may lead to incorrect results. For both models, we are using the word cloud. The world cloud use term Stopwords which refers to preposition which is ignored while visualizing the words of the file.



```
from wordcloud import WordCloud, STOPWORDS

def plot_cloud(wordcloud):
    plt.figure(figsize=(30, 20))
    plt.title('Clean comments')
    plt.imshow(wordcloud)
    plt.axis("off")

    plt.savefig("wordcloud"+".png", bbox_inches='tight')

text = trainFile['comment_text']
text = (" ").join(text)
text = text.replace('\n', '')
wordcloud = WordCloud(width= 500, height = 500, random_state=1, background_color='salmon', colormap='Pastel1', collocations=False, stopwords = STOPWORDS).generate(text)
plot_cloud(wordcloud)
```

3. How are words turned into language that is understandable by machines?

We use tokenization so the data makes more sense for the model and make an easy decision while classification and prediction.

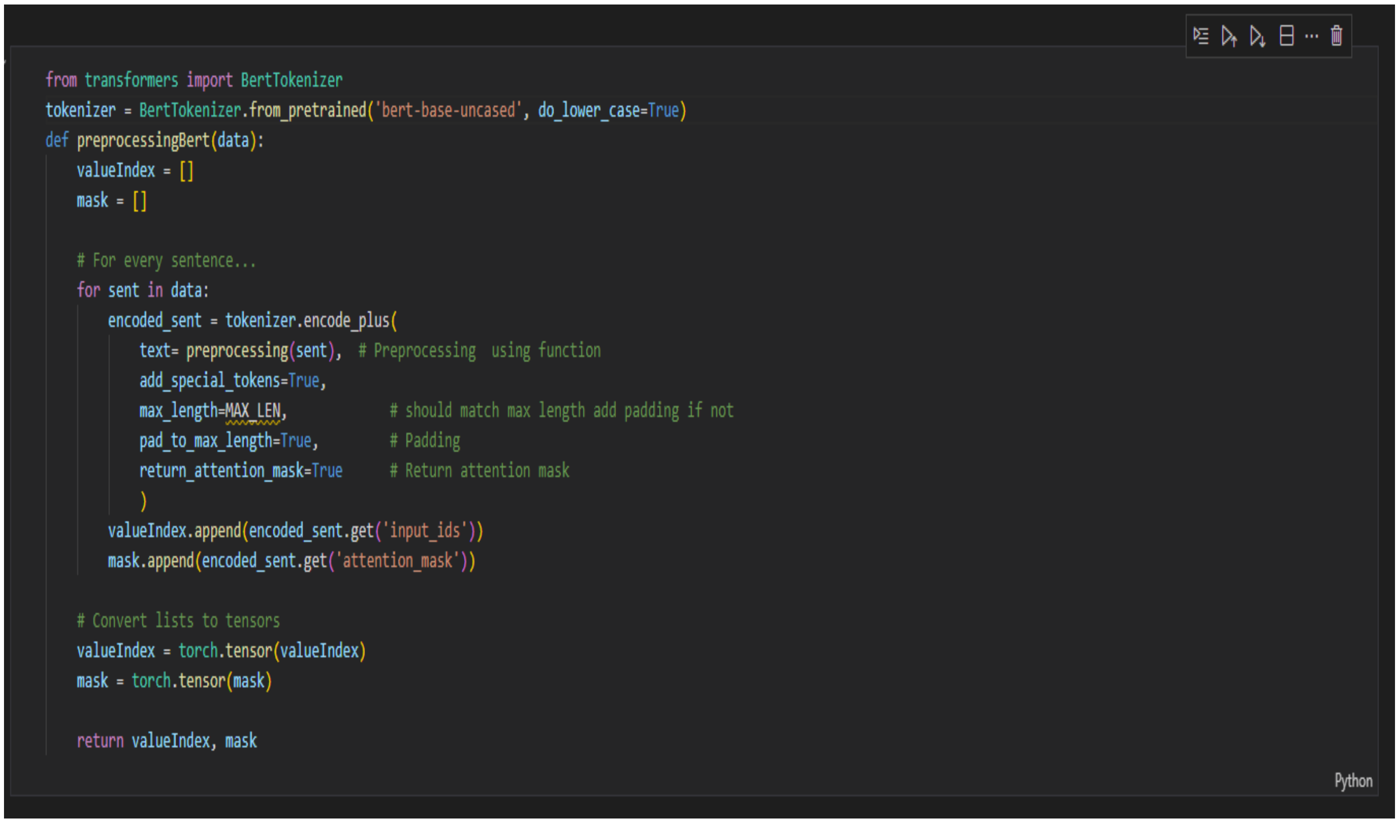


```
import tensorflow as tf
from keras_preprocessing.text import Tokenizer
```
✓ 0.7s Python

```
### Word tokenization example

text = 'Contrary to popular belief, Lorem Ipsum is not simply random text. It has roots in a piece of classical Latin literature from 45 BC, making it over 2000 years old
maxFeatures = 20000
tokenizer = Tokenizer(num_words=maxFeatures,
                      filters='!"#$%&()*+,-./:;<=>?@[\\]^_`{|}~\t\n',
                      lower=True,
                      split=" ",
                      char_level=False)

tokenizer.fit_on_texts(text)

print(tokenizer.word_counts)
print(tokenizer.document_count)
```
✓ 0.1s Python

```
OrderedDict([('c', 5), ('o', 12), ('n', 6), ('t', 11), ('r', 11), ('a', 11), ('y', 3), ('p', 5), ('u', 3), ('l', 9), ('b', 2), ('e', 10), ('i', 12), ('f', 3), ('m', 6),
('s', 8), ('d', 2), ('x', 1), ('h', 1), ('4', 1), ('5', 1), ('k', 1), ('g', 1), ('v', 1), ('2', 1), ('0', 3)])
163
```

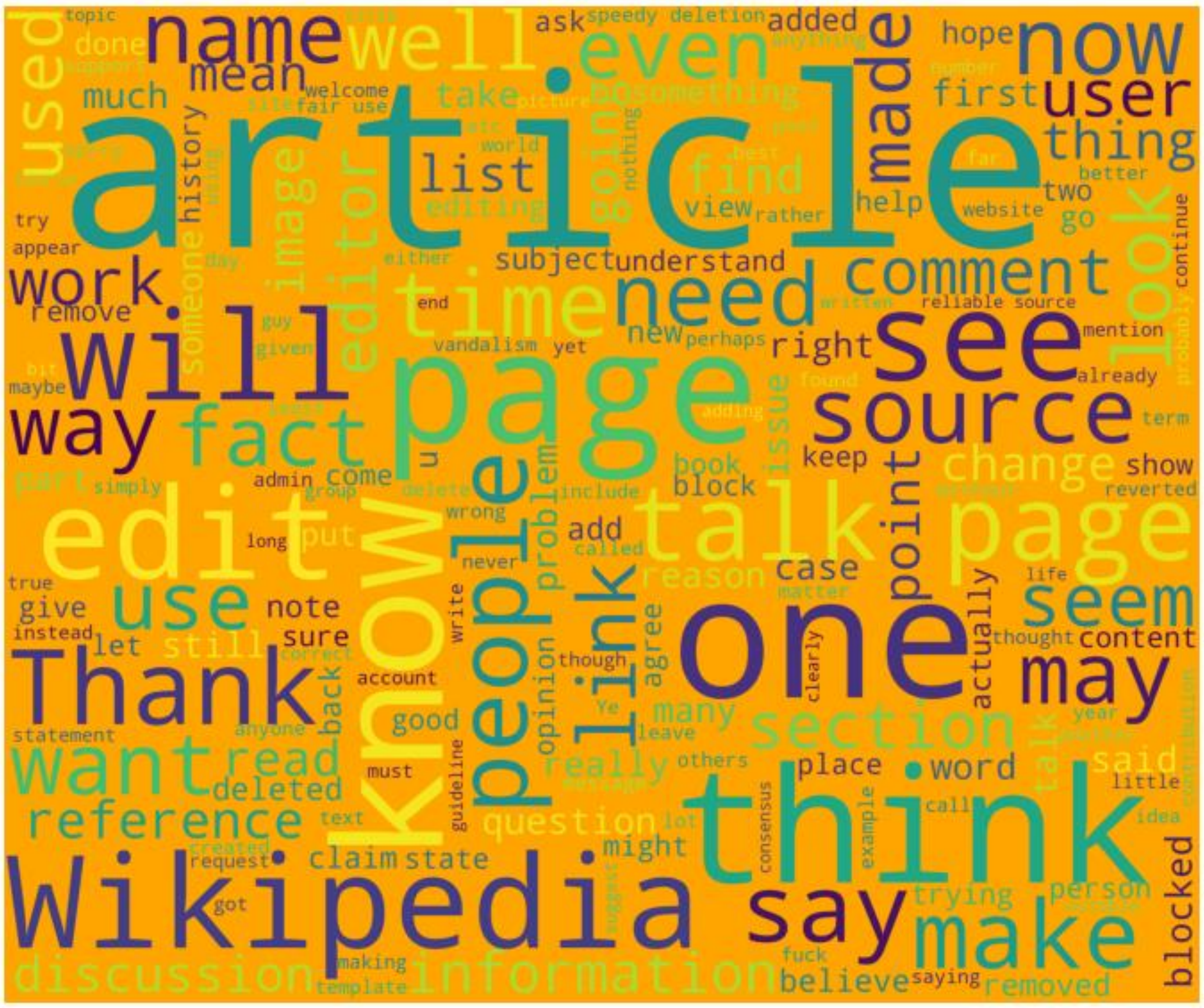
For the Bert Model:



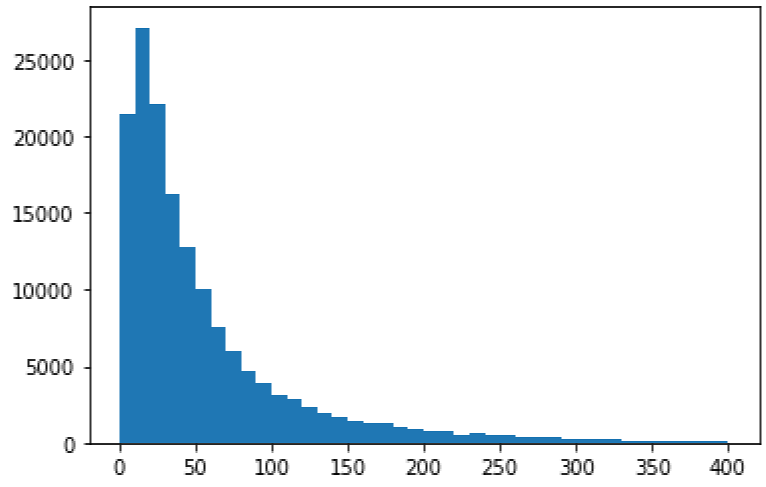
```
from transformers import BertTokenizer
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased', do_lower_case=True)
def preprocessingBert(data):
    valueIndex = []
    mask = []

    # For every sentence...
    for sent in data:
        encoded_sent = tokenizer.encode_plus(
            text= preprocessing(sent),  # Preprocessing  using function
            add_special_tokens=True,
            max_length=MAX_LEN,             # should match max length add padding if not
            pad_to_max_length=True,         # Padding
            return_attention_mask=True      # Return attention mask
            )
        valueIndex.append(encoded_sent.get('input_ids'))
        mask.append(encoded_sent.get('attention_mask'))

    # Convert lists to tensors
    valueIndex = torch.tensor(valueIndex)
    mask = torch.tensor(mask)

    return valueIndex, mask
```
Python

Analyze:

Data analysis will assist us to understand the kind of data we have and the values we have considered. Visualizing the dataset and learning about its characteristics might help with analysis. Analyzing how often certain terms appear in NLP is one way to conduct analysis. The word cloud can be used to do this. The larger word will be the one that appears the most frequently.



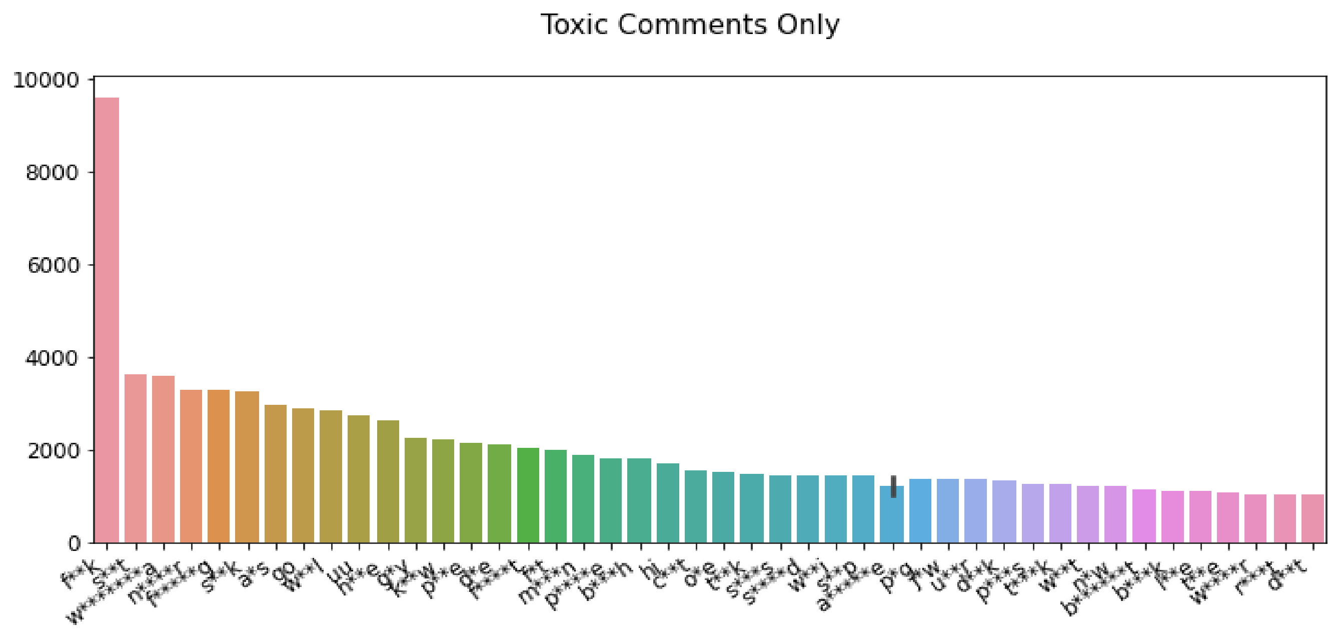
Since the dataset we are considering for the project is already split into a train, sample, and test data sets we have already classified some data that is in the train data set. So, our next analysis for the project will be the consistencies which are the length of the input string that will be given to the model.



As we see from the above graph, we visualize that the length of most comments is between 0 to 500 so the ideal length we will be considered for the model input is 200. If the data is less than 200 then the padding will be added if it is more than it is safe to truncate the part which doesn't fit.

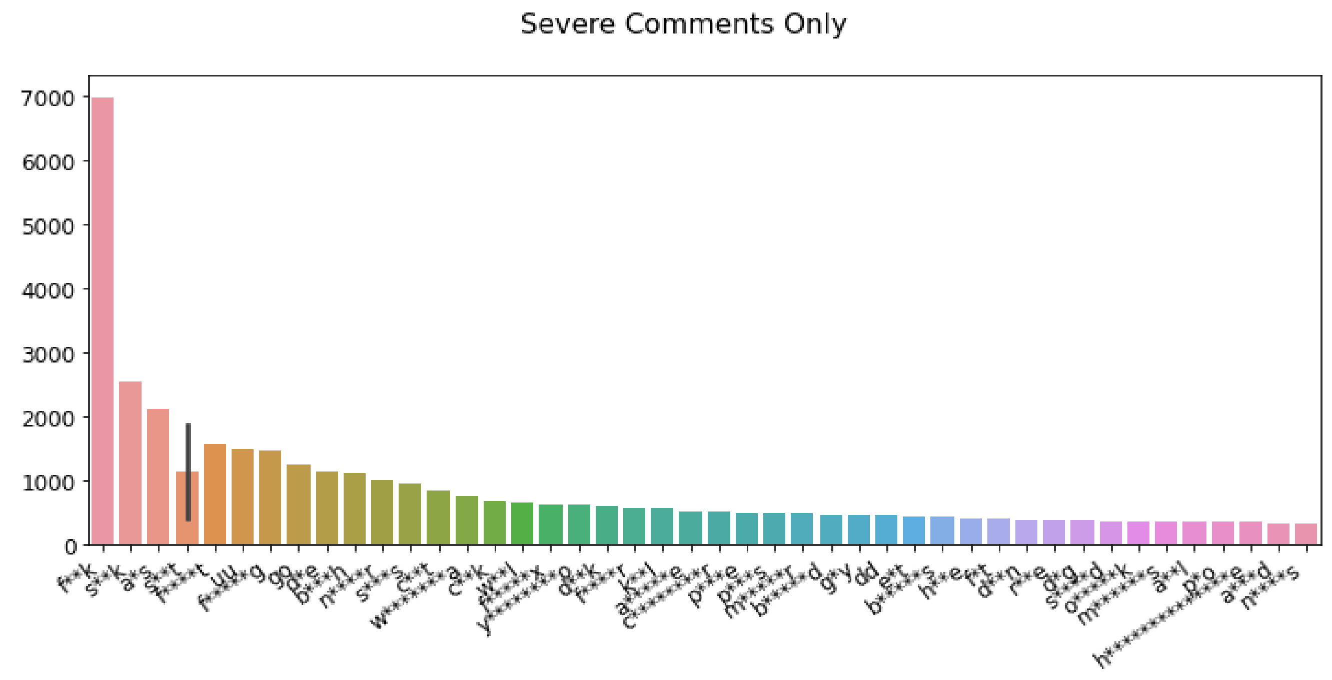
Another part to visualize is the words that will classify the comments. So below is the visualization of the words that are considered for classifying the comments. The categories we are considering are toxic, severe toxic, hate, clean, threat, insult, and obscene.

For toxic comments:

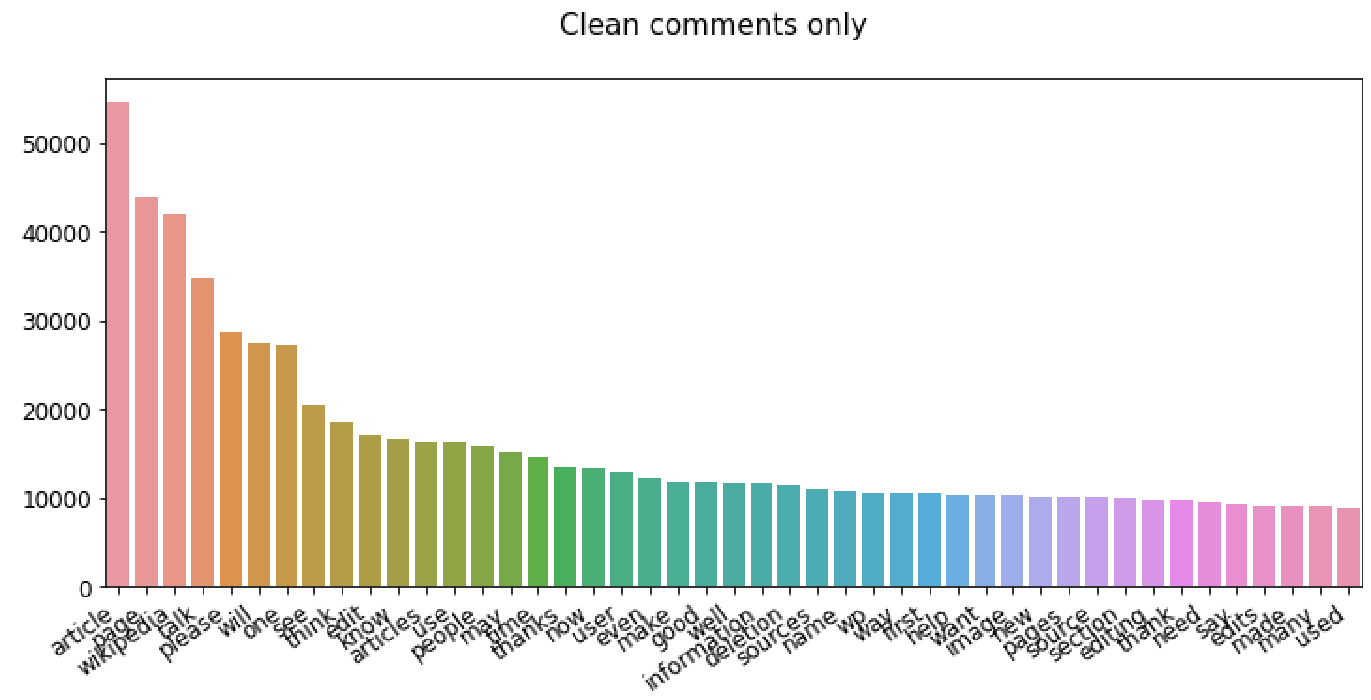


Name: Gaurav Taneja
ID: 1001955801

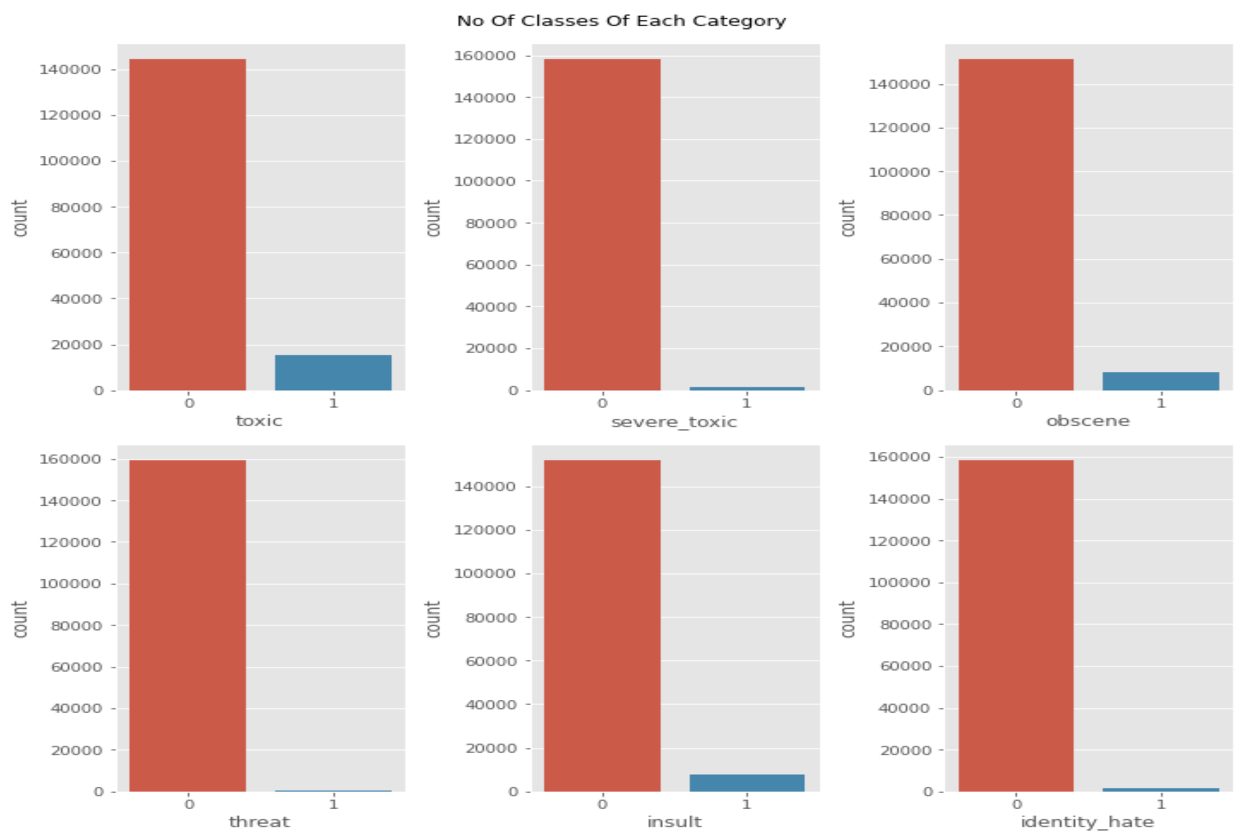
For Severe Comment:

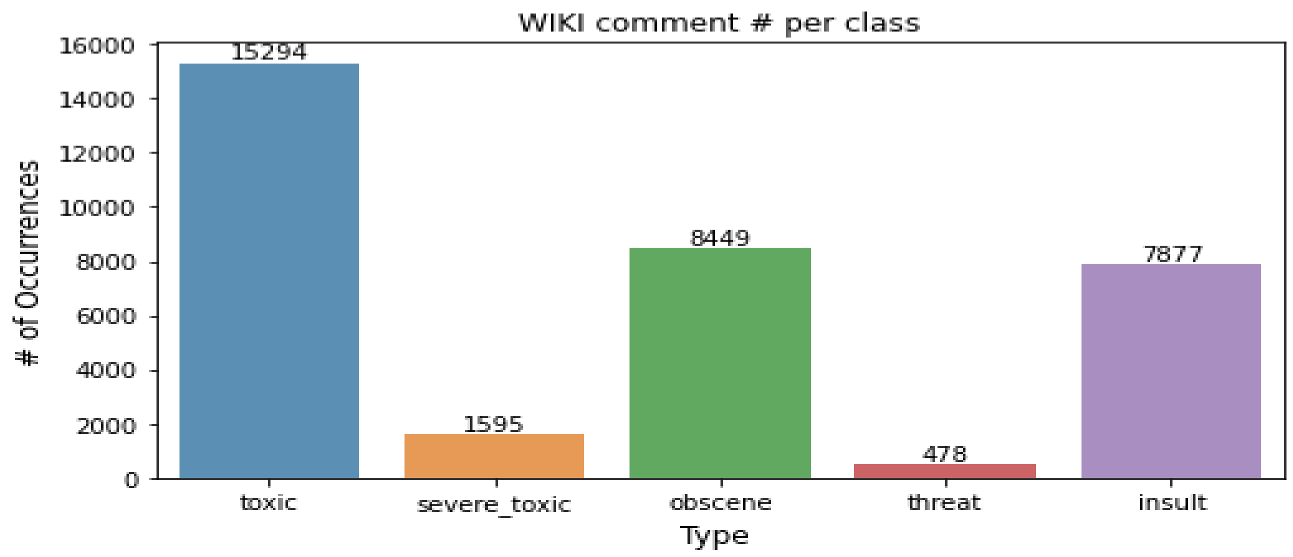


For clean comments:



Another visualization is the occurrence of a category of comments in the train dataset. Below are the two visualizations of the train dataset classifications.





The first graph shows the presence of the comments out of the entire dataset 0 means not 1 means is part of that classification. The second graph shows the total of comments present in a category in the dataset.

Interpret the results:

To interpret the result of the dataset we ideally split the data into train and test datasets. But in this case, the data set is already split. Even the train data sets are classified. So next step is to train the model. In this project, we are using two models the Keras model, and one is Bert model.

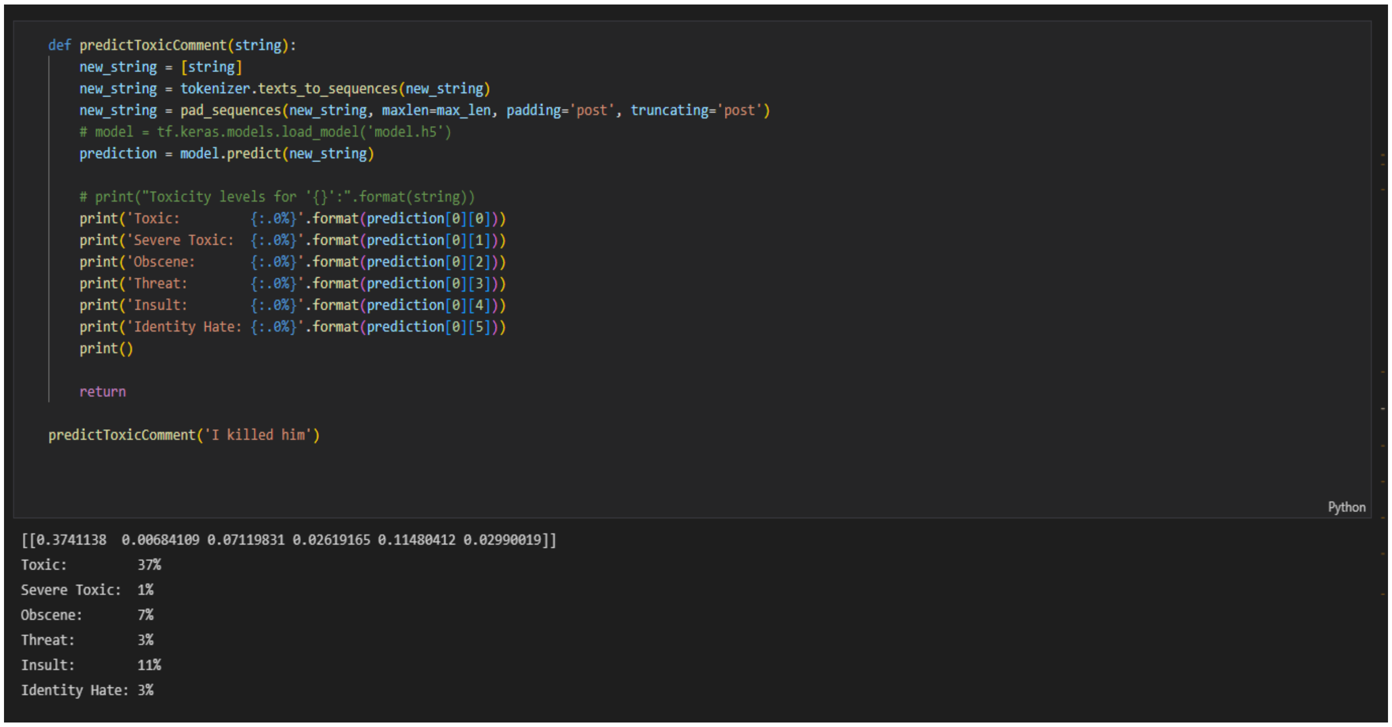
Keras Model:

Keras is a neural network API for Python that is tightly integrated with TensorFlow, which is used to build machine learning models. Keras' models provide a simple, user-friendly way to define a neural network, which TensorFlow will then build for you.

Bert Model:

BERT is a machine learning framework for natural language processing that is open source (NLP). BERT is intended to assist computers in understanding the meaning of ambiguous language in the text by establishing context using surrounding text.

We have built a function that uses Keras model and interprets the output for the input to the function.

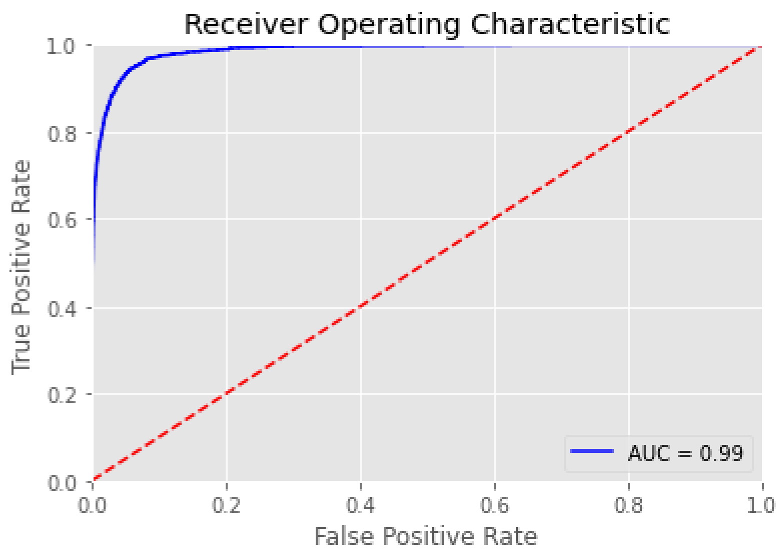


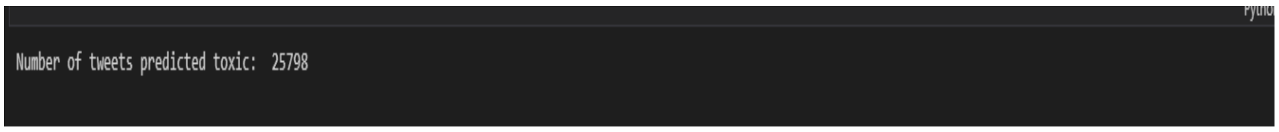
```python
def predictToxicComment(string):
    new_string = [string]
    new_string = tokenizer.texts_to_sequences(new_string)
    new_string = pad_sequences(new_string, maxlen=max_len, padding='post', truncating='post')
    # model = tf.keras.models.load_model('model.h5')
    prediction = model.predict(new_string)

    # print("Toxicity levels for '{}':".format(string))
    print('Toxic:         {:.0%}'.format(prediction[0][0]))
    print('Severe Toxic:  {:.0%}'.format(prediction[0][1]))
    print('Obscene:       {:.0%}'.format(prediction[0][2]))
    print('Threat:        {:.0%}'.format(prediction[0][3]))
    print('Insult:        {:.0%}'.format(prediction[0][4]))
    print('Identity Hate: {:.0%}'.format(prediction[0][5]))
    print()

    return

predictToxicComment('I killed him')
```

Python

```
[[0.3741138  0.00684109 0.07119831 0.02619165 0.11480412 0.02990019]]
Toxic:         37%
Severe Toxic:  1%
Obscene:       7%
Threat:        3%
Insult:        11%
Identity Hate: 3%
```

For the Bert model, we have plotted the prediction. The output is stored in the file that is generated by considering the test data prediction stored in a new CSV file. Using the trapezoidal rule, compute the Area Under the Curve (AUC). Given points on a curve, this is a general function. See the roc AUC score for calculating the area under the ROC curve. ROC curves are commonly used in binary classification to investigate a classifier's output. To apply the ROC curve and ROC area to multi-label classification, the output must be binarized.





```
Number of tweets predicted toxic:  25798
```

Conclusion:

Managed to demonstrate that deep learning models can accurately predict the target from the comment's toxicity level. We also discovered that large models were based on a massive amount of publicly available data. We discovered some syntactic features that may aid in better predicting the target of toxic content, but we have yet to apply them to improve the performance of our classifiers. We touched on several possibilities provided by current NLP techniques throughout this project, but several additional experiments remain possible in the future. We'd like to investigate the impact of emojis on the toxicity level of comments, as well as their potential to improve their, potential detection by assessing and leveraging their toxicity level.

References:

1. Worked on the project in the previous semester. Worked with the Bert model and learned various models and chose to work with the basic Bert Model.
2. https://www.tensorflow.org/guide/keras/preprocessing_layers
3. https://huggingface.co/docs/transformers/model_doc/bert
4. https://www.kaggle.com/competitions/jigsaw-toxic-comment-classification-challenge/data