TOXIC COMMENT CLASSIFICATION

SAMSUNG INNOVATION CAMPUS PROGRAM

ARTIFICIAL INTELLIGENCE

By

Nehanshu Jacob Singh

Roll No: (190013125033)

Rashmi Mishra

Roll No: (190013125043)

Prakhar Namdev

Roll No: (190013125036)

Chiranjeev Veer Srivastava

Roll No: (190013125019)

Under the guidance of

Dr. Sachin Kumar

Abstract

It is crucial to act politely when participating in online forums. "Wilful and repetitive harm perpetrated through the medium of electronic text" is how cyberbullying is defined. It entails communicating with targets using websites, blogs, instant messaging, chat rooms, e-mail, cell phones, websites, and personal online profiles while also using derogatory, threatening, and/or sexually explicit remarks and photos. As a result, it is essential but impossible for human moderators to identify and remove toxic communication from public forums.

The categorizing of various toxic remark types into one or more categories, such as "toxic," "severe toxic," "obscene," "threat," "insult," and "identity hate," is known as toxic comment categorization. The topic is particularly fascinating due to the heated discussions around how the general health of society has been damaged by the toxic information on the internet.

The classification of unstructured text into dangerous and benign categories using unique applications of Natural Language Processing techniques is presented in this work. Social media has increased career prospects in the twenty-first century while also developing into a distinctive space for people to interact, allowing individuals to openly voice their opinions. In the meanwhile, these users include various organizations and beings utilizing this architecture, and exploit this freedom to advance their negative viewpoint (i.e. insulting, verbal sexual harassment, threads, Obscene, etc.). The Youth Risk in 2017According to the Behaviour Surveillance System (Centres for Disease Control and Prevention), 14.9% of high school pupils used electronic devices bullied over the previous year, before the survey. The initial outcome could be an Open Source approach utilized by app developers in support of antibullying efforts. The assessment of the findings demonstrated that LSTM has a true positive rate that is 20% greater than that of the popular Naive Bayes approach. These results suggested that wise use of data science is able to form a more wholesome setting for virtual communities.

Table of Contents

1. Introduction

The internet forums and discussion boards have now enabled people and have given them the opportunity to completely voice their viewpoints on a variety of issues and events. These internet comments occasionally use explicit language that can offend readers. Several categories, including Toxic, Severe Toxic, Threat, Insult, and Identity Hate, can be used to categorise comments that include abhorrent, ludicrous, highly twisted, hazardous, repulsive, or filthy language. Due to the fear of harassment and abuse, many people refrain from speaking their minds and give up on exploring other points of view.

Companies such as Twitter, Instagram etc. have started flagging remarks and omitting those who are charged or convicted of using profanity in order to shield people from witnessing derogatory terms on internet forums or social media platforms. To filter out the foul language and shield internet users from experiencing online harassment and cyberbullying, numerous ML programs have been proposed and applied.

1.1 About the Organization

Name of the Organization: Samsung Innovation Campus

Course: Artificial Intelligence

About: Samsung Innovation Campus provides ICT education to youth who want to get a jobs in the ever evolving world of technology. Young people who want to develop their technical talents can learn about ICT technology and improve their skills through the program.

Samsung Innovation Campus also attempts to provide more underprivileged youth with access to education. They have also planned to build an online education platform to help more young people with diverse educational opportunities and diverse forms of training, helping them leverage their unique talents and abilities with education. Samsung Innovation Campus defines core technologies leading the 4th Industrial Revolution as' AI, IoT, Big Data, Cloud, Mobile Platform, and Coding & Programming'. In addition to these they also offers 'soft skill', which allows students to strengthen their employability capabilities.

2. Definition

2.1 Literary Review on Related Work

There is a significant number of research papers on the toxic comment classification problem, but, to date, there has not been a systematic literature review of this research theme, making it difficult to assess the maturity, trends and research gaps. Most commonly used approach is by using Automatic or semi-automatic detection of toxic comment is done by using different machine learning methods, mostly different deep neural networks architectures.

To detect toxicity in comments, Chu et al. (2017b) explored separately long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) based recurrent neural network (RNN) and convolutional neural network (CNN) architectures. They observed that CNN performed better and prove more computationally efficient when paired with character than with word embeddings. Yenala et al. (2017) propose an architecture that synthesizes CNN and bidirectional LSTM to detect inadequate queries in Web search. Their model shown to significantly outperform pattern based and laborious hand-coded features. More recently, a CNN model fed by word vectors to classify hate-speech on Twitter (Gamback and Sikdar, 2017), achieved an F1 score of 78.3% to improve performance over an LR baseline. Rather than toxicity, identifying constructiveness in news comments is studied in the work by Kolhatkar and Taboada (2017) that uses an LSTM based classifier with a highest test accuracy of 72.6%

2.2 Problem Statement

To create a multi-headed model that can recognise various forms of toxicity, such as threats, obscenities, insults, and hate speech motivated by an individual's identity. The objective is to develop a classifier model that can foretell whether incoming text is improper (toxic).

- Scrutinize the dataset to better comprehend the labels' distribution, correlations, and definitions of toxic and clean comments.

- Examine the performance of various machine learning tactics and opt the most effective one for this issue.
- Create the final model with the top-performing parameters and algorithm, then evaluate it using a holdout subset of the data.

2.3 Technology Used

Platform: Jupyter Notebook

Packages used:

- Pandas
- Numpy
- Matplotlib
- Seaborn
- Nltk
- Wordcloud
- Xgboost
- Light Gradient Boosting
- Random forest classifier
- Logistic regression

```
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.metrics import
accuracy_score,classification_report,confusion_matrix,f1_score
from sklearn.metrics import roc_curve, roc_auc_score
from sklearn.model_selection import cross_val_score,GridSearchCV
from lightgbm import LGBMClassifier
from sklearn.linear_model import LogisticRegression
from xgboost import XGBClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import KFold
```

2.4 Data Set

The dataset we used to assess the proposed methodology is the same one employed by Kaggle's Toxic Comment Classification Challenge. The well-known website Kaggle offers a variety of predictive modelling and analytics competitions where users must suggest the top prediction model for a specific dataset and task. **The Conversation AI team5**, a group created by Jigsaw and Google, has provided the relevant dataset for the Toxic Comment Classification Challenge. The comments from Wikipedia's talk page that have been flagged as having toxic behaviour are included in such a dataset. The information used is made up of a sizable collection of Wikipedia comments that have been divided into groups based on how harmful they are to people. The training set, **train.csv**, includes comments with their binary labels. We estimate the toxicity probabilities for these remarks in the test set, **test.csv**.

3. Analysis

3.1 Pre-processing of Text

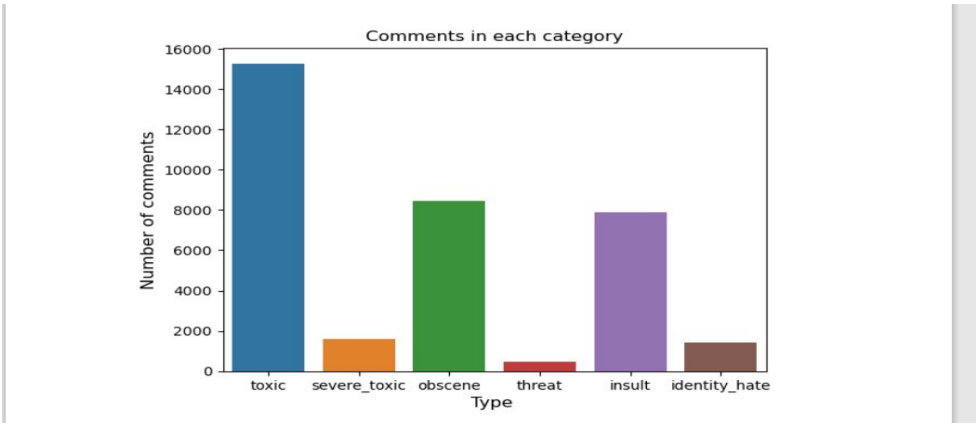
The effectiveness depends on the data's pre-processing because it lessens ambiguity in feature extraction. Data pre-processing is therefore a crucial stage in text classification. All text is converted to lowercase as part of the pre-processing, and stop words, URLs, usernames, and other special characters are also eliminated.

3.2 Feature Extraction

While it is simple for humans to categorise images or words, it is more challenging for computers, which only work with numbers and, to be more precise, analyse those numbers as electrical impulses. Any data must be transformed into that format in order for the computer to process it and return the outcome. Therefore, feature extraction is crucial to text processing. Therefore, we must vectorize the input data before training the model. By employing n-grams and the TF-IDF which is said to be the **"Term Frequency Inverse Document Frequency",** features can be extracted. Unigrams and bigrams are two ways to measure the importance of a given characteristic in a text. Features are therefore filtered using the maximum weight.

3.3 Data Visualization

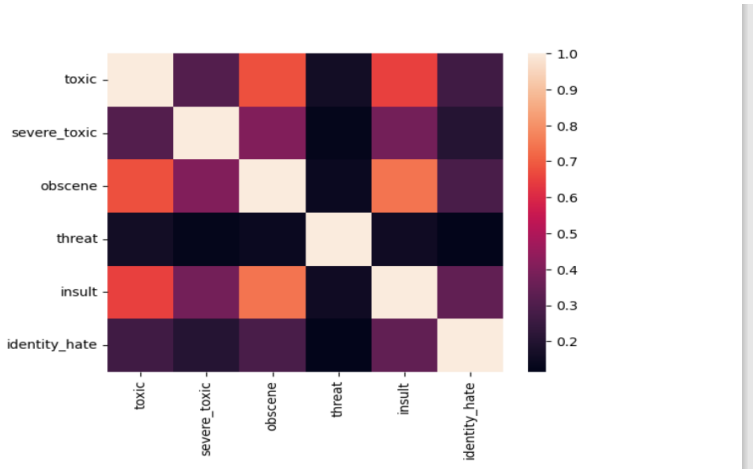
There are 159571 comments in this dataset. One input feature—the string data for the comments—and six labels—toxic, severe toxic, obscene, threat, insult, and identity hate—represent the different categories of toxic comments—make up the data. The distribution of the labels across the dataset, is broken down in the figure given below. More than 14000 of the harmful remarks are toxic, and the least comments are in the category of threat around less than 2000.



More information about these converging categories is given by the correlation matrix below. Threats are unlikely to be racially or homophobically motivated, nor are they likely to be extremely destructive. However, insults are frequently vulgar, and identity hatred seldom ever overlaps at all.

As they will have many contributing characteristics, the categories with a lot of overlap will be more challenging to predict, whereas "identity hate" will have more distinctive characteristics and be simpler to do so.

Correlation Matrix:



3.4 Tokenization

Tokenization involves cutting the raw text into manageable pieces. Tokenization divides the original text into tokens, which are words and sentences. These tokens aid in context comprehension or model development for NLP. By examining the word order in the text, tokenization aids in comprehending the text's meaning.

The words "It is raining," for instance, can be tokenized into "It," "is," and "raining."

Tokenization can be done using a variety of tools and frameworks. Some of the libraries that can be utilised to do the work include NLTK, Genism, and Keras. You can tokenize individual words or entire sentences. The process of breaking up text into words using a technology is known as word tokenization, while the process of doing the same for sentences is known as sentence tokenization. Stop words are those in the text that don't provide any

context to the sentence and removing them won't change how the content is processed for the intended purpose.

```
from tensorflow.keras.preprocessing.sequence import pad_sequences
X_train = train_data["comment_text"]
y_train = train_data.iloc[:, 2:]
y_train[y_train['toxic'] == 1]
tokenizer = keras.preprocessing.text.Tokenizer()
tokenizer.fit_on_texts(X_train)
X_train = tokenizer.texts_to_sequences(X_train)
X_train = pad_sequences(X_train, maxlen=100)
```

3.4 Algorithms and Techniques

Using high dimensional data for classification is a challenge in natural language processing. I'll vectorize the data and evaluate various categorization methods. The term frequency-inverse document frequency (tf-idf) statistic technique used to vectorize the text data considers both the frequency of words or character n-grams in the text and the relevancy of those tokens over the entire dataset. The weight of more uncommon tokens is increased while the weight of common tokens is decreased by the inverse document frequency.

Additionally, a variety of engineered features including various facets of the comment content, such as the typical word length, capitalization, and exclamation point usage, will be created. In order to improve the solution, I will perform the benchmark test without these features and try them out. I will test many methods with default parameters using the benchmark features and vectorization to find the best method of solving the issue.

Although deep learning algorithms may be too resource-intensive for an algorithm that must run instantly every time a remark is submitted on one of the most popular websites on the Internet, recurrent neural networks perform well on this problem and dominate Kaggle leader boards. Efficiency would be a key necessity if this were a real-world commercial issue. Additionally, stacking—a cumbersome fix that makes training and prediction more difficult—would be needed to modify the model to account for secondary features. Model stacking reduces efficiency, and I've used it in Kaggle tournaments before.

Transparency is yet another factor to take into account, and it's by far the primary cause why neural networks were rejected for this application. To make sure the model is not picking up bias related to race, gender, sexual orientation, culture, other unanticipated categories from the data curators, I want to be able to readily audit the model. It will be much simpler for a third party to examine the effects of particular characteristics on the model and make necessary corrections to counter bias using SVM, Naive Bayes, and LightGBM Classifier, RandomForestClassifier, XGBoost , Logistic Regression .

1. RandomForestClassifier-:

- The Random forest or Random Decision Forest is a supervised Machine learning algorithm used for classification, regression, and other tasks using decision trees.
- The Random forest classifier creates a set of decision trees from a randomly selected subset of the training set. It is basically a set of decision trees (DT) from a randomly selected subset of the training set and then it collects the votes from different decision trees to decide the final prediction.
- In this classification algorithm, we will use different datasets to train and test the model

2. LGBM (Light Gradient Boosting Machine)-:

- **LightGBM** is a gradient boosting framework based on decision trees to increases the efficiency of the model and reduces memory usage.
- It uses two novel techniques: **Gradient-based One Side Sampling** and **Exclusive Feature Bundling (EFB)** which fulfils the limitations of histogram-based algorithm that is primarily used in all GBDT (Gradient Boosting Decision Tree) frameworks. The two techniques of GOSS and EFB described below form the characteristics of LightGBM Algorithm. They comprise together to make the model work efficiently and provide it a cutting edge over other GBDT frameworks.
- **Gradient-based One Side Sampling Technique for LightGBM:** Different data instances have varied roles in the computation of information gain. The instances with larger gradients (i.e., under-trained instances) will contribute more to the information gain. GOSS keeps those instances with large gradients (e.g., larger than a predefined threshold, or among the top percentiles), and only randomly drop those instances with small gradients to retain the accuracy of information gain estimation. This treatment can lead to a more accurate gain estimation than uniformly random sampling, with the same target sampling rate, especially when the value of information gain has a large range.

3. Logistic Regression-:

Logistic regression estimates the probability of an event occurring, such as voted or didn't vote, based on a given dataset of independent variables.

Types of logistic regression-:

Binary logistic regression: In this approach, the response or dependent variable is dichotomous in nature—i.e. it has only two possible outcomes (e.g. 0 or 1). Some popular examples of its use include predicting if an e-mail is spam or not spam or if a tumour is malignant or not malignant. Within logistic regression, this is the most commonly used approach, and more generally, it is one of the most common classifiers for binary classification.

Multinomial logistic regression: In this type of logistic regression model, the dependent variable has three or more possible outcomes; however, these values have no

specified order. For example, movie studios want to predict what genre of film a moviegoer is likely to see to market films more effectively. A multinomial logistic regression model can help the studio to determine the strength of influence a person's age, gender, and dating status may have on the type of film that they prefer. The studio can then orient an advertising campaign of a specific movie toward a group of people likely to go see it.

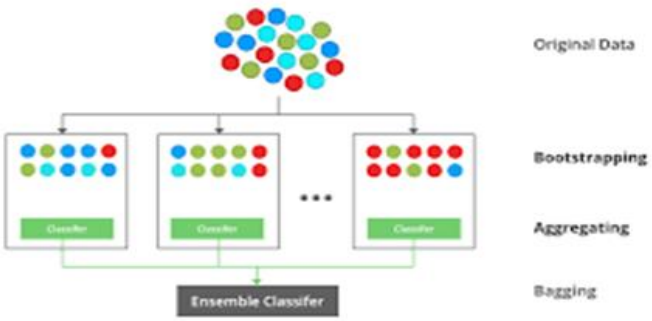
Ordinal logistic regression: This type of logistic regression model is leveraged when the response variable has three or more possible outcome, but in this case, these values do have a defined order.

4. XGBoost-:

- **XGBoost** is an optimized distributed gradient boosting library designed for efficient and scalable training of machine learning models. It is an ensemble learning method that combines the predictions of multiple weak models to produce a stronger prediction.

- XGBoost stands for “Extreme Gradient Boosting” and it has become one of the most popular and widely used machine learning algorithms due to its ability to handle large datasets and its ability to achieve state-of-the-art performance in many machine learning tasks such as classification and regression.

- One of the key features of XGBoost is its efficient handling of missing values, which allows it to handle real-world data with missing values without requiring significant pre-processing. Additionally, XGBoost has built-in support for parallel processing, making it possible to train models on large datasets in a reasonable amount of time.



XGBoost Working

Visualising Gradient boost

4. Steps Involved

4.1 Examine any missing values:

First and foremost, I choose to check for missing values in the downloaded data after importing the training and test data into the pandas dataframe. I found that there were no missing records using the "isnull" function on both the training and test data, therefore I continued with my project at this point.

4.2 Normalization of Text:

I made the decision to begin with data pre-processing because I was now assured that there are no missing records in my data. First, I made the decision to normalise the text data because comments from internet forums frequently utilise erratic language, special characters in place of letters (like @rgument), and digits in place of letters (e.g. n0t). The following is a list of the text normalisation steps carried out: -

- Eliminating spaces between text characters.
- Eliminating Characters That Recur.
- lowering the case of data.
- Elimination of Punctuation
- removing any unused spaces between words.
- deleting "\n".
- Remove characters that aren't English.

To carry out the actions stated above: A dictionary with typical definitions for curse words that are regularly used on social media sites or online forums. Second, made the decision to develop a function that completes all of the aforementioned actions and outputs clean data. Both the training data and the test data were used for these steps.

4.3 Lemmatization:

Now that the data is organised and consistent, lemmatization can be done. Lemmatization is the process of combining a word's several inflected forms into a single unit for analysis. For instance, we don't want studying, studies, and study to be treated as three different words by the machine learning algorithm because they aren't. The verbs "study" and "studies" are reduced to their root form through lemmatization. Lemmatization was put into practise by importing "WordNetLemmatizer" from the "nltk" library, creating a function called "lemma," and using it on the clean data obtained in Step 2 to execute lemmatization.

```
from nltk.stem import WordNetLemmatizer
import nltk
from nltk.corpus import  stopwords
import string

nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('omw-1.4')

[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\HP\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\HP\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package omw-1.4 to
[nltk_data]     C:\Users\HP\AppData\Roaming\nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!

True
```

4.4 Removal of Stop Words:

We all know that one of the most important phases in text pre-processing for use-cases involving text classification is stopword removal. Eliminating stopwords ensures that the words that define the text's meaning are given more attention. I used the "spacy" package to get rid of stopwords from my data. The "STOP WORDS" list provided by Spacy can be used to eliminate stopwords from any textual data.

4.5 Indexing, Tokenization, and Index Representation

We are all aware that, regardless of the use case, machine learning and deep learning models operate on numerical data. Therefore, the data must be transformed into its equivalent machine-readable form in order to train a deep-learning model utilising the clean text data. The actions listed below must be taken in order to accomplish this feat:

Tokenization: We must separate the statement into separate terms. For instance, "I love cats and dogs" will be transformed into "I", "love", "cats", "and," "dogs."

Indexing: We arrange the words in a manner like to a dictionary and assign each one a unique index, such as "I," "love," "cats," "and," and "dogs," in that order.

We could use an index to represent the terms in the comments in order, and then feed this chain of indexes to our deep-learning model. For e.g. [1,2,3,4,2,5].

The aforementioned actions can be quickly carried out by utilising the "Tokenizer" class from the "Keras" library. This class enables the vectorization of text corpora by converting each text into a vector with a binary coefficient for each token, based on word count, tf-idf, etc., or into

a series of integers (each integer representing the index of a token in a dictionary). The example code below shows how to turn text data into sequence vectors.

```
max_features=100000
tokenizer = Tokenizer(num_words=max_features)
tokenizer.fit_on_texts(list(processed_train_data))
list_tokenized_train = tokenizer.texts_to_sequences(processed_train_data)
list_tokenized_test = tokenizer.texts_to_sequences(processed_test_data)
```

4.6 Padding:

Online forums and social media sites often have comments that range in length from one word responses to lengthy arguments. Sentences of varying length are translated into sequence vectors of varying length, and we are unable to feed inconsistently sized vectors to our deep learning model. We employ Padding to get around this problem. With the use of padding, we may lengthen the shorter sentences by adding the missing zeros, while also shortening the longer sentences to match the length of the shorter ones. I applied post padding (i.e., for shorter sentences, 0's will be appended at the end of the sequence vector) and fixed the sentence length at 200 words using the "pad sequences" function from the "Keras" library. We can begin building our deep-learning models as soon as we have finished padding our sequence vectors.

5. Practical Implementation

```
# Importing the necessary libraries
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

# Training data
train=pd.read_csv("train.csv")
print("There are {} comments".format(len(train)))
train.head()
```

```
There are 159571 comments
```

	id	comment_text	toxic	severe_toxic	obscene	threat	insult	identity_hate
0	0000997932d777bf	Explanation\nWhy the edits made under my usern...	0	0	0	0	0	0
1	000103f0d9cfb60f	D'aww! He matches this background colour I'm s...	0	0	0	0	0	0
2	000113f07ec002fd	Hey man, I'm really not trying to edit war. It...	0	0	0	0	0	0
3	0001b41b1c6bb37e	"\nMore\nI can't make any real suggestions on ...	0	0	0	0	0	0
4	0001d958c54c6e35	You, sir, are my hero. Any chance you remember...	0	0	0	0	0	0

```
# Test data
test=pd.read_csv("test.csv")
print("There are {} comments".format(len(test)))
test.head()
```

```
There are 153164 comments
```

	id	comment_text
0	00001cee341fdb12	Yo bitch Ja Rule is more succesful then you'll...
1	0000247867823ef7	== From RfC == \n\n The title is fine as it is...
2	00013b17ad220c46	" \n\n == Sources == \n\n * Zawe Ashton on Lap...
3	00017563c3f7919a	:If you have a look back at the source, the in...
4	00017695ad8997eb	I don't anonymously edit articles at all.

```
print("Train shape is",train.shape)
print("Test shape is",test.shape)
print("Description of train data set",train.describe())
print("Description of test data set",test.describe())
```

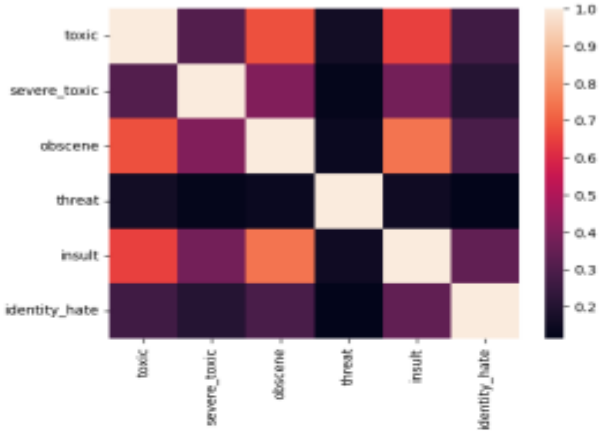
```
Train shape is (159571, 8)
Test shape is (153164, 2)
Description of train data set                toxic   severe_toxic
obscene         threat  \
count  159571.000000  159571.000000  159571.000000  159571.000000
mean        0.095844       0.009996       0.052948       0.002996
std         0.294379       0.099477       0.223931       0.054650
min         0.000000       0.000000       0.000000       0.000000
25%         0.000000       0.000000       0.000000       0.000000
50%         0.000000       0.000000       0.000000       0.000000
75%         0.000000       0.000000       0.000000       0.000000
max         1.000000       1.000000       1.000000       1.000000

              insult  identity_hate
count  159571.000000  159571.000000
mean        0.049364       0.008805
std         0.216627       0.093420
min         0.000000       0.000000
25%         0.000000       0.000000
50%         0.000000       0.000000
75%         0.000000       0.000000
max         1.000000       1.000000
Description of test data set                        id
comment_text
count              153164
153164
unique             153164
153164
top     00001cee341fdb12  Yo bitch Ja Rule is more succesful then
you'll...
freq                    1
1
```

```
# Checking for null values
print(train.isnull().sum())
```

```
id               0
comment_text     0
toxic            0
severe_toxic     0
obscene          0
threat           0
insult           0
identity_hate    0
dtype: int64
```

```
# Checking correlation in the dataset
print(sns.heatmap(train.corr()))
plt.show()
```

```
AxesSubplot(0.125,0.11;0.62x0.77)
```

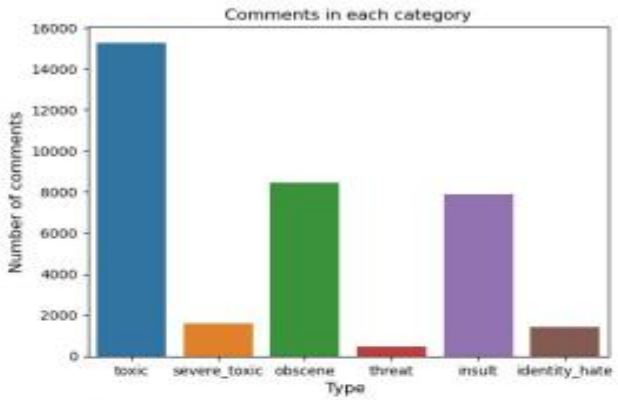



```
# checking the skewness for the features:
train.skew()
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_12244\3462418898.py:2:
FutureWarning: Dropping of nuisance columns in DataFrame reductions
(with 'numeric_only=None') is deprecated; in a future version this
will raise TypeError.  Select only valid columns before calling the
reduction.
  train.skew()
```

```
toxic             2.745854
severe_toxic      9.851722
obscene           3.992817
threat           18.189001
insult            4.160540
identity_hate    10.515923
dtype: float64
```

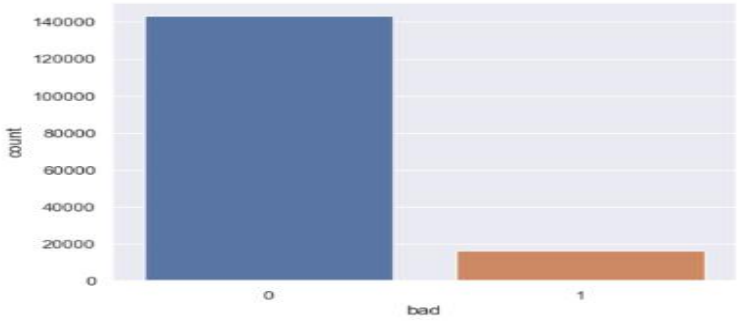
```
columns = ['toxic', 'severe_toxic', 'obscene', 'threat', 'insult',
'identity_hate']
df= train.groupby(columns)\
                  .size()\
                  .sort_values(ascending=False)\
```




```
# Wheter a row has any of the 6 types of toxicity?
# Prints the count of unique value in 'bad' column

cols_target =
['toxic','severe_toxic','obscene','threat','insult','identity_hate']
target_data = train[cols_target]

train['bad'] =train[cols_target].sum(axis =1)
print(train['bad'].value_counts())
train['bad'] = train['bad'] > 0
train['bad'] = train['bad'].astype(int)
print(train['bad'].value_counts())

# plot
sns.set()
sns.countplot(x="bad" , data = train)
plt.show()
```

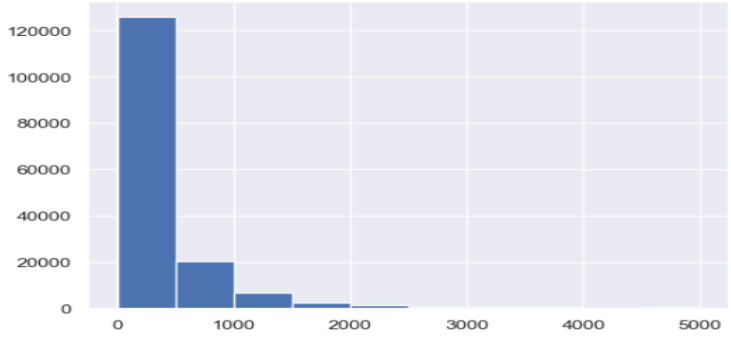
```
0    143346
1      6360
3      4209
2      3480
4      1760
5       385
```

```
6         31
Name: bad, dtype: int64
0    143346
1     16225
Name: bad, dtype: int64
```




```
# Comment's length distribution
train['len']= train['comment_text'].str.len()
print('Max length: {}, Min length: {}, Average Length :
{}'.format(max(train['len']),min(train['len']),train['len'].mean()))
train['len'].hist()
```

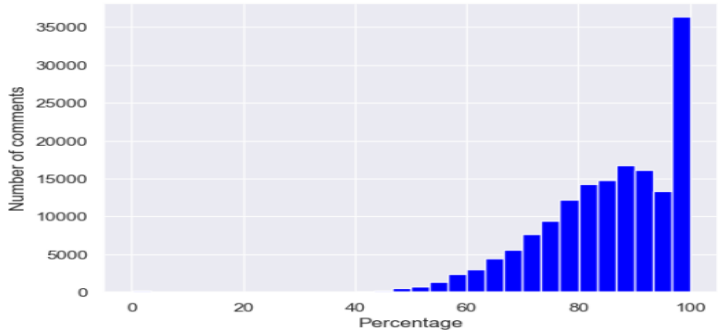
```
Max length: 5000, Min length: 5, Average Length :  394.13883475067524
<AxesSubplot:>
```




```
# Counting unique words
count_word = train["comment_text"].apply(lambda x:
len(str(x).split()))
#Unique word count
count_unique_word = train["comment_text"].apply(lambda x:
len(set(str(x).split())))
unique_percent = count_unique_word*100/count_word
unique_percent
```

```
0          95.348837
1         100.000000
2          92.857143
3          72.566372
4         100.000000
              ...
159566     93.617021
159567    100.000000
159568    100.000000
159569     92.000000
159570     88.888889
Name: comment_text, Length: 159571, dtype: float64
```

```
unique_percent.hist(bins=30,color='blue')
plt.suptitle("Plot for Unique word distribution")
plt.xlabel("Percentage")
```




```
from nltk.stem import WordNetLemmatizer
import nltk
from nltk.corpus import  stopwords
import string

nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('omw-1.4')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\HP\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\HP\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package omw-1.4 to
[nltk_data]     C:\Users\HP\AppData\Roaming\nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!

True
```

```
3           397
4            47
```

```
# Length after cleaning
print ('Original Length', train.len.sum())
print ('Clean Length', train.clean_length.sum())
```

```
Original Length 62893128
Clean Length 43575180
```

```
# toxic and severe_toxic comments-Word cloud
from wordcloud import WordCloud
hams = train['comment_text'][train['toxic']==1]
spam_cloud =
WordCloud(width=600,height=400,background_color='black',max_words=50).
generate(' '.join(hams))
plt.figure(facecolor='k')
plt.imshow(spam_cloud)
plt.axis('off')
plt.savefig('wordcloud.png', facecolor='k', bbox_inches='tight')
```


```
hams = train['comment_text'][train['severe_toxic']==1]
spam_cloud =
WordCloud(width=600,height=400,background_color='black',max_words=50).
generate(' '.join(hams))
plt.figure(facecolor='k')
plt.imshow(spam_cloud)
plt.axis('off')
plt.savefig('wordcloud.png', facecolor='k', bbox_inches='tight')
```


```
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.metrics import
accuracy_score,classification_report,confusion_matrix,f1_score
from sklearn.metrics import roc_curve, roc_auc_score
from sklearn.model_selection import cross_val_score,GridSearchCV
from lightgbm import LGBMClassifier
from sklearn.linear_model import LogisticRegression
from xgboost import XGBClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import KFold

#  Convert text into vectors using TF-IDF
from sklearn.feature_extraction.text import TfidfVectorizer
tf_vec = TfidfVectorizer(max_features = 10000, stop_words='english')
features = tf_vec.fit_transform(train['comment_text'])
x = features

print(train.shape)
print(test.shape)

(159571, 11)
(153164, 2)

y=train['bad']
x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=46,tes
t_size=.20)

# LogisticRegression
LG = LogisticRegression(C=1, max_iter = 3000)
LG.fit(x_train, y_train)
```

```
y_pred_train = LG.predict(x_train)
print('Training accuracy is {}'.format(accuracy_score(y_train,
y_pred_train)))
y_pred_test = LG.predict(x_test)
print('Test accuracy is
{}'.format(accuracy_score(y_test,y_pred_test)))
print(confusion_matrix(y_test,y_pred_test))
print(classification_report(y_test,y_pred_test))
```

```
Training accuracy is 0.960346556370245
Test accuracy is 0.9547234842550525
[[28512   183]
 [ 1262  1958]]
              precision    recall  f1-score   support

           0       0.96      0.99      0.98     28695
           1       0.91      0.61      0.73      3220

    accuracy                           0.95     31915
   macro avg       0.94      0.80      0.85     31915
weighted avg       0.95      0.95      0.95     31915
```

```
# xgboost
import xgboost
xgb = xgboost.XGBClassifier()
xgb.fit(x_train, y_train)
y_pred_train = xgb.predict(x_train)
print('Training accuracy is {}'.format(accuracy_score(y_train,
y_pred_train)))
y_pred_test = xgb.predict(x_test)
print('Test accuracy is
{}'.format(accuracy_score(y_test,y_pred_test)))
print(confusion_matrix(y_test,y_pred_test))
print(classification_report(y_test,y_pred_test))
```

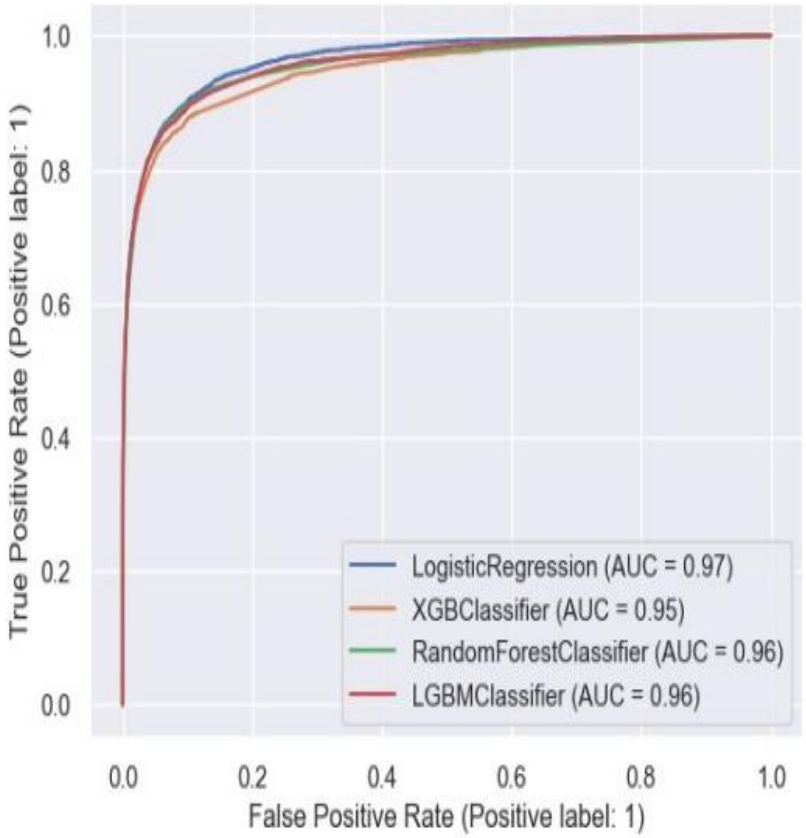
```
Training accuracy is 0.9616625932192768
Test accuracy is 0.9531568228105907
[[28496   199]
 [ 1296  1924]]
              precision    recall  f1-score   support

           0       0.96      0.99      0.97     28695
           1       0.91      0.60      0.72      3220

    accuracy                           0.95     31915
   macro avg       0.93      0.80      0.85     31915
weighted avg       0.95      0.95      0.95     31915
```

```
#RandomForestClassifier
RF = RandomForestClassifier()

RF.fit(x_train, y_train)
y_pred_train = RF.predict(x_train)
print('Training accuracy is {}'.format(accuracy_score(y_train,
y_pred_train)))
y_pred_test = RF.predict(x_test)
print('Test accuracy is
{}'.format(accuracy_score(y_test,y_pred_test)))
print(confusion_matrix(y_test,y_pred_test))
print(classification_report(y_test,y_pred_test))
```

```
Training accuracy is 0.9986761295982954
Test accuracy is 0.9545668181106063
[[28289   406]
 [ 1044  2176]]
              precision    recall  f1-score   support

           0       0.96      0.99      0.98     28695
           1       0.84      0.68      0.75      3220

    accuracy                           0.95     31915
   macro avg       0.90      0.83      0.86     31915
weighted avg       0.95      0.95      0.95     31915
```

```
# LGBM Classifier
lgbm = LGBMClassifier()
lgbm.fit(x_train, y_train)
y_pred_train = lgbm.predict(x_train)
print('Training accuracy is {}'.format(accuracy_score(y_train,
y_pred_train)))
y_pred_test = lgbm.predict(x_test)
print('Test accuracy is
{}'.format(accuracy_score(y_test,y_pred_test)))
print(confusion_matrix(y_test,y_pred_test))
print(classification_report(y_test,y_pred_test))
```

```
Training accuracy is 0.9609262392680329
Test accuracy is 0.9547234842550525
[[28468   227]
 [ 1218  2002]]
              precision    recall  f1-score   support

           0       0.96      0.99      0.98     28695
           1       0.90      0.62      0.73      3220

    accuracy                           0.95     31915
   macro avg       0.93      0.81      0.86     31915
```




```
# Making predictions on test data
test_data =tf_vec.fit_transform(test['comment_text'])
test_data
prediction=LG.predict(test_data) # using the Logistic regression model
prediction
```

```
array([0, 0, 0, ..., 0, 0, 0])
```

6. Result

A classifier for Malignant or toxic comments that operates online and forecasts from raw data. Based on the AUC-ROC score the Logistic regression model performs the best.

7. Conclusion

Returning to the original issue, how can we apply this approach to improve online discourse etiquette? It depends on the platform, I suppose. For instance, in online gaming, you would want to identify toxic gamers before enforcing limits based on the frequency or seriousness of crimes.
Social media: You might not want to restrict people's freedom of speech on social media sites like Facebook and Twitter, though. People might be more cautious about what they say and, hopefully, would reconsider posting the poisonous comment they were about to make if real time toxicity trackers were displayed above comments as they were being typed. Users who have a history of acting rudely online won't need to have toxicity trackers added to them. We can use AI to identify these people, much as in gaming, and then apply the toxicity tracker to them. Performing a case study to examine if individuals who have a history of being toxic alter their conduct while participating in online conversations with it turned on might be fascinating.

8. Future Scope

We have discovered that machine learning holds the key to classifying text and using predictions to gauge its toxicity. There are several locations where we can get more precise when identifying the harmful statement and its level of toxicity.

As many comments are incorrectly categorised as "hate," we can concentrate on functionality and error assessment of the model in our future work. In the past, many algorithms were successfully used on data in the English language, but in the future, we might think of having data in local languages. We can indeed work on the follow-up steps for toxic remark identification, such as automatic user blocking and automatic removal of offensive comments from social media sites.

References

Chu, T., Jue, K., and Wang, M. (2017b). Comment abuse classification with deep learning. Technical report, Stanford

University.http://web.stanford.edu/class/cs224n/reports/2762092.pdf**.**

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. Neural Computation, 9(8):1735–1780.

Yenala, H., Chinnakotla, M. K., and Goyal, J. (2017). Convolutional bi-directional LSTM for detecting inappropriate query suggestions in web search. In Advances in Knowledge Discovery and Data Mining, pages 3–16, Jeju, South Korea.