May 23, 2018

# *Toxic Comment Cla****Toxic Comment Classification***

## *Identify and classify toxic online comments*

AMIT CHAKRABORTY



*CONTENTS*

* *INTRODUCTION*
* *DATA*
* *APPROACH(TEXT MINING)*
* *DATA PRE-PROCESSING*

1. *STOP-WORD REMOVAL*
2. *PUNCTUATION REMOVAL*

* *VISULIZATION OF THE TEXT(WORD-CLOUD)*
* *MACHINE LEARNING*

1. *SENTIMENT ANALYSIS*
2. *VADAR ANALYSIS*

* *CALCULATION FOR TEXT PREDICTION*
* *PYTHON CODE TO PREDICT THE COMMENTS*
* *REFERENCE*
* ***INTRODUCTION:***

Discussing things you care about can be difficult. The threat of abuse and harassment online means that many people stop expressing themselves and give up on seeking different opinions. Platforms struggle to effectively facilitate conversations, leading many communities to limit or completely shut down user comments.

The [Conversation AI](https://conversationai.github.io/) team, a research initiative founded by [Jigsaw](https://jigsaw.google.com/) and Google (both a part of Alphabet) are working on tools to help improve online conversation. One area of focus is the study of negative online behaviors, like toxic comments (i.e. comments that are rude, disrespectful or otherwise likely to make someone leave a discussion). So far they’ve built a range of publicly available models served through the [Perspective API](https://perspectiveapi.com/), including toxicity. But the current models still make errors, and they don’t allow users to select which types of toxicity they’re interested in finding (e.g. some platforms may be fine with profanity, but not with other types of toxic content).

In this competition, you’re challenged to build a multi-headed model that’s capable of detecting different types of toxicity like threats, obscenity, insults, and identity-based hate better than Perspective’s [current models](https://github.com/conversationai/unintended-ml-bias-analysis). We will be using a dataset of comments from Wikipedia’s talk page edits. Improvements to the current model will hopefully help online discussion become more productive and respectful.

* ***Data:🡪***

We are provided with a large number of Wikipedia comments which have been labeled by human raters for toxic behavior. The types of toxicity are:

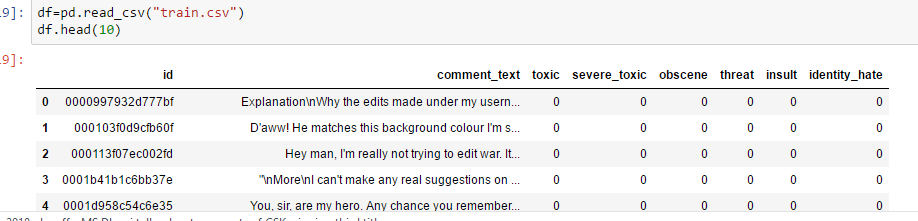
1. toxic
2. severe\_toxic
3. obscene
4. threat
5. insult
6. identity\_hate

## ***File descriptions***

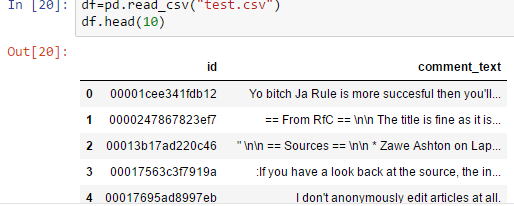
* ***train.csv****- the training set, contains comments with their binary labels*
* ***test.csv****- the test set, we must predict the toxicity probabilities for these comments. To deter hand labeling, the test set contains some comments which are not included in scoring.*

*Below is the view of the some comments.*

*Train Data-set:*

**

*Test Data-Set:*

**

Approach:

* *Text Mining:*

*Text mining, also referred to as text data mining, roughly equivalent to text analytics, is the process of deriving high-quality*[*information*](https://en.wikipedia.org/wiki/Information)*from*[*text*](https://en.wikipedia.org/wiki/Plain_text)*. High-quality information is typically derived through the devising of patterns and trends through means such as*[*statistical pattern learning*](https://en.wikipedia.org/wiki/Pattern_recognition)*. Text mining usually involves the process of structuring the input text (usually parsing, along with the addition of some derived linguistic features and the removal of others, and subsequent insertion into a*[*database*](https://en.wikipedia.org/wiki/Database)*), deriving patterns within the*[*structured data*](https://en.wikipedia.org/wiki/Structured_data)*, and finally evaluation and interpretation of the output. 'High quality' in text mining usually refers to some combination of*[*relevance*](https://en.wikipedia.org/wiki/Relevance_(information_retrieval))*,*[*novelty*](https://en.wikipedia.org/wiki/Novelty_(patent))*, and interestingness. Typical text mining tasks include*[*text categorization*](https://en.wikipedia.org/wiki/Text_categorization)*,*[*text clustering*](https://en.wikipedia.org/wiki/Text_clustering)*,*[*concept/entity extraction*](https://en.wikipedia.org/wiki/Concept_mining)*, production of granular taxonomies,*[*sentiment analysis*](https://en.wikipedia.org/wiki/Sentiment_analysis)*,*[*document summarization*](https://en.wikipedia.org/wiki/Document_summarization)*, and entity relation modeling (i.e., learning relations between*[*named entities*](https://en.wikipedia.org/wiki/Named_entity_recognition)*).*

*Text analysis involves*[*information retrieval*](https://en.wikipedia.org/wiki/Information_retrieval)*,*[*lexical analysis*](https://en.wikipedia.org/wiki/Lexical_analysis)*to study word frequency distributions,*[*pattern recognition*](https://en.wikipedia.org/wiki/Pattern_recognition)*,*[*tagging*](https://en.wikipedia.org/wiki/Tag_(metadata))*/*[*annotation*](https://en.wikipedia.org/wiki/Annotation)*,*[*information extraction*](https://en.wikipedia.org/wiki/Information_extraction)*, mining techniques including link and association analysis,*[*visualization*](https://en.wikipedia.org/wiki/Information_visualization)*, and*[*predictive analytics*](https://en.wikipedia.org/wiki/Predictive_analytics)*. The overarching goal is, essentially, to turn text into data for analysis, via application of*[*natural language processing*](https://en.wikipedia.org/wiki/Natural_language_processing)*(NLP) and analytical methods.*

*A typical application is to scan a set of documents written in a*[*natural language*](https://en.wikipedia.org/wiki/Natural_language)*and either model the document set for*[*predictive classification*](https://en.wikipedia.org/wiki/Predictive_classification)*purposes or populate a database or search index with the information extracted*

*Text- Mining Pre Processing:*

***Stop Word Removal***

*Many words in documents recur very frequently but are essentially meaningless as*

*They are used to join words together in a sentence. It is commonly understood that stop*

*Words do not contribute to the context or content of textual documents. Due to their*

*High frequency of occurrence, their presence in text mining presents an obstacle in*

*understanding the content of the documents. Stop words are very frequently used*

*common words like ‘and’, ‘are’, ‘this’ etc They are not useful in classification of*

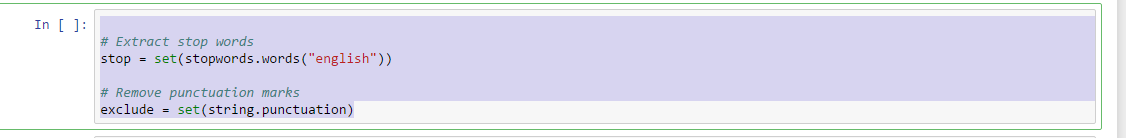
*documents. So they must be removed. However, the development of such stop*

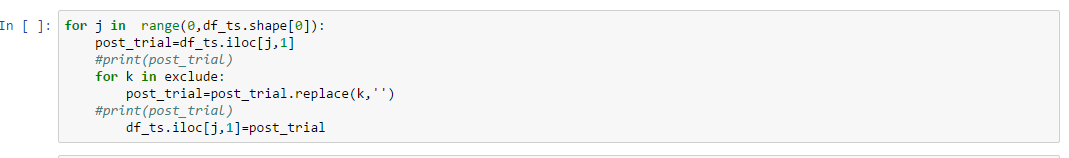
*words list is difficult and inconsistent between textual sources. This process also*

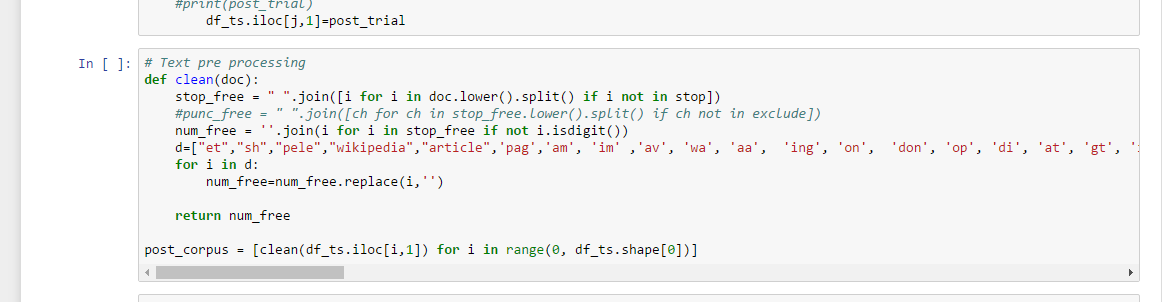
*reduces the text data and improves the system performance. Every text document deals with*

*these words which are not necessary for text mining applications.*

*Below are the line I used to remove the STOPWORD and PUNCTUATION from the raw data set:*

**

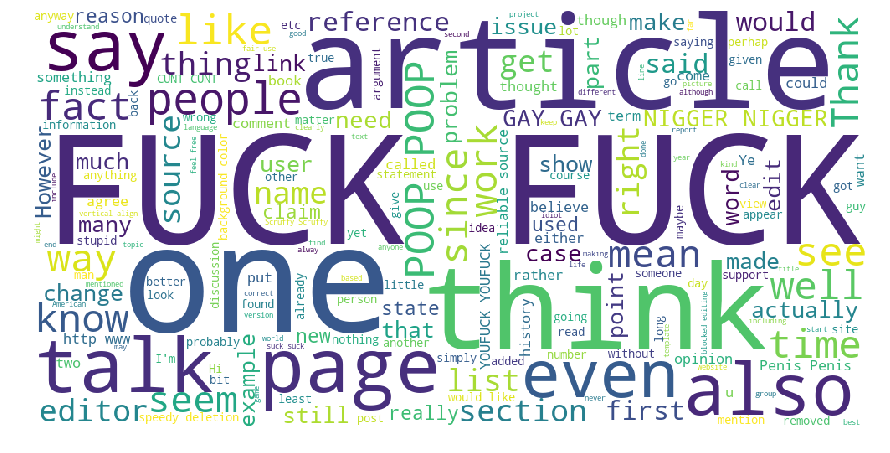
**

**

***Word-Cloud”🡪***

*A****tag cloud****(****word cloud****, or****weighted list****in visual design) is a novelty visual representation of text data, typically used to depict*[*keyword metadata (tags)*](https://en.wikipedia.org/wiki/Tag_(metadata))*on websites, or to visualize free form text. Tags are usually single words, and the importance of each tag is shown with font size or color. This format is useful for quickly perceiving the most prominent terms and for locating a term alphabetically to determine its relative prominence. When used as website navigation aids, the terms are hyperlinked to items associated with the tag*.

Below is the representation of the Word-Cloud used for the data-set with Toxic Words.



After applying the word- cloud for every toxic , severe toxic or other variable I got the following words which frequency is more in the whole data-set

fucky,pissed,piss,fucked,tosser,dumb,fartchina,shhole,fucksex,gay,shut,idiot,stupid,balls,ass,hate,sex,fuck,cocksucker,fucker,cocksucker,nigger,bitchesfuck,bollocks,dickhead,cunt,bastard,moron,tommy,bitches,bark,useless,bastard,ass,bollocks,stupid,nigger,loser,mor.bchesfuck,bitch,sucku,sexsex,penis,penissmall,bark,gay,nigga,bunksteve,vandalism,,pig,fat,jew,suck,,cucker,,frickin,,mexican,,fag,anal,,bchfuck,,rape,assfuck,hate,die,wanker,poop,sexual,homo,heil,nipple,bullshit,shit,buttsecks,

For every data set the following words has been used in a very frequent way. So for determining a toxic comment the above words plays an important role. More over in the data-set there was some other un-usual words which was needed to be removed.

Some of the words has been given below:

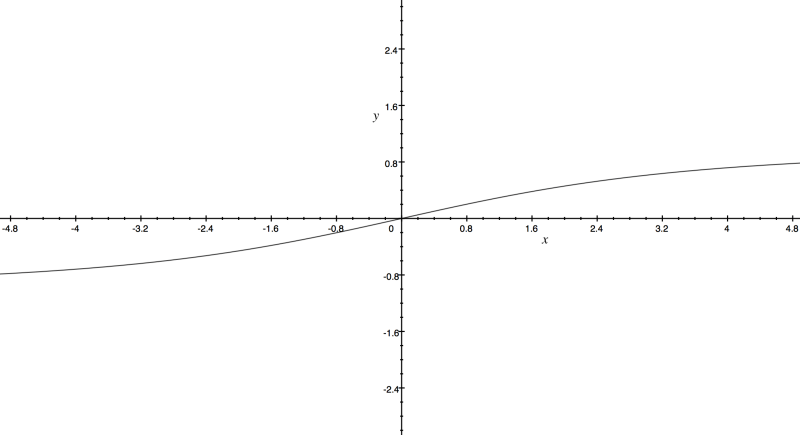
"et","sh","pele","wikipedia","article",'pag','am', 'im' ,'av', 'wa', 'aa', 'ing', 'on', 'don', 'op', 'di', 'at', 'gt', 'it', 'lik','dont','page’,’ get', 'like’,’ second', 'life','white', 'holocaust', 'research', 'mum', 'full', 'bia', 'spreading', 'wide', 'topics', 'scjessey','dvd', 'mentions', 'guide', 'mike', 'knob', 'ignores','example','spread', 'false', 'information', 'person', 'ignorant', 'sources', 'let', 'arses', 'user', 'makes','make', 'first', 'last', 'warning','jews', 'shave', 'head', 'bald', 'go', 'meetings', 'doubt', 'words', 'bible','uh', 'two', 'become', 'done', 'didnt', 'take', 'long', 'nope', 'certainly', 'aware', 'sitting', 'front',

* ***Machine Learning(Sentiment Analysis):***

*Opinion mining (sometimes known as sentiment analysis or emotion AI) refers to the use of*[*natural language processing*](https://en.wikipedia.org/wiki/Natural_language_processing)*,*[*text analysis*](https://en.wikipedia.org/wiki/Text_analytics)*,*[*computational linguistics*](https://en.wikipedia.org/wiki/Computational_linguistics)*, and*[*biometrics*](https://en.wikipedia.org/wiki/Biometrics)*to systematically identify, extract, quantify, and study affective states and subjective information. Sentiment analysis is widely applied to*[*voice of the customer*](https://en.wikipedia.org/wiki/Voice_of_the_customer)*materials such as reviews and survey responses, online and social media, and healthcare materials for applications that range from*[*marketing*](https://en.wikipedia.org/wiki/Marketing)*to*[*customer service*](https://en.wikipedia.org/wiki/Customer_relationship_management)*to clinical medicine.*

*Generally speaking, sentiment analysis aims to determine the attitude of a speaker, writer, or other subject with respect to some topic or the overall contextual polarity or emotional reaction to a document, interaction, or event. The attitude may be a judgment or evaluation (see*[*appraisal theory*](https://en.wikipedia.org/wiki/Appraisal_theory)*), affective state (that is to say, the emotional state of the author or speaker), or the intended emotional communication (that is to say, the emotional effect intended by the author or interlocutor).*

# *VADER Sentiment Analysis:*

* VADER (Valence Aware Dictionary for sEntiment Reasoning) is a model used for text sentiment analysis that is sensitive to both polarity (positive/negative) and intensity (strength) of emotion. Introduced in 2014, VADER text sentiment analysis uses a human-centric approach, combining qualitative analysis and empirical validation by using human raters and the wisdom of the crowd.*

*In this post, I’ll discuss how VADER sentiment analysis calculates the sentiment score of an input text. It combines a dictionary, which maps lexical features to emotion intensity, and five simple heuristics, which encode how contextual elements increment, decrement, or negate the sentiment of text.*

*VADER sentiment analysis (well, in the Python implementation anyway) returns a sentiment score in the range -1 to 1, from most negative to most positive.*

*The sentiment score of a sentence is calculated by summing up the sentiment scores of each VADER-dictionary-listed word in the sentence. Cautious readers would probably notice that there is a contradiction: individual words have a sentiment score between -4 to 4, but the returned sentiment score of a sentence is between -1 to 1.*

*They’re both true. The sentiment score of a sentence is the sum of the sentiment score of each sentiment-bearing word. However, we apply a normalization to the total to map it to a value between -1 to 1.*

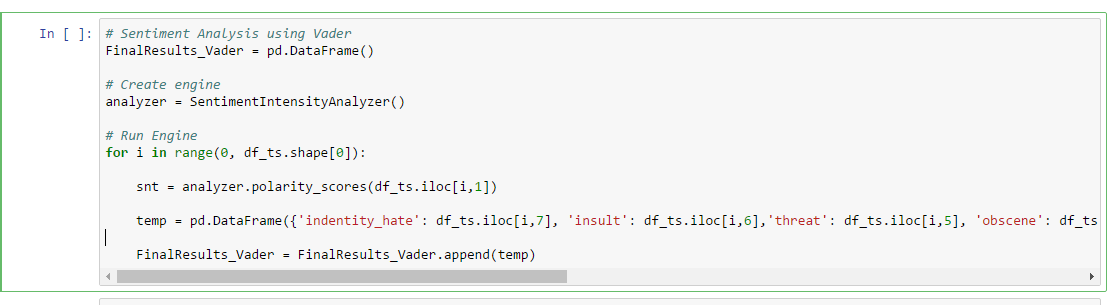
*The normalization used by Hutto is*

*\large \dfrac{x}{\sqrt{x^2 + \alpha}}*

*where x is the sum of the sentiment scores of the constituent words of the sentence and \alpha is a normalization parameter that we set to 15. The normalization is graphed below.*

*We see here that as x grows larger, it gets more and more close to -1 or 1. To similar effect, if there are a lot of words in the document you’re applying VADER sentiment analysis to, you get a score close to -1 or 1. Thus, VADER sentiment analysis works best on short documents, like tweets and sentences, not on large documents.*

***Below is the Python Code to run the Vadar Analysis Engine:***

******

* ***Calculation:***

*With Toxic value 1 total number of observation(Toxic)=15294*

*With Toxic value 1 total number of observation(Severe\_Toxic)=41*

*So change in percentile=41/15294 \*100=26%*

*By doing the same calculation we can get the following value in percentile with respect Toxic value =1*

*Treat=73*

*Obscene=11.53*

*Insult:7.91*

*Identity\_Hate=0.89*

* *Python Code for the Comment Analysis:*

*import csv*

*import os*

*import pandas as pd*

*from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer*

*from nltk.corpus import stopwords*

*import string*

*import nltk*

*import textmining*

*import matplotlib.pyplot as plt*

*from wordcloud import WordCloud*

*os.chdir("S:/New folder/train.csv")*

*df=pd.read\_csv("train.csv")*

*df\_ts.head(2)*

*# Extract stop words*

*stop = set(stopwords.words("english"))*

*# Remove punctuation marks*

*exclude = set(string.punctuation)*

*for j in range(0,df\_ts.shape[0]):*

*post\_trial=df\_ts.iloc[j,1]*

*#print(post\_trial)*

*for k in exclude:*

*post\_trial=post\_trial.replace(k,'')*

*#print(post\_trial)*

*df\_ts.iloc[j,1]=post\_trial*

*# Text pre processing*

*def clean(doc):*

*stop\_free = " ".join([i for i in doc.lower().split() if i not in stop])*

*#punc\_free = " ".join([ch for ch in stop\_free.lower().split() if ch not in exclude])*

*num\_free = ''.join(i for i in stop\_free if not i.isdigit())*

*d=["et","sh","pele","wikipedia","article",'pag','am', 'im' ,'av', 'wa', 'aa', 'ing', 'on', 'don', 'op', 'di', 'at', 'gt', 'it', 'lik','dont','page','get', 'like','second', 'life','white', 'holocaust', 'research', 'mum', 'full', 'bias', 'spreading', 'wide', 'topics', 'scjessey','dvd', 'mentions', 'guide', 'mike', 'knob', 'ignores','example','spread', 'false', 'information', 'person', 'ignorant', 'sources', 'let', 'arses', 'user', 'makes','make', 'first', 'last', 'warning','jews', 'shave', 'head', 'bald', 'go', 'meetings', 'doubt', 'words', 'bible','uh', 'two', 'become', 'done', 'didnt', 'take', 'long', 'nope', 'certainly', 'aware', 'sitting', 'front',]*

*for i in d:*

*num\_free=num\_free.replace(i,'')*

*return num\_free*

*post\_corpus = [clean(df\_ts.iloc[i,1]) for i in range(0, df\_ts.shape[0])]*

*df\_trial=pd.DataFrame({'Comm':post\_corpus})*

*# Create document term matrix*

*tdm\_severetoxic = textmining.TermDocumentMatrix()*

*for i in post\_corpus:*

*tdm\_severetoxic.add\_doc(i)*

*df\_tri=pd.DataFrame({'Comm':post\_corpus})*

*#Plot wordcloud*

*wordcloud = WordCloud(width = 1000, hieght = 500, stopwords = STOPWORDS, background\_color = 'white').generate(*

*''.join(df\_trial['Comm']))*

*plt.figure(figsize = (15,8))*

*plt.imshow(wordcloud)*

*plt.axis('off')*

*plt.show()*

*import os*

*os.chdir("S:/New folder/train.csv")*

*List=open("toxics.txt").readlines()*

*l=[]*

*for i in List:*

*l.append(i.lower().split(","))*

*# Create engine*

*FinalResults\_Vader = pd.DataFrame()*

*analyzer = SentimentIntensityAnalyzer()*

*h=int(input("Enter the number of comment u want to check"))*

*k=[]*

*for i in range(0,h):*

*a=df.iloc[i,1]*

*for k in exclude:*

*a=a.replace(k,'')*

*x=''.join(a)*

*temp=0*

*snt = analyzer.polarity\_scores(a)*

*b=list(snt.items())[3][1]*

*for j in l[0]:*

*for i in x.split():*

*#print("x is ",i)*

*if(i.lower()==j):*

*analyzer.polarity\_scores(j)*

*temp=temp+list(snt.items())[3][1]*

*#print("Temp in the loop")*

*b=0*

*temp=b+temp*

*#print("Temp outside the loop",temp)*

*if(temp>0.8):*

*print("This is good comment.",temp)*

*elif(temp>=-1 or temp<0.8):*

*# c = pd.DataFrame({'Toxic': temp,'treat':(temp\*0.73)/100,'severe\_toxic':(temp\*.27)/100,'obsence':(temp\*11.52)/100,'insult':(temp\*7.91)/100,'hate':(0.89\*temp)/100}, index = [0])*

*dict\_test={'Toxic': temp,'treat':(temp\*0.73)/100,'severe\_toxic':(temp\*.27)/100,'obsence':(temp\*11.52)/100,'insult':(temp\*7.91)/100,'hate':(0.89\*temp)/100}*

*c = pd.DataFrame.from\_dict(list(dict\_test.items()))*

*pd.DataFrame.from\_dict(dict\_test, orient = 'index')*

*s = pd.Series(dict\_test, name = a)*

*dl = pd.DataFrame(s)*

*new=dl.T*

*FinalResults\_Vader = FinalResults\_Vader.append(new)*

*FinalResults\_Vader.to\_csv("My Final value.csv")*

*print("Negative toxity is saved in My Final value.csv")*

*Reference:*

* <http://datameetsmedia.com/vader-sentiment-analysis-explained/>
* <https://en.wikipedia.org/wiki/Sentiment_analysis>
* https://stackoverflow.com/questions/15899861/efficient-term-document-matrix-with-nltk