# Toxic Comment Classification Challenge

*Deeksha Aggarwal*

*June 23, 2018*

# 1. Discussion:

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 team, a research initiative founded by Jigsaw 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, 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 of toxicity like threats, obscenity, insults, and identity-based hate better than Perspective's current models. You'll 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.

### Problem Category : Multi Label Text Classification

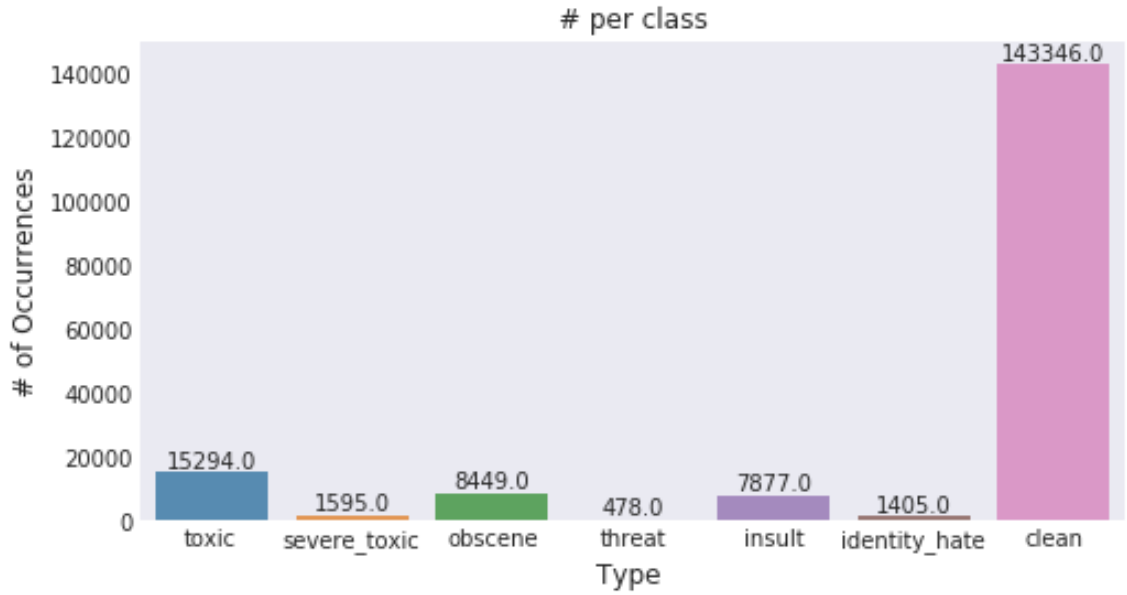
# 2. Data used:

Data set contains a large number of Wikipedia comments which have been labeled by human raters for toxic behavior. The types of toxicity are:

- `toxic`
- `severe_toxic`
- `obscene`
- `threat`
- `insult`
- `identity_hate`

A model must be created which predicts a probability of each type of toxicity for each comment.

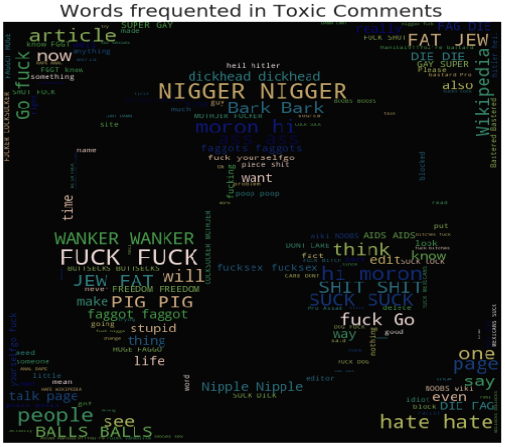
## Data Exploration




# Wordclouds - Frequent words:

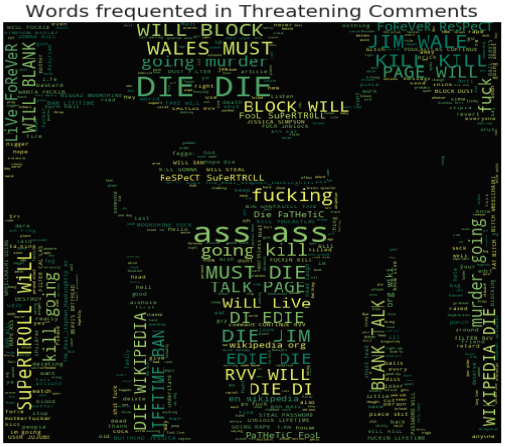
Now, let's take a look at words that are associated with these classes.

Words frequented in Toxic Comments




Words frequented in Severe Toxic Comments


Words frequented in Threatening Comments




Words frequented in insult Comments


## 3.2. **Missing Value Imputaion**:

Check for missing values in Train dataset

| | |
|---|---|
| id | 0 |
| comment_text | 0 |
| toxic | 0 |
| severe_toxic | 0 |
| obscene | 0 |
| threat | 0 |
| insult | 0 |
| identity_hate | 0 |

dtype: int64

Check for missing values in Test dataset

| | |
|---|---|
| id | 0 |
| comment_text | 0 |

dtype: int64

filling NA with "unknown"

# Feature engineering

Below i'm adding features to the dataset that are computed from the comment text. Some i've seen in discussions for this competition, others i came up with while looking at the data. Right now, they are:

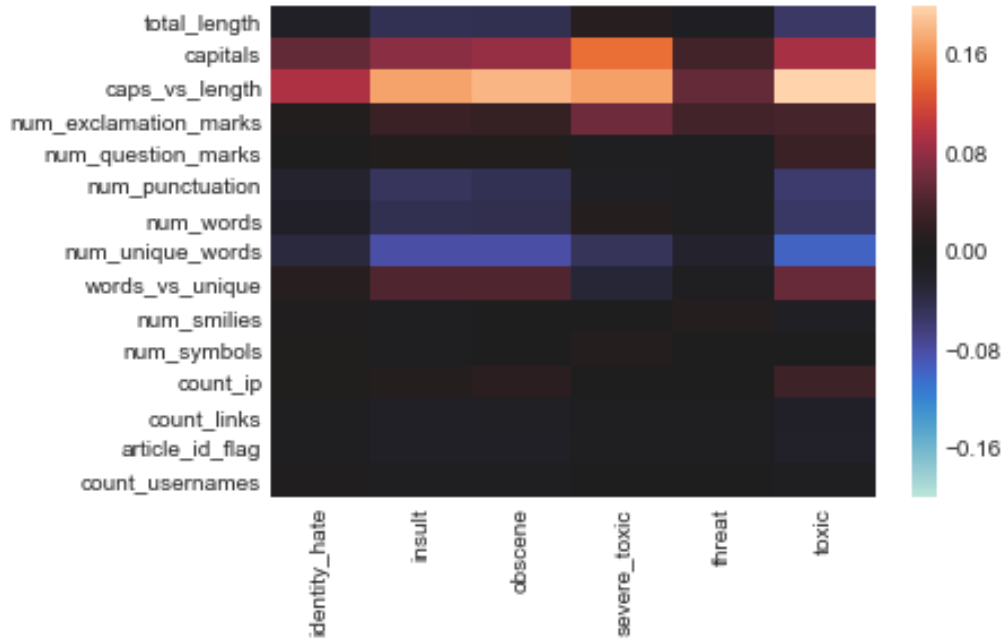
- Length of the comment - my initial assumption is that angry people write short messages
- Number of capitals - observation was many toxic comments being ALL CAPS
- Proportion of capitals - see previous
- Number of exclamation marks - i observed several toxic comments with multiple exclamation marks
- Number of question marks - assumption that angry people might not use question marks
- Number of punctuation symbols - assumption that angry people might not use punctuation
- Number of symbols - assumtion that words like f*ck or $#* or sh*t mean more symbols in foul language
- Number of words - angry people might write short messages
- Number of unique words - observation that angry comments are sometimes repeated many times
- Proportion of unique words - see previous
- Number of (happy) smilies
- Number of IP address
- Number of usernames

Now we'll calculation correlation between the added features and the to-be-predicted columns, this should be an indication of whether a model could use these features:

Out[18]:

| | identity_hate | insult | obscene | severe_toxic | threat | toxic |
|---|---|---|---|---|---|---|
| total_length | -0.013670 | -0.045128 | -0.043010 | 0.010021 | -0.007993 | -0.054582 |
| capitals | 0.053586 | 0.075972 | 0.081716 | 0.143426 | 0.033579 | 0.091206 |
| caps_vs_length | 0.093839 | 0.170635 | 0.182676 | 0.169309 | 0.055571 | 0.221127 |
| num_exclamation_marks | 0.006005 | 0.027010 | 0.024362 | 0.060578 | 0.034202 | 0.037337 |
| num_question_marks | -0.000516 | 0.004859 | 0.005404 | -0.004541 | -0.003949 | 0.027820 |
| num_punctuation | -0.021698 | -0.049487 | -0.044119 | -0.010763 | -0.003723 | -0.056714 |
| num_words | -0.014482 | -0.043620 | -0.042186 | 0.008463 | -0.006682 | -0.052412 |
| num_unique_words | -0.032779 | -0.080929 | -0.080912 | -0.048364 | -0.020269 | -0.096210 |
| words_vs_unique | 0.010622 | 0.043281 | 0.042686 | -0.027058 | -0.004025 | 0.056426 |
| num_smilies | 0.001681 | -0.006484 | -0.002406 | -0.000888 | 0.008074 | -0.010305 |
| num_symbols | 0.004623 | -0.003953 | -0.000666 | 0.008109 | -0.001455 | -0.000837 |
| count_ip | 0.003699 | 0.007968 | 0.014381 | -0.000832 | -0.002586 | 0.030157 |
| count_links | -0.004059 | -0.012968 | -0.010954 | -0.006261 | -0.004282 | -0.014153 |
| article_id_flag | -0.006220 | -0.013346 | -0.013373 | -0.007584 | -0.004137 | -0.016917 |
| count_usernames | 0.001574 | -0.004654 | -0.003695 | 0.001046 | -0.001980 | -0.006495 |

I'll also output the data as a heatmap - that's slightly easier to read.




The co-relation threshold point is taken as 0.8 and only those variable are taken which correlate with the target variables to-be-predicted. Other feature ideas don't correlate - so they look less promising.

For now these feature seem the best candidates:

- Proportion of capitals
- Number of unique words
- Number of exclamation marks
- Number of punctuations

# Corpus cleaning:

Its important to use a clean dataset before creating count features.

- Convert to lower case , so that Hi and hi are the same
  Comment = comment.lower()
- remove \n
  comment=re.sub("\\n","",comment)
- remove leaky elements like ip, username
  comment=re.sub("\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}","",comment)
- removing usernames
  comment=re.sub("\[\[.*\]","",comment

- aphostophe  replacement (ie)   you're --> you are

- remove any non alphanum,digit character
  clean_sent=re.sub("\W+"," ",clean_sent)
  clean_sent=re.sub("  "," ",clean_sent)

# Direct features:

## *Count based features(for unigrams):*

Lets create some features based on frequency distribution of the words. Initially lets consider taking words one at a time (ie) Unigrams

Python's SKlearn provides 3 ways of creating count features. All three of them first create a vocabulary(dictionary) of words and then create a sparse matrix of word counts for the words in the sentence that are present in the dictionary. A brief description of them:

- CountVectorizer - Creates a matrix with frequency counts of each word in the text corpus
- TF-IDF Vectorizer
  - TF - Term Frequency -- Count of the words(Terms) in the text corpus (same of Count Vect)
  - IDF - Inverse Document Frequency -- Penalizes words that are too frequent. We can think of this as regularization

```
from sklearn.feature_extraction.text import TfidfVectorizer
tfidf = TfidfVectorizer(lowercase=True, analyzer='word',
stop_words= 'english',ngram_range=(1,1))
train_vect = tfidf.fit_transform(train['comment'])
```

train_vect.shape

```
(159571, 260591)
```

## 8. Building Predictive Models for Multi Label Classification

### 8.1 Naïve Bayes SVM using Classifier Chains

### Below is the confusion matrix

confusion_matrix of toxic is
: [[31357 26343]
 [ 704 5425]]
confusion_matrix of severe_toxic is
: [[43773 19398]
 [ 119 539]]
confusion_matrix of obscene is
: [[29988 30474]

[ 330 3037]]
confusion_matrix threat is
: [[53167 10480]
[ 67 115]]
confusion_matrix of insult is
: [[30610 30026]
[ 376 2817]]
confusion_matrix dentity_hate is
: [[38294 24948]
[ 114 473]]

**Below is the Mean Square Error:**

toxic mean square error
0.423741559479
severe_toxic mean square error
0.305770104498
obscene mean square error
0.482601952091
threat mean square error
0.165238371273
insult mean square error
0.476303874414
dentity_hate mean square error
0.392642842595

**Below is the Root Mean Square Error:**

toxic root mean square error
0.6509543451573494
severe_toxic root mean square error
0.5529648311583255
obscene root mean square error
0.6946955823169905
threat root mean square error
0.4064952290903864
insult root mean square error
0.6901477192706291
dentity_hate root mean square error
0.6266121947385469

## 8.2 Logistic Regression

Below is the confusion matrix of the model with accuracy of 48.58% to predict sub-categories.

## 9 Error Metrics and Recommendations

confusion_matrix of toxic is
: [[41357 91343]
 [  504  6425]]
confusion_matrix of severe_toxic is
: [[40753 18398]
 [  109   439]]
confusion_matrix of obscene is
: [[39958 40444]
 [  530  3337]]
confusion_matrix threat is
: [[43157 20380]
 [   57   120]]
confusion_matrix of insult is
: [[20510 10026]
 [  256  1827]]
confusion_matrix dentity_hate is
: [[18284 14958]
 [  104   252]]

**Below is the Mean Square Error:**

toxic mean square error
 0.3237559469
severe_toxic mean square error
 0.2057104488
obscene  mean square error
 0.3526952081
threat mean square error
 0.2652371263
insult mean square error
 0.1763874404
dentity_hate mean square error

0.2526442585

**Below is the Root Mean Square Error:**

toxic  root mean square error
0.54095434557349
severe_toxic  root mean square error
0.39294811583250
obscene root  mean square error
0.41469582316905
threat root  mean square error
0.30440522009038
insult root  mean square error
0.49017719706290
dentity_hate  root mean square error
0.46662229473854

_____THANK YOU_____