International
Institute of Information
Technology Bangalore

DOCUMENTATION

MACHINE LEARNING

GEN 511

TOXIC COMMENTS CLASSIFICATION

TEAM MEMBERS :
I.Cherish (IMT2017022) cherish.chowdary@iiitb.org
S.Sanjay (IMT2017037) sanjaykumar.reddy@iiitb.org
S.Prasanth (IMT2017525) sivaramannagari.prasanth@iiitb.org

23rd November, 2019

Contents

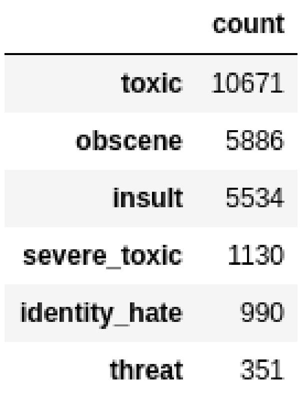
Disclaimer: As the dataset contains offensive text, this paper also contains some of it, for demonstration purposes.

1 Introduction

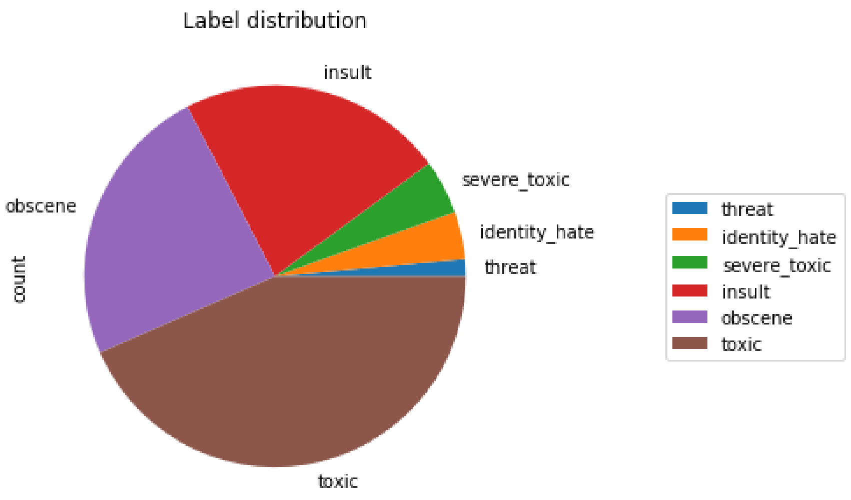
With the recent growth of people on the internet, civil conversations are seeing a decline. A large proportion of online comments present on public domains are usually constructive, however a significant proportion are toxic in nature. Although comment moderators can delete and inform users of breeches of decency and rules for posting comments, these moderators are mostly overwhelmed by the large volume of comments to review. So far we have 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 type of toxicity they're interested in finding.
In this project, we tried to build a multi-headed model that's capable of detecting different types of of toxicity like threats, obscenity, insults, and identity-based hate.

2 Data Visualization

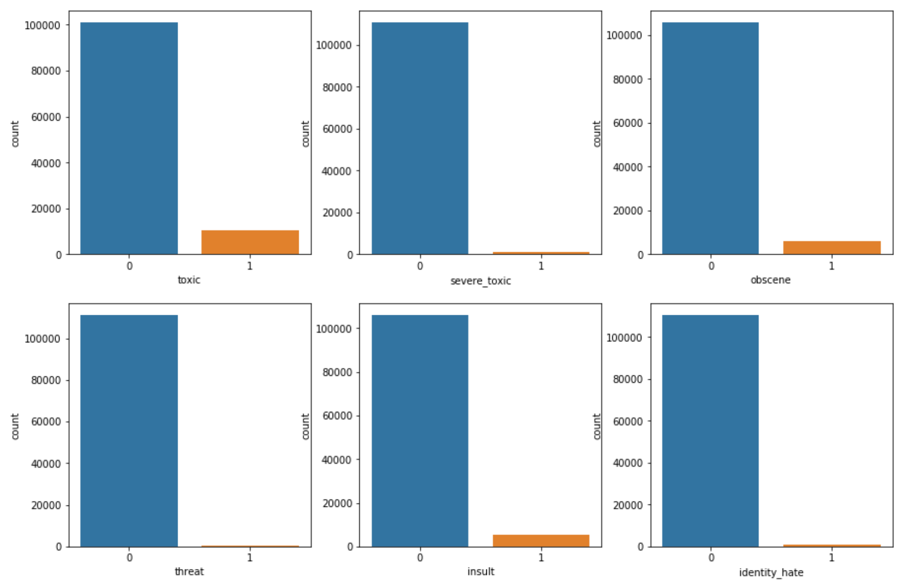
The dataset provided to us has the file: "train.csv". We used python3 and its packages to examine and manipulate the data. The training dataset has 8 columns and 111699 rows. The pie-chart shown below describes the distribution of labels in the dataset.



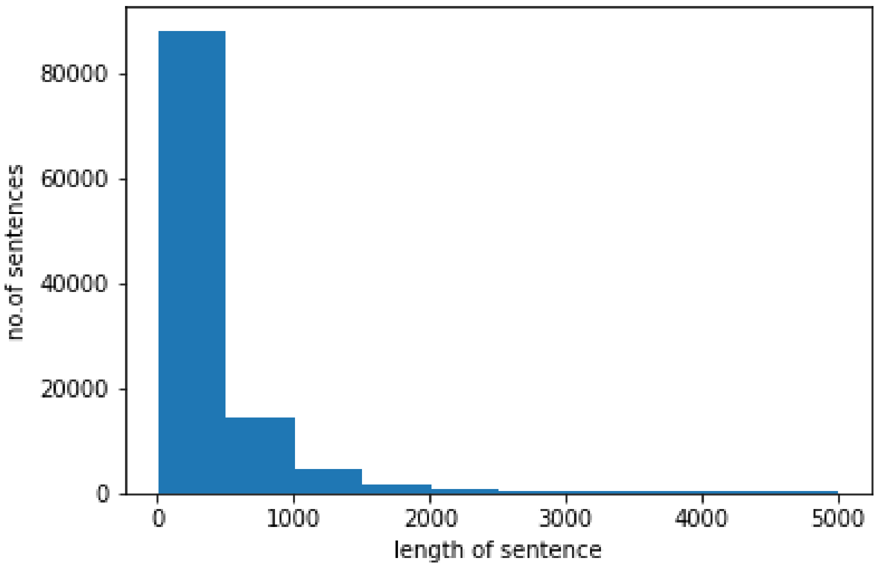
	count
toxic	10671
obscene	5886
insult	5534
severe_toxic	1130
identity_hate	990
threat	351



To get a clear vision of what is in these datasets, the following diagrams were used. From the diagrams, we can observe that the "threat" and "identity hate" classes have the lowest number of comments. The most number of comments belongs to "toxic" and "obscene" classes.

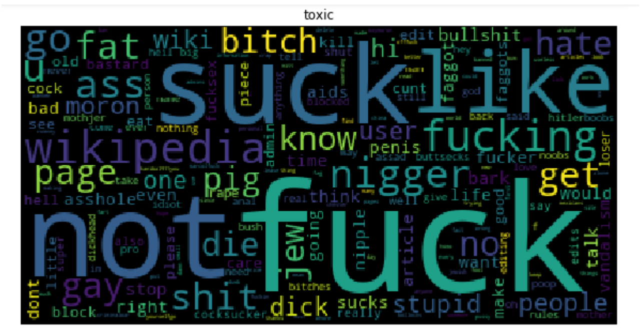


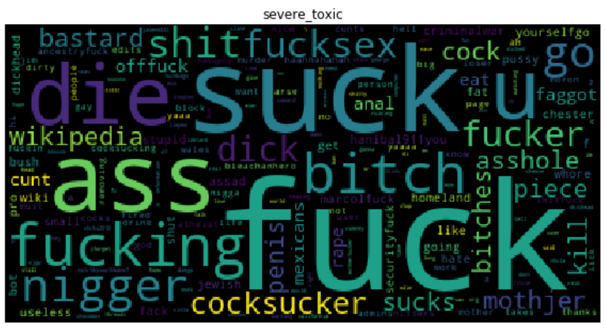
The below histogram describes the length of sentences. We can see that there are about 18000 sentences with 500 to 1000 characters. The majority of comments have length upto 500.

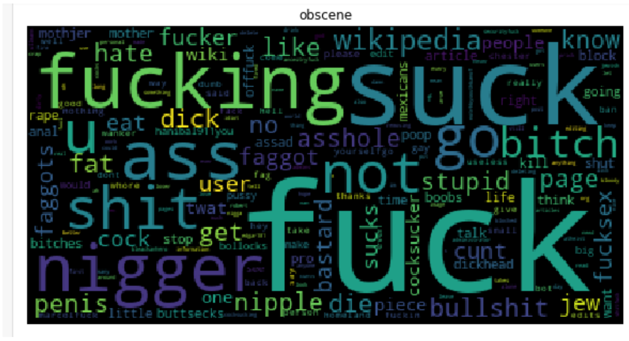


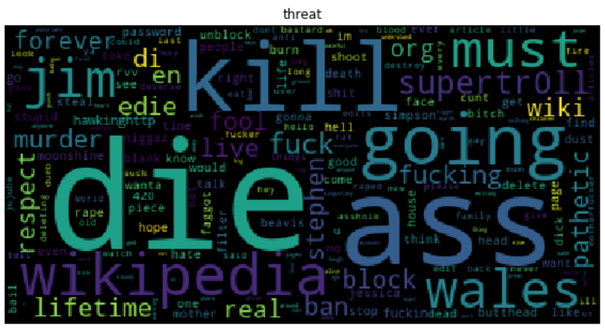
Word clouds of each label

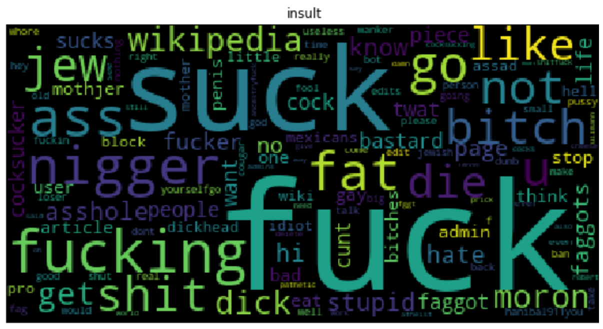
The following diagrams shows the most commonly used words for each label. The size of each word determines the frequency of that word in each label.

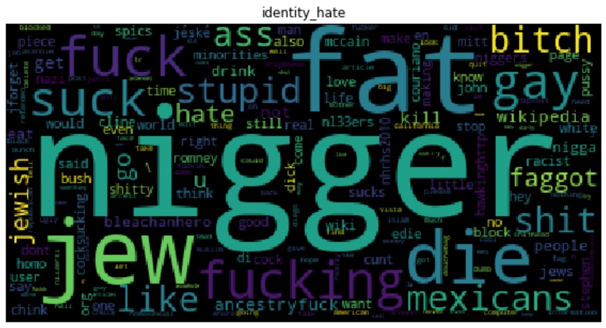












3 Data Preprocessing

3.1 Removing URLs

The first step in data cleaning is to remove URLs. URLs become hard to remove once symbols and characters are removed, so they should be filtered out first. We removed all words that start with 'http'.

3.2 Converting phonemes

Phonemes are letters with different sounds.For example, ä. Observing the dataset, we found that many comments had phonemes where alphabets can be used. So we replaced them with their corresponding alphabets.

3.3 Stop words

Stop words are the words that are frequently used in communication. Stop-words usually have little lexical content and they do not alter the general meaning of a sentence. Their presence in a text makes the model to fail in distinguishing it from other texts since stop words are used in all sentences. So, we used the built-in dictionary of stop words from Natural Language Toolkit and removed the word 'not' from it, since this word can impact the comment. After removing 'not' from the dictionary, we used it to remove all the other stop words from the text.

3.4 Removing symbols and punctuation

Since most of the embeddings don't provide vector values for punctuation and other special characters, we removed the symbols and punctuation from the text.

3.5 Removing contractions

Contractions are words that we write with an apostrophe. Examples of contractions are words like "I've"(I have) or "let's"(let us). We expanded these contractions to standardize our text.

3.6 Stemming and Lemmatization

Stemming is the process of converting derived words to their root form. This process chops of the end of words in the hope of achieving this goal correctly most of the time, and often include removal of derivational affixes.
Lemmatization usually refers to doing things properly with the use of a vocabulary and morphological analysis of words, normally aiming to remove endings only and to return the base or dictionary form of a word, which is known as the lemma.
We used both porter stemmer and WordNet lemmatizer. The comments had different forms of a word, such as organize, organizes and organizing. But they all represent the same word in our problem. So using Stemming/Lemmatization for those three words gives a single word, which helps algorithm learn better. We used a combination of Stemming and Lemmatization in our model.

3.7 Vectorization

Learning models cannot take text data directly as input. It should be transformed into some encoding scheme like traditional schemes used TF-IDF weights for each word existing in the corpus. Most simple TF-IDF weights are calculated by:

$$\textbf{TF.IDF} = TF * log(\frac{N}{DF})$$

Here TF is the term frequency, DF is the document frequency and N is the total number of documents in the corpus. The term frequency refers to the number of times a word has appeared in a single document whereas document frequency refers to the number of documents in which that particular word has appeared.

3.8 Adding new features

We tried adding new features to the dataset computed from the data. Those are:
1. Length of comments: Length might sometimes influence the data.
2. Number of symbols: Some comments had words like 'f**k', which might not reflect in training the model, since symbols are being removed.
The above features had a correlation of less than 0.05 with all labels. So, we decided that adding these features does not benefit the model.

4 Model

Since this is a classification problem, we tried using models like Logistic regression, Random Forest, SVM and XGBoost for predicting the labels. After doing the preprocessing and once the data set is ready, we calculated the accuracy of the corresponding models. The below table shows the accuracies of the models in public data set.

Model	**Accuracy**
Logistic Regression	0.9826
SVM	0.8893
Random Forest	0.7689
XGBoost	0.9353

As you can see Logistic Regression clearly has a greater accuracy, but we felt that its overfitting. So we tried tuning the hyperparameters. We increased inverse regularization parameter to 5 and changed the random state to 4. To get a better model, we used ensembling methods. We combined many models with different weights in search of a better model. In the end, we used Logistic Regression and Random Forest with weights 0.8 and 0.2 respectively. This model gave an accuracy of **0.98415** on the public dataset and **0.98506** on the private dataset.

5 Detailed Error Analysis

To find out the remaining problems we performed an extensive error analysis on the result of the ensemble. We found some problems.

5.1 Metaphors and Comparisons

Subtle metaphors and comparisons often require understanding of implications of language or additional world knowledge. So it is difficult to classify the metaphors into its labels.
For example, "Who are you a sock puppet for?", is an insult but does not contain any toxic words.

5.2 Sarcasm and irony

As sarcasm and irony detection is a hard task itself, it also increases difficulty of toxic comment classification, because the texts usually state the opposite of what is really meant.

5.3 Usage of swear words in false positives

Classifiers often learn that swear words are strong indicators for toxicity in comments. This can be problematic when non-toxic comments contain such terms.
For example, “Oh, I feel like such an idiot now.Sorry, bud”, does not contain any toxicity but it contains swear words. So the model might assume that it is toxic even though it is not.

6 Conclusion

In this project, we worked on multiple approaches for toxic comment classification. We used different pre-processing methods like stemming, lemmatization, removing stop words etc to prepare the data. Different ensembling methods were used to build our model.

7 Links to the data and pickle files

1.Train data:
https://drive.google.com/file/d/1a2YQupQMnjslkYoqQeDe1E4ThiZZVokg/view?usp=sharing

2.Test data:
https://drive.google.com/file/d/14SLVntk2ubwFwLvGhu$_4VO0T7px59yhw/view?usp = sharing$

3.Logistic Regression Model Object:
https://drive.google.com/file/d/1b0mHJCrQVCfxKMTiC0FGBS7GnpXrZUqJ/view?usp=sharing

4.Random Forest Model Object:
https://drive.google.com/file/d/1cQdTdwcBeYYoIUaqjErK6rwFXW$_fw711/view?usp = sharing$

8 References

1. https://towardsdatascience.com/attack-toxic-comments-kaggle-competition-using-fast-ai-b9eb61509e79

2. https://sijunhe.github.io/blog/2018/05/01/kaggle-toxic-comment

3. https://www.geeksforgeeks.org/generating-word-cloud-python