NLP Final project
Toxic Comments Classification

Kamil Sabbagh
Karam Shbeb

May 2022

1 Introduction To The Topic

It's difficult to talk about topics you care about online. Because of the possibility of online abuse and harassment, many people have stopped expressing themselves and stopped searching out diverse viewpoints. Many communities have limited or altogether shut off user comments as platforms fail to successfully facilitate dialogues.
NLP presents a solution to such problem. Using NLP models will allow the communities to add toxicity filters to hide or delete toxic comments which damage the experience of the users and prevent them from expressing themselves.
We will be working to create a tool to help improve online conversation. by focusing on the negative online behaviors, like toxic comments (i.e. comments that are rude, disrespectful or otherwise likely to make someone leave a discussion). we used methods and ideas explained in the course to build an NLP model which work as a classifier to toxic comments, as the tool to help this cause.

2 Review of existing approaches

The topic of "Toxic Comments Classification" has been presented on Kaggel Challenges, with many solutions and discussion. Therefore, it was our main source of findings to existing approaches to study and analyze.

From our study submissions on the contest we have seen different approaches by contestants on Kaggel. The contestants have used methods such as Logistic Regression, Transformers, SVM, LSTM and GRU. The best accuracy score reached was 98.856%, with an mean of 98.7961% for the top ten submissions

By examining previous solutions of highest scores we conducted a list of the key findings and similarities between them as follows:

- The best vectorization method is **TFIDF Vectorizer**
- The best (stand alone) model to make predictions is Logistic Regression The data is complete. i,e, it has no empty or null values
- the length of the sentence does not play a factor to decide if a comment is toxic or not.
- using the ID of a person who has wrote a toxic comment before as a parameter leads to over fitting on the training set. Therefore, should not be used as a factor to decide if a comment is toxic or not.
- Some of the most successful methods fail to classify toxic comments when punctuations are add. 41% of the testing data is unlabeled therefore unusable. So there for testing only the left 61% is usable.

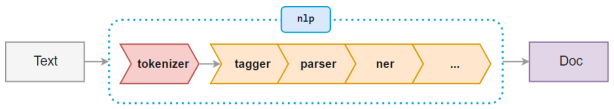


Figure 1: NLP Pipeline proposed by course notes

Using TFIDF Vectorizer does all steps of the NLP pipeline, and it was the most used, and most successful method of vectorization. Many of the highest scoring teams have used Logistic Regression as the model to make the prediction of the test data. We note that many contestants have used Logistic regression as a part of a larger model using boosting and ensemble learning. Nevertheless the highest ranked solution uses the Logistic Regression as a stand alone model. Most of the submission has found that training and testing data does not have any empty or null values. Some teams have considered the length of comments to see it can play a factor in deciding if a comment is toxic or not; they all agree that the length of the comment does not effect the toxicity of a comment. Another group of teams have considered if the user has been detected to wrote toxic comments before as a factor to decide of a new comment by him is toxic as well. Such approach led to over fitting on the training data and drop in the accuracy on the testing data. Finally teams have noticed that the testing data has some labels with the value -1, which means that it is not labeled. Therefore, such data is not usable and should not be considered while testing. This unlabeled data make about 41% of the whole data, which leave about 61% of the testing data to be used.

3 Your solution and experiments / demo

We will divide our solution to four sections: preprocessing, visualisation, modeling, and evaluation.

3.1 Preprocessing

To preprocess the data, we apply a ”cleaning” function to all the comments. The cleaning pipeline:

- Change all the letters in the comments to lower cases.
- Use regular expressions to remove unnecessary newlines, spaces, and usernames.
- tokenize the comments using nltk TweetTokenizer
- check tokens and implement apostrophe replacement (e.g: can’t –¿ can not)
- remove stop words

3.2 visualisation

In visualisation, we first looked at the representation of all the labels of toxicity in the comments.

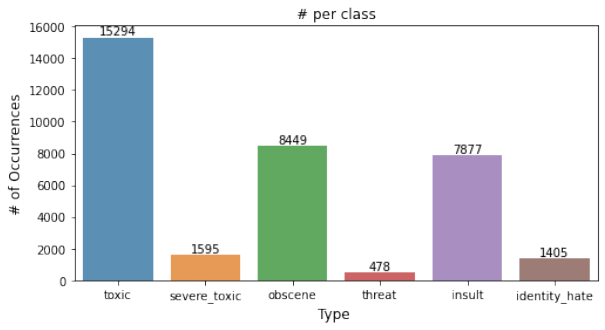


Figure 2: Distribution of the labeled comments

from the figure we can see the the most represented labels are: Toxic, obscene, and insult. In that order. while identity hate, severe toxic, and threat were the least represented.

Then we taken a look the number of toxicity labels appeared per comment. Plotting the results gives:

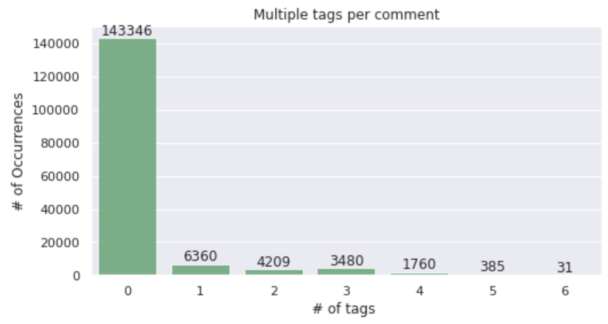
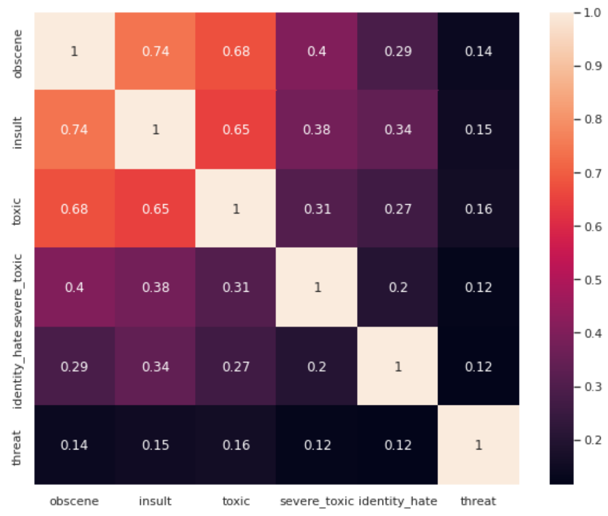


Figure 3: Distribution of the labeled comments

from this figure, we can see that the most of data are with zero toxic labels (i,e are clean comments). and there exists cases where a single comment has few has several or all of the the labels.

then we've taken a look at the correlation between the label, and plotted the heat map of this relations.



	obscene	insult	toxic	severe_toxic	identity_hate	threat
obscene	1	0.74	0.68	0.4	0.29	0.14
insult	0.74	1	0.65	0.38	0.34	0.15
toxic	0.68	0.65	1	0.31	0.27	0.16
severe_toxic	0.4	0.38	0.31	1	0.2	0.12
identity_hate	0.29	0.34	0.27	0.2	1	0.12
threat	0.14	0.15	0.16	0.12	0.12	1

Figure 4: Distribution of the labeled comments

From the plot we can see that some labels are higher correlated, e.g. insult-obscene has the highest at 0.74, followed by toxic-obscene and toxic-insult.

After that, we have taken a look at the length of the comment, and plotted the length of the comment with the frequency of such length.

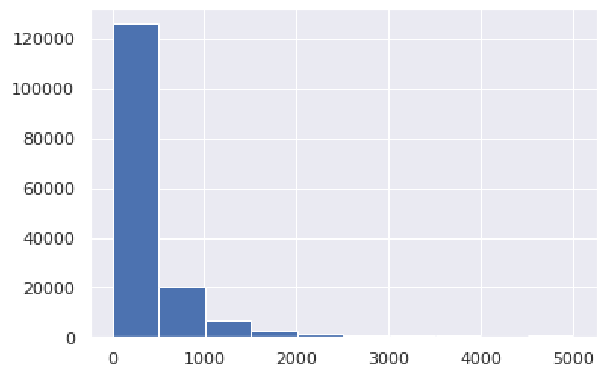


Figure 5: Distribution of the labeled comments

Most of the text length are within 500 characters, with some up to 5,000 characters long.

Finally we have plotted the most frequent words related to toxic labels. for example plotting the most frequent words in the obscene comments gives the following plot.

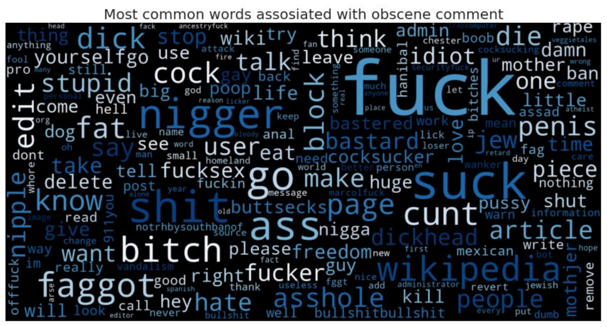


Figure 6: Distribution of the labeled comments

3.3 modeling

To build our model, the first and most import observation was that some comments have several labels. And this observation, makes the use of multi label classifiers absolute. Since some comments can have several labels, and due to the fact that multi label classifiers are designed to classify only one label per a set of feature (in our case the vector representing the comment). Therefore our task has transferred from multi label classification to several binary classifications, one classifier per label.

Therefore, we have several models and compered their results

3.3.1 Logistic Regression using Binary Relevance

This is probably the simplest which treats each label as a separate single classification problems. Nonetheless, in this method we are ignoring the correlation among the various labels.

3.3.2 Logistic Regression using Classifier chains

In this method, the first classifier is trained on the input X. Then the subsequent classifiers are trained on the input X and all previous classifiers' predictions in the chain. This method attempts to draw the signals from the correlation among preceding target variables.

3.3.3 Linear SVM using Binary Relevance

We build 6 classifiers, each of them to classify each label. It is similar to 3.3.1 but we use SVM instead of logistic regression.

4 evaluation

41.77% of our test samples don't have attached labels, which leave us with around 90 thousands comments to test our models on. Our evaluation metrics contain f1, recall and hamming loss. Recall score shows us how many True Positives the model has classified from the total number of positive values. The F1-score combines the precision and recall of a classifier into a single metric by taking their harmonic mean. we use f1-score mainly to compare models to each others and not only to evaluate our final model. Since we have several labels and our data is not balanced (i.e. we have more comments labelled as toxic than threat) we decided to use another metric to give a weight to each label, and here where hamming loss comes.

5 What have you learned

From this assignments, we have learned the power and robustness of NLP to achieve high accuracy on big data. Which is impossible to be done by humans.

We have exercised topic and methods discussed in the classes/labs such as tokenization, preprocessing, and vectorization. and used our accumulated knowledge to build an ML model to build upon these methods to achieve our goal of toxic comments classification.

We have experienced building a NLP pipeline and exploring the data in order to know what methods are better to use in terms of vectorizing, building the model and evaluating it. we have tried 2 vectorizing approaches (TF-IDF and word2vec), we tried 3 ML models and then picked the one who performed better in our task. We made observations on the comments that our model was unable to classify it correctly and made the following suggestion for future improvements to this classifier:

- Adding a spell checker before vectorizing would be helpful to classify comments when write misspell a word to fool the model.
- Since all comments in the training set were collected from Wikipedia, we recommend using another training set alongside to guarantee diversity.