Toxicity Classification

Zhihao Guo, Shangquan Sun, Yicheng Zhou
[1] School of Information, University of Michigan

Abstract

The purpose of this project is to detect toxicity of comments, which mainly extracted from online conversations, where toxicity of comments is defined as "anything rude, disrespectful or otherwise likely to make someone leave a discussion". We first embedded each comment into a vector and built different machine learning models such as LSTM, XGBoost to determine whether the comment is toxic and how toxic it is. We then compared the results and further incorporated some other information about the commenter that can be found in the dataset into the model, such as their race and gender. Our model has shown excellent classification power and we also built an user interface that takes input sentences from users and determines its toxicity. We also gave some representative words for toxicity and non-toxicity.

Introduction

Based on the data, the goal of the team is to build a model that recognizes toxicity and minimizes this type of unintended bias with respect to mentions of identities. Dataset labels will be used for identity mentions and to optimize a metric designed to measure unintended bias.

For each comment, the dataset also includes other features about this commenter besides the comment text, such as the commenter's race and gender. The output is called "toxicity", which is scored by human between 0 to 1. In order to classify whether a comment is toxic or not, we set up threshold of 0.5 to determine the toxicity.

Methodology and Model

Data Overview and Pre-processing:

There are 1.8 millions comments in the train sample, with 45 features. After we removed the rows with missing values, there were 235,087 data left. Among them, the ratio of toxic comments versus non-toxic comments is about 1:12, so the data is very imbalance. We examined the test data and found the same pattern too.

LinearSVM:

At beginning, a preliminary model of linearSVM has been trained. The word embedding method we used was tri-gram with each entry being the TF-IDF value of this word or words.The test accuracy is around 70%, which is acted as a baseline for further study.

XGBoost:

We started with the text pre-processing by converting all the words into lower case. We did not do anything further such as stemming or removing stop words because we thought it may be related to the toxicity of the comments.

Then we used tri-grams with entries being TF-IDF values to vectorize each comment. We chose the vocabulary size to be 8,000.

We incorporate the other features about commenters other than the comment text by assigning each sample different weights based on those features.

We further balanced the data by assigning true and false labels different weights according to their percentage in the whole data.

Simple LSTM:

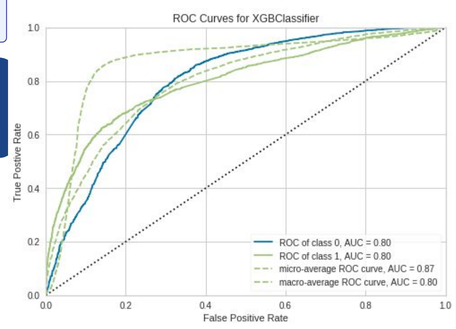
For LSTM, we did the same text pre-processing as XGBoost and embedded the text using LSTM embedding. We also chose a vocabulary size of 8,000 and assigned weights based on other features and label distributions.

Results

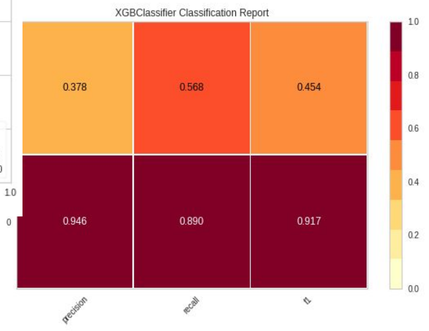
For XGBoost and LSTM, we evaluated the classification quality using Precision, Recall, F1-score, and we plotted ROC curve and calculated AUC values.

For XGBoost, the classification for non-toxic comment was very good, with high scores in both Precision and Recall. However, the classification for toxic comments has some issues, with very low precision, which means that the false positive rate is very high and our classifier predicts a lot of comments that are actually not toxic to be toxic. This issue come from the fact that our data is very imbalanced. Even after we reassign the weights, it is still affecting the prediction of our classifier. The ROC curves looks fine and the AUC is round 0.8. This shows that our XGBoost classifier has relatively good classification ability.

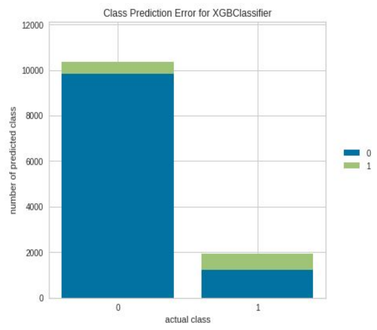
Some explanatory plots are as below:



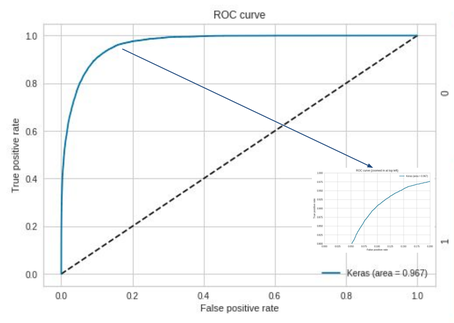
ROC Curve



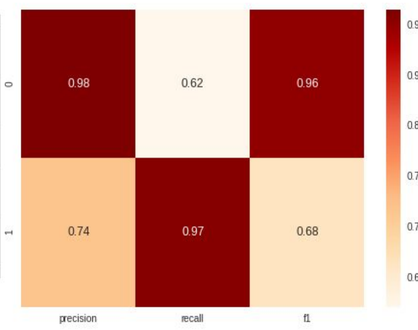
Precision, Recall Heatmap



For LSTM, the classification for non-toxic comment was very good, with high scores in both Precision and Recall. It only slides a little bit on the Recall of non-toxic comments, which means it is not capturing a lot of non-toxic comments. However, the LSTM classifier did a great job and shown great classification power, with ideal ROC curve and AUC being 0.97. Some plots are shown:



ROC curve



Precision, Recall Heatmap

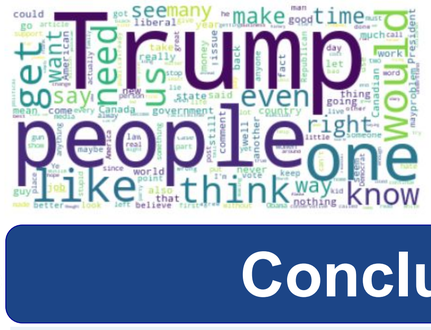
Word Cloud and UI

Other than the model, we did some other analysis on the data.

First, we built a small user interface that allows users input the sentence that they are interested in, and our algorithm will return the probability of this sentence being toxic. For example, the sentence "thank you" has 0.16% probability of being toxic, which means that it is not toxic at all, and that matches the semantic meaning of this sentence perfectly.

We also plotted the word cloud to show some representative words. We see that "Trump" is the most representative word for toxic comments, which is very interesting

Representative words for all toxic comments



Representative words for toxic comments only nouns.

Conclusions

After comparing different models, we found that LSTM method shown the best classification power. The AUC (Area under curve) for LSTM model is 0.967 and all its the Precision and Recall are above 50%. The result meant that we now build a model that can accurately classify whether a certain comment is toxic or not. In reality, such detection can help to maintain a stable internet environment.

Reference

1. https://www.kaggle.com/c/jigsaw-unintended-bias-in-toxicity-classificati, 2018

Contact: zhihguo@umich.edu, sunsean@umich.edu, yczhou@umich.edu