Kindness Project: Toxic Comment Classifier

Project Creator: Ting-Hsuan Lin Mentor: Dipanjan Sarkar

Introduction

Motivation

Specialized in children's literature, I have been interested in children's digital interactions. However, from surveys and personal experience, I notice negative behaviors are easily triggered when people interact with each other online emotionally. I hope to apply machine learning solutions to tackle the issue and encourage more positive behaviors online.

Problem

Current solution to mitigate online profanity and negative behaviors is through manual monitoring, which faces the challenge of high costs and low efficiency.

Business Use Case

Companies with messaging features and online communication platform can reduce the cost of manual monitoring with machine learning solutions to effectively detect and filter toxic messages.

Data

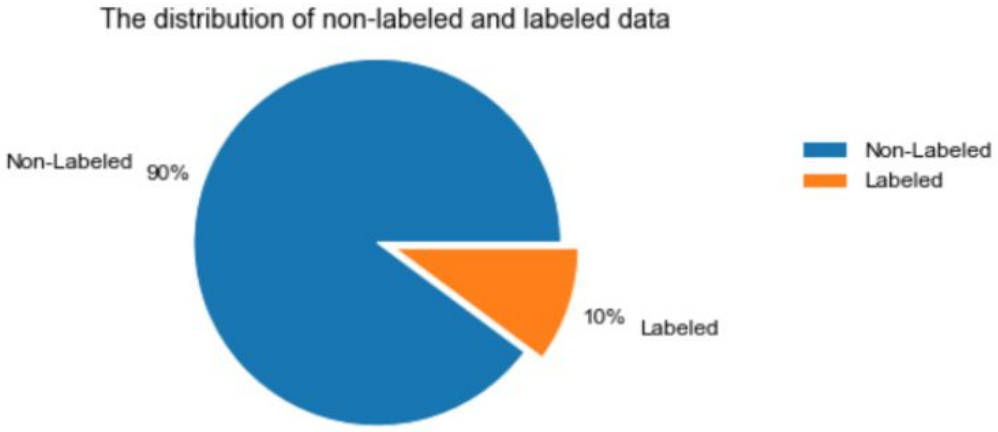
The training data covers 160K comments from English Wikipedia talk page edits, dating from 2004 to 2015. Each comment is manually labeled by 5K workers with one or more toxic types, including toxic, severe toxic, obscene, threat, insult, identity hate.

Data Preprocessing

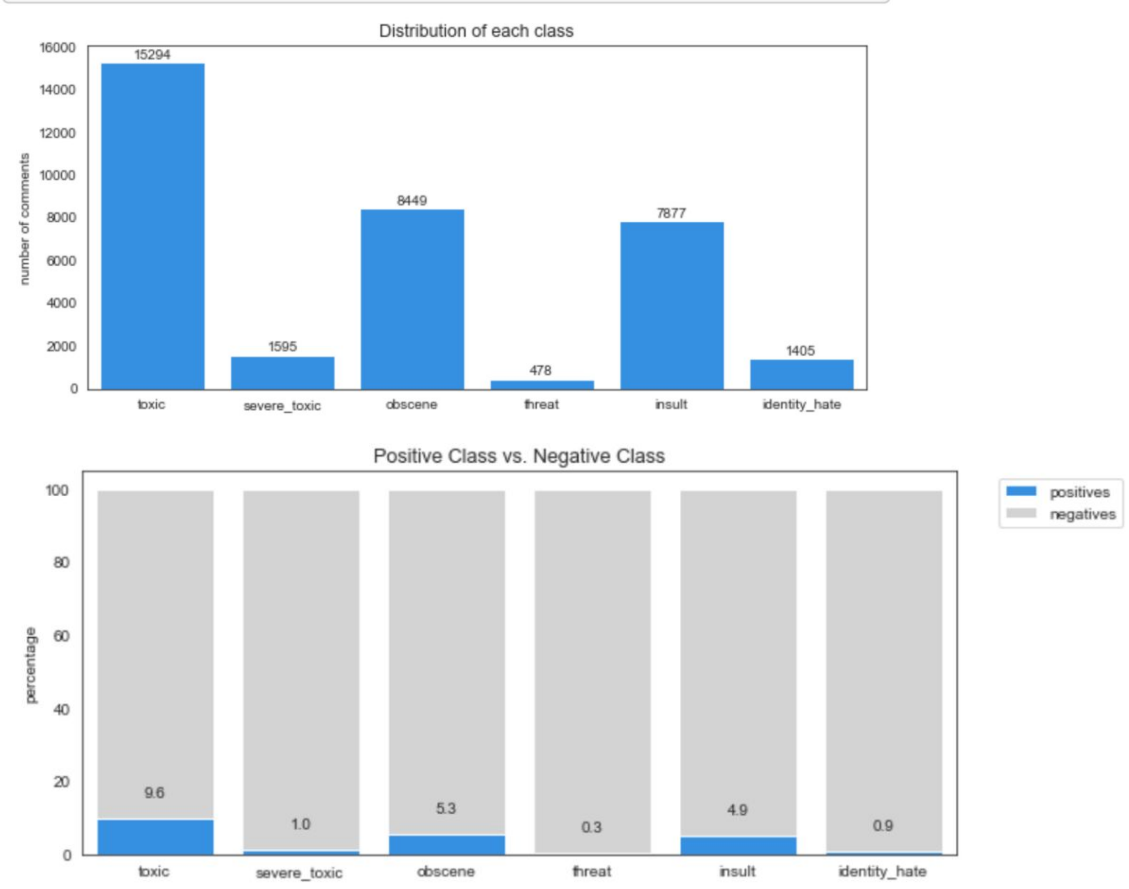
Each row represents a comment with a unique id and 6 binary labels: toxic, sever toxic, obscene, threat, insult, identity hate. 1 means positive and 0 means negative. Each comment can have multiple labels or no label at all, depending on whether it contains toxic messages. Since comment data is the independent variable used to predict whether it belongs to certain toxic type, I decide to take following steps to normalize the content: remove noise including HTML tags, special characters and stop words, expanding contractions and making the content form consistent.

Exploratory Data Analysis

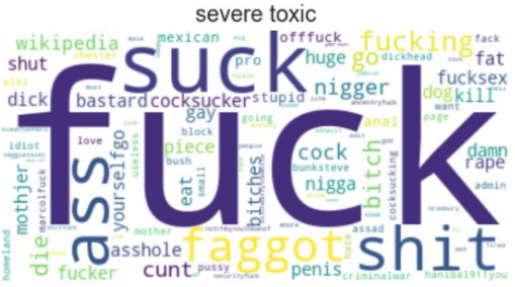
The dataset is imbalanced with 90% of data non-toxic and only 10% being marked with 6 types of profanity.

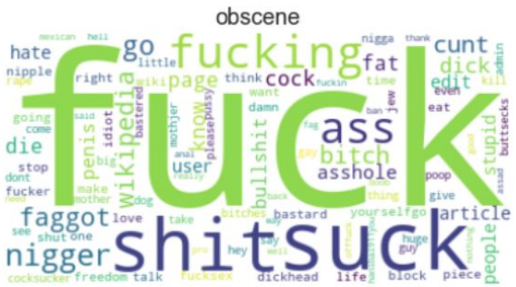


Within each type, the distribution of positive and negative data is first evaluated with exact number of data. Since each category has at least 30 positive data points, there are no under-sampling issue. When the distribution is examined from percentage perspectives, it is obvious that threat type suffers from the imbalanced issue the most and toxic the least. Based on the distribution, I assume that the prediction performance for threat might be worst and toxic might be the best.



Since it is a multi-class dataset, which means that, one comment might get more than one labels, keywords such as "fuck" and "suck" are commonly seen across different groups. Though the keywords seem to be similar among groups at first glance, "obscene" class is highly associated with sex-related words, "threat" is related to physical-abuse words, "identity hate" is associated with race and gender identity.





Feature Engineering

I apply three algorithms to extract content features, including Bag-Of-Word, TF-IDF and word2vec. Bag-Of-Word and TF-IDF both calculates how many times a word appears in a single documents and uses the occurrence frequency as feature value and ignores the order of words. TF-IDF, however, adds additional weights to words to reflect their relevant importance with the purpose to increase the weights of unique words, which appear less often but are more representative. Word2vec on the other hand captures the syntax relationship

between words by constructing a multi-dimensional vector per unique word. In order to use the method to extract word feature, we need to produce document-level embeddings. The process ensures that each input (numerical representation of a comment) fitting to the model shares same dimension.

Modeling

Metrics Selection

I consider the nature of this dataset and the business use case when choosing suitable metrics to evaluate model performance. First, the dataset is imbalanced with positive class containing much fewer data than negative class in every toxic type. The imbalance situation makes “accuracy”, the most frequent classification metric, unsuitable. Because accuracy measures the percentage of correct prediction out of total prediction, a model predicting majority group all the time ends with high accuracy score but is incapable of distinguishing between positive and negative class.

Second, it is inevitable for models to misclassify, but different types of errors come with different cost, and I want to optimize lower cost. Imagine two scenarios: When a model fails to detect a toxic comment and Wikipedia does not take any action, it is most likely that some users will report the message and the feedback data will be used in the next model training iteration. When the model misclassifies a neural comment as toxic and Wikipedia decides to take it down, the commenter is irritated and leaves Wikipedia community, which leads to the drop in user retention. The outcome of false alarm is more expensive than miss detection, so false negative error is more tolerable than false positive error. Metrics that show the trade-off between false positive and false negative are helpful.

I choose below four metrics based on the above criteria:

- ROC-AUC

ROC AUC score is the metric suggested by Kaggle competition. It measure the model’s ability to distinguish positive and negative class by evaluating the probability of getting right when predicting positive class and the probability of getting wrong when predicting negative class.

Additionally, ROC curve shows how a model performs under different threshold and AUC score provides an average score to tell a model’s overall performance.

- F1-Score/Precision/Recall

Precision and recall both focus on positive class and ignore true negative, which is the major class in toxic project. They are insensitive to the issue of data imbalance, and so does F1, the combination of precision and recall. Besides, I care more about false positive than false negative, models with higher precision and low recall are preferable when a tie happens.

In summary, I adopt Precision, Recall, F1 score and ROC AUC as the evaluation metrics.Models with higher F1 score and ROC AUC are considered better performer. In case of a tie, models with higher precision at the expense of slightly low recall are preferred.

Benchmark Model

I use dummy classifier provided by scikit-learn which predicts the most frequent class, in this case is negative class, all the time as the benchmark model. Its average ROC AUC score is 0.5, meaning the model has no ability to distinguish positive and negative class. Its precision and recall is 0.0 because it never predicts any positive class.

Model Selection

TF-IDF over Bag-Of-Word

I started by building logistic regression, the most widely used classifier, and tested it with TF-IDF and Bag-Of-Word (BoW) to decide which vectorizer to use with other classification models. TF-IDF vectorization performs better (0.97) than BoW model (0.95) at ROC AUC score, while TF-IDF gives slightly lower f1 score (0.45) than BoW (0.51). In terms of precision, TF-IDF performs much better. Combining these, I decide to apply TF-IDF as feature extraction method when building other models.

Classifiers without Tuning

I implement four commonly used classification models including logistic regression, multinomial naive bayes, decision tree and random forest with TF-IDF technique. The reason why SVM is not used is because the model takes large computational power and does not work well with high-dimensional features. I use cross validation to quickly get how each model performs and from there I select the best performer for hyper parameter tuning.

Logistic regression has the highest ROC AUC score (0.98) and fairly good F1 score (0.45). Its performance in general is better than tree-based model. So, it is likely that there is a linear relationship between independent variables and dependent variable. Considering that it is a simple model yet with good result, it is considered to be the top performer.

Random forest slightly outperforms decision tree in ROC AUC and precision. It is generally considered as a better model than decision tree in terms of prediction ability. The advantage of decision tree, being intuitive and easy to interpret, happens not to be what this project needs.

Naive bayes, though a standard model for text classification task, does not perform well in this case. It is likely that multinomial naive bayes cannot work well with continuous variable values generated with TF-IDF because it expects variables with discrete value as input. It also shows a very low F1-score because it fails to predict any positive case in highly imbalanced type (sever toxic, threat and identity hate).

Model Optimization

I focus on tuning the regularization parameters including “C” and “penalty” of logistic regression model in order to avoid overfitting. Because the number of words/features (170K) is larger than the number of data (160K), it is likely that logistic regression is overfitting.

Model Evaluation

The model without tuning achieves 0.69 at ROC AUC and 0.50 at F1 on training set (and 0.68 at ROC AUC and 0.48 at F1 on testing set). The model with C and penalty tuned via grid search with 3-fold cross validation turns out to (score 0.97 at ROC AUC on cross validation set) at 0.97, while score 0.68 at ROC and 0.48 at F1 on testing set.

The small difference between train and test performance results implies the model is not overfitting and can generalize good results when given unseen data.