Bigotry Detection

Michel Stahli

Springboard Capstone

September 23rd, 2022

Introduction

Discussion platforms, social networking and media platforms, and gaming platforms all face the challenge of comment filtering. Platforms have varying rules and goals for their discussion space specific to their culture and use. However, most platforms want to foster a comfortable space to increase user growth and usage. Avoiding clear prejudice against groups of people would follow as being a necessity. Even apart from moral and ethical aspects, allowing bigotry on a platform pushes away potential users and customers.

This project shows that using simple natural language processing (NLP) machine learning (ML) models is an effective way to detect bigoted language in comments. This bigotry detection model and the methods associated with it can be replicated with ease and used in a variety of business contexts. (See the Discussion section for examples). Though we focus on bigotry in our business case, other specific content and language concerns would have similar implementation.

Business Case

This project will use labeled Reddit data to create a detection model that flags comments that likely contain bigotry. The application is for businesses willing to build a reviewing team to go through comments flagged by the detection model, à la human-in-the-loop (HITL). This has the obvious drawback of requiring more resources in terms of human-worked-hours, but avoids the potential of filtering out comments that were never an issue in the first place. The project will assume that the business has the capacity to review up to 10% of comment traffic on their platform. The methodology would be identical for any other business capacity.

The Data

Two datasets were used for this project and combined for the final model.

Reddit Comment Dataset:
https://web.archive.org/web/20160304012220/http://idibon.com/toxicity-in-reddit-communities-a-journey-to-the-darkest-depths-of-the-interwebs/

The main dataset was scraped from Reddit and annotated using CrowdFlower by Idibon between February and March of 2015. Comments were scraped from subreddits that contained especially toxic (negative) or supportive (positive) content. Each row contained a unique comment, and included various labels concerning its sentiment in the following columns. Our target column, *bigotry*, was defined by the author, Ben Bell, as follows:

"[The] use of bigoted (racist/sexist/homophobic etc.) language, whether targeting any particular individual or more generally, which would make members of the referenced group feel highly uncomfortable"

A confidence column for bigotry measured the consensus of the annotators on whether the comment was an instance of bigotry. I chose a cut-off value above two thirds, assuming a reviewing team would choose a similar diplomatic threshold. This left us with a remaining 8814 rows of comments, out of which 515 contain bigotry.

Columns used:

bigotry - a label for whether or not the comment contains bigotry

text – the text of the Reddit comment

upvotes – number of times the comment was upvoted **note on real time nature**

Emotion Sensor Dataset:

I used a second dataset which included a list of 1004 unique words with scores on seven emotions; disgust, surprise, neutral, anger, sad, happy, and fear. I downloaded the dataset from Kaggle. Though the original page and dataset was taken down recently, it was re-uploaded to data.world here:

https://data.world/elie707/emotions-sensor-dataset

Exploratory Data Analysis

Combining the Datasets:

I combined the two datasets by adding the seven emotion columns to the Reddit Comment Dataset. The values under each emotion column is the average score for that emotion taken from all words in the comment. As an example, the comment:
"Yeah man, back then OS's didn't need viruses to crash, they just did it by themselves!" Contains three words from the *Emotion Sensor Dataset: "*yeah", "need", and "virus". The scores for each emotion would be the average score for those three words. For the detection model dataset a score of zero was assigned for each emotion in the case that no words in the comment were found in the *Emotion Sensor Dataset.*

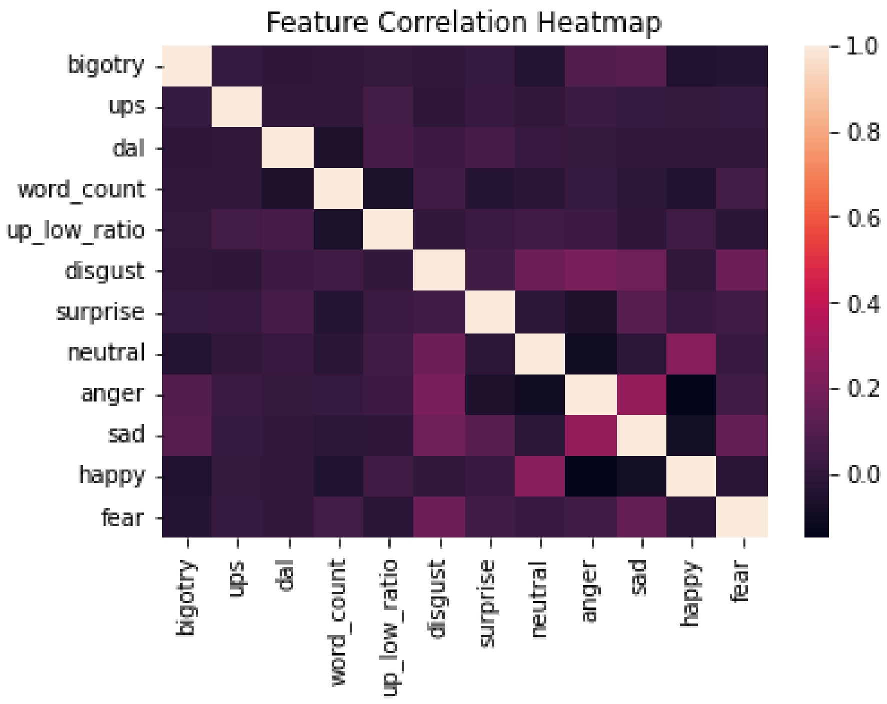


Figure 1. Heatmap of all features used in the final model apart from the actual text.

Luckily there are no noticeable correlations between the model feature variables and thus no major concern for multicollinearity. The highest correlations are between *anger* and *sad* (0.28), *happy* and *neutral* (0.24), and, unsurprisingly, between *anger* and *disgust* (0.20). The correlation between *anger* and *sad* is the only slightly worrisome case, since both features also have similar distributions even when accounting for *bigotry*. (See figure 3 below)

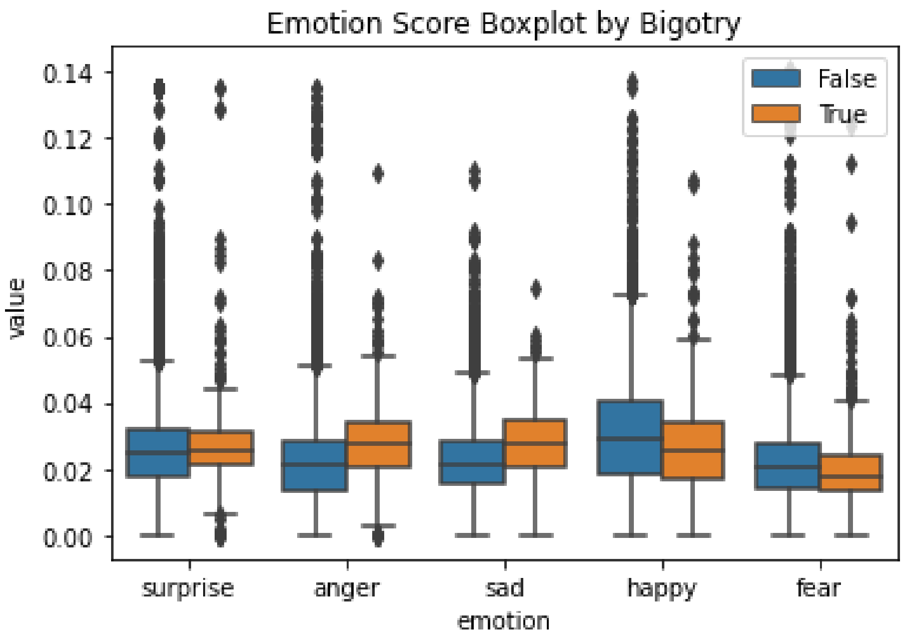


Figure 2. Columns 'neutral' and 'disgust' mostly contain values close to zero and thus were left out.

Taking a closer look at the values of the emotion scores, we can see there is quite a spread of outliers for all emotions. *Anger*, and *sad* look nearly identical for bigoted comments. It is important to note that comments without words in the *Emotion Sensor Dataset* tend to contain bigotry less. The difference in lower quartile ranges for each emotion lowers drastically once zero is imputed as a missing value. It makes sense that bigoted comments would primarily include stronger language (scores further from zero).

Running RandomForestClassifier on only the emotion features with *bigotry* as the target, we plot the feature importance below. As expected, *anger* is most important, with *fear* and *sad* following. We could have expected *disgust* to be higher than *surprise* or *sad,* though the similarity in distribution of *anger* and *sad* is a plausible explanation.

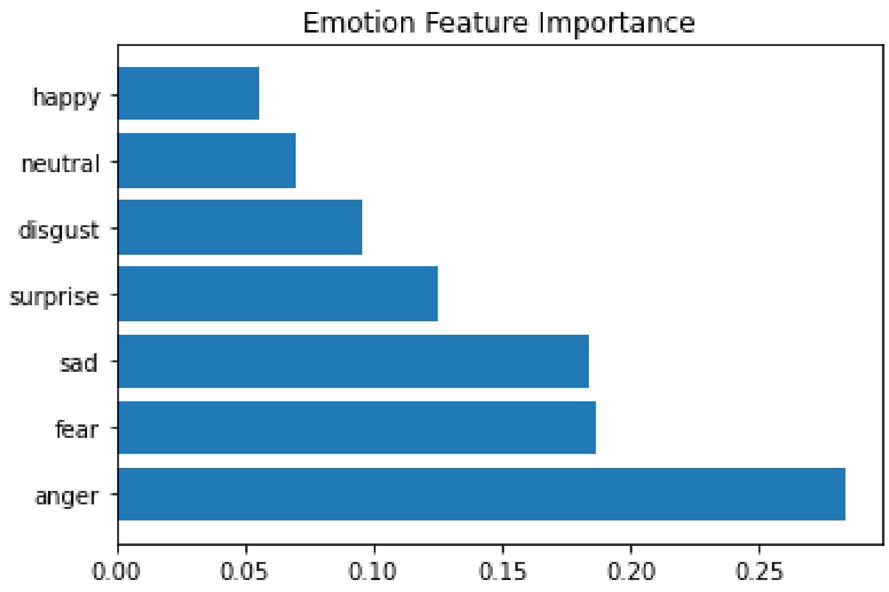


Figure 3. Random Forest classifier for bigotry on emotion features.

Word Feature Importance

After text cleaning, lemmatization, and count-vectorizing (min_df = 3) the comments, I found most predictive words using a Naïve Bayes Model on the text only. The fifteen most predictive words and probabilities are displayed below.

Table 1 Most Likely to Predict Bigotry

Rank	Word	Probability
1	faggot	0.8792
2	nigga	0.8675
3	nigger	0.8359
4	bigoted	0.7443
5	misogynistic	0.7443
6	aboriginal	0.7443
7	penis	0.7081
8	motherfucker	0.6599
9	gay	0.6359
10	cunt	0.6335
11	atheist	0.5927
12	slut	0.5927
13	motherfucking	0.5927
14	homophobic	0.5927
15	butthurt	0.5927

Table 2 Least Likely to Predict Bigotry

Rank	Word	Probability
1	gem	0.002
2	glass	0.0052
3	thank	0.0056
4	le	0.008
5	favorite	0.0084
6	nope	0.0087
7	keyboard	0.009
8	minute	0.0095
9	small	0.0096
10	sound	0.0106
11	before	0.0107
12	nice	0.0108
13	since	0.0112
14	used	0.0119
15	wish	0.012

Table 1 and 2 contain most and least predictive words for bigotry using Multinomial Bayes.

These words are mostly what we would expect. The most predictive words are especially explainable as many of them express bigotry directly in meaning or relate to groups of people known to face prejudice. As for “le” in Table 2, this seems to be a text cleaning error where the word “less” was modified to “le”.

The word “glass” seems to be less predictive because it tends to be used in the context of more technical conversations, like “keyboard”, and because “glasses” also counts for “glass” due to the text cleaning. Related discussions range from drinking a “glass of water”, prescription “glasses”, to zoo animals behind the “glass”.

The other more puzzling example is “nope” being so unlikely to predict bigotry. Most instances of “nope”, follow a similar pattern: “NOPE, skyrim is one of the best games iv ever played”; “Nope. That's a common misconception based a logical error. Zen is not the opposite of Zen…”.

Modeling

Preparation and Feature Selection

The goal is to make the detection model as general as possible, and useful in any stage of a platform’s growth. Consequently, features subreddit-id and comment author were dropped. To stay true to the HITL aspect of the model, any other labeled columns (such as, sentiment of the comment), were dropped as well. The feature *upvotes* was kept since metrics related to the speed and frequency of up-voting or similar platform mechanics could give insight even shortly after a comment is posted. Engineered features based solely on the text of a comment included a readability score (Dale-Chall), uppercase to lowercase ratio, and word count. The seven *Emotion Dataset* columns were kept as well.

In the interest of time, only one configuration of CountVectorizer hyperparameters was implemented. Apart from the count-vectorized text, the final model included only numerical variables, which were normalized. The resulting final training set included 302 word columns, and the eleven numerical columns discussed above for a total of 313 features.

Model Selection

Using GridSearchCV for hyperparameters, four models were compared and the best chosen based on ROC-AUC score.

Table 3

Model Name	**ROC-AUC Score**	**Best Hyperparameters**
Logistic regression	0.7774	C=0.16238, penalty='l1', solver='liblinear'
Random forest classifier	0.7734	criterion='entropy', max_depth=15, max_features=None, n_estimators=500
Gradient Boosting	0.7558	learning_rate=0.1, max_depth=6, n_estimators=100
Support vector classification	0.7322	C=1, gamma=0.1, kernel='rbf'

Table 3 shows model comparison using GridSearchCV four-fold cross-validation with 'roc_auc' scoring.

Logistic regression beat the Random forest classifier by 0.0040, and was chosen as the best classifier.

Thresholding

Using the default threshold of 0.5, our logistic regression model only identified 10.1% of bigoted comments. Continuing with our business case assumptions, we filter out thresholds that would cause our detection model to flag more than 10% of comments, per the given reviewing team capacity. Since we assume the reviewing team goes through all comments flagged by the detection model, we want to maximize recall, the percentage of bigoted comments that our model detects.

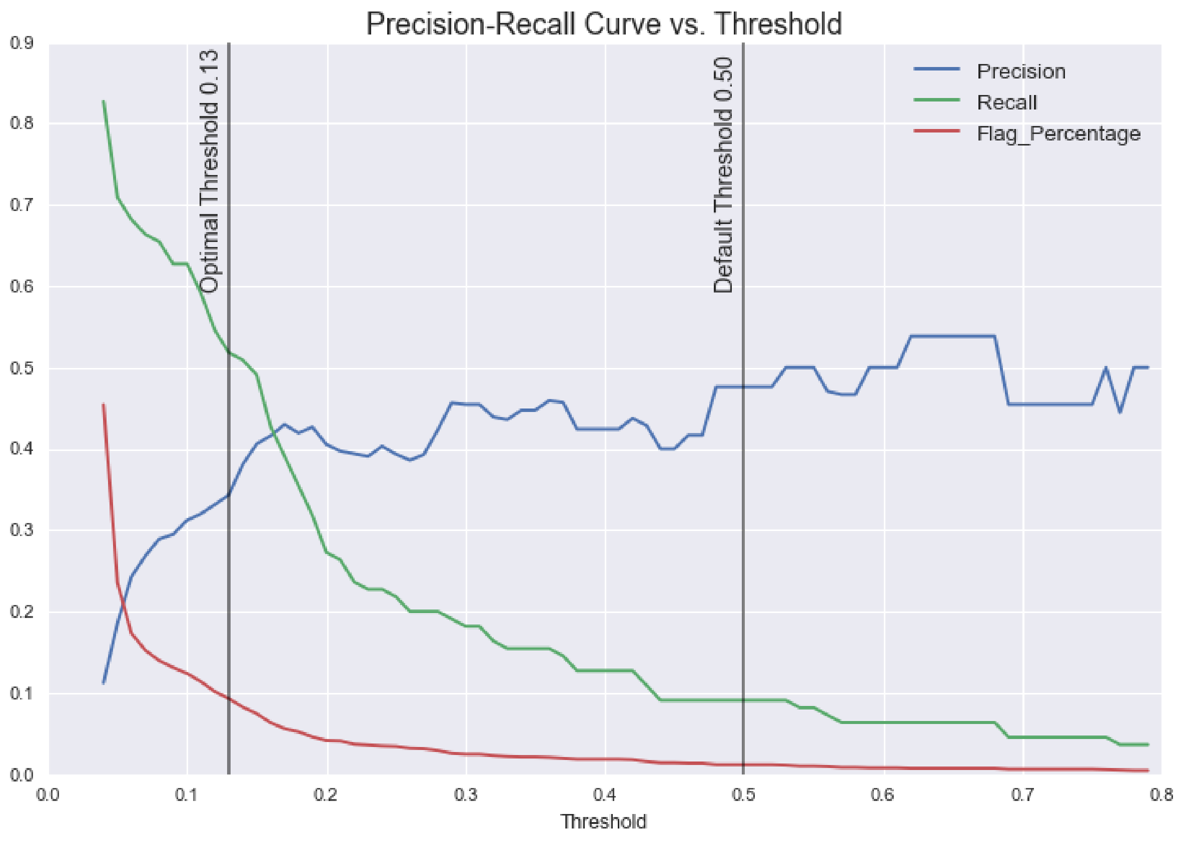
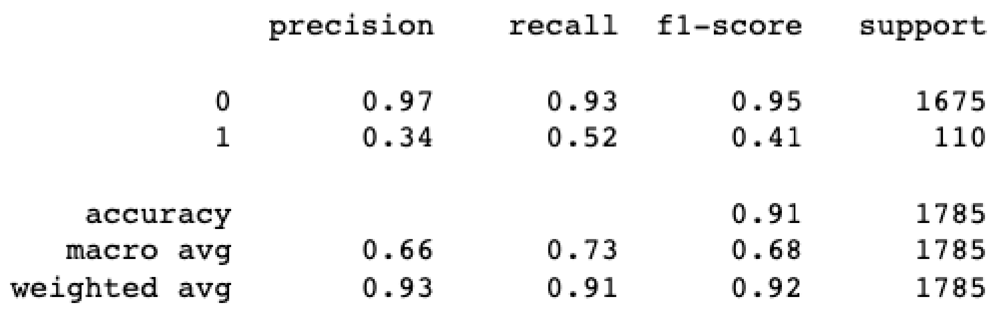


Figure 5 shows the optimal threshold value is just below a 10% flag percentage.

As evidenced above, the threshold 0.13 yields the highest recall score, at 51.8%. Precision would be more of a priority if we used a different HITL system as discussed in the conclusion. The final classification report with the optimal 0.13 threshold is pictured below.

Final Classification Report



```
              precision    recall  f1-score   support

           0       0.97      0.93      0.95      1675
           1       0.34      0.52      0.41       110

    accuracy                           0.91      1785
   macro avg       0.66      0.73      0.68      1785
weighted avg       0.93      0.91      0.92      1785
```

Figure 6: Best logistic regression classifier with 0.13 Threshold

Given the business case we are fairly satisfied, since our recall is high. The comparatively low precision (0.34) would be more of a concern in modified business cases as discussed below.

Discussion

Other Implementation Considerations

There are many ways to create more sophisticated mechanisms that minimize the need for human worked-hours, though most come at the cost of false positives. One way is to incorporate rule-based approaches. Many platforms have an automatic repercussion component for certain offenses, but with a built-in appeal system. As an example, if any especially damning terms (see Table 1 for inspiration) are found in a comment, the comment is automatically deleted and the poster faces a penalty. If the poster believes the rule does not apply to their post they can appeal and a platform reviewer would investigate the case.

If we used an appeal-only reviewing system as outlined above, our detection model, would still reduce bigoted comments by 51.8%. The reviewing team workload would depend primarily on how many flagged users appeal, though it's likely that not all users of true bigoted comments would do so. The downside is that our low precision means 6.6% of all users would be wrongfully flagged and need to appeal.

Another way is to limit which comments are reviewed based on their predicted probability. This can allocate reviewing resources to comments the detection model is most uncertain about. For instance, for comments that have a predicted probability between 0.1 - 0.3 all could be reviewed, but for comments with probability higher than 0.8 only 10% would be reviewed etc. The success of this would depend largely on the precision across thresholds, though the reviewing traffic should still drop. Looking at the PR-Threshold graph (fig. 5), our precision unfortunately does not change much even with higher thresholds, so the amount of false flags may not decrease as much as we would hope.

The other modification would be to only deal with comments that are reported by other users. In this case, the data would be significantly different and more balanced, and would most likely result in different ML model configurations. The main caveat of this method is that some comments may be missed, and potentially for undesirable reasons. For example, a bigoted subgroup culture could grow. Perhaps a combination of multiple models, one for non-reported comments based on a high threshold, and one for reported comments tuned to tease out more subtle differences could be used.

Conclusion

Using the bigotry detection model in our original business case, we can successfully identify over 51% of bigoted comments. Our rather extreme HITL expectation of zero false positives (wrongfully flagged as bigotry) post human review, means full implementation would reduce bigoted comments on the given platform by over 50%. Though even halving the reviewing capacity would decrease bigoted comments by noticeable margins (25%).

For future steps–creating a more complex HITL system with differing reviewing team allocations based on predicted probability thresholds would be ideal. Testing this model on milder data would also be useful, since our comments err on the extreme side. For potential technical improvements: trying more hyperparameter combinations, especially for the text vectorizer; incorporating the emotion scores differently; and aggregating ML models with text and numeric data separately may all raise our detection model performance..

As online correspondence seems to continue to increase, this project and its process will remain relevant for some time. More generally, this project shows that with even fairly humble means and simple models, natural language processing is advantageous and functional in solving business problems.