# Progress Report

Gabriel Britain

## Algorithm Description

In our project abstract, we mentioned that we would like to work with the following algorithms, as we believed they would be good starting places:

1. Naive Bayes
2. Support Vector Machines (discussed in Jonathan's paper)

While reading about suitable algorithms, I read about another classifier, Random Forest that performs well on imbalanced datasets, that I will discuss later on.

Each model (with the exception of SVC) is placed in a Scikit-Learn Pipeline, which makes building models significantly easier. From there, some of the hyperparameters of the model are tuned using an exhaustive search to achieve a maximum weighted F1 score.

### Naive Bayes

It should be noted that in order to perform multilabel classification with Naive Bayes, it's necessary to train one Naive Bayes model per label.

The foremost issue we encountered was the fact that Multinomial Naive Bayes does not perform well on imbalanced datasets. By the nature of its assumptions, Naive Bayes becomes biased towards one class.

While searching for a way to remedy this inherit bias, I found an excellent paper published by Rennie et. al, which discussed multiple different ways of counteracting the biases found in Multinomial Naive Bayes. It just so happened that the library we were using had an implementation of the model described by Rennie et. al, ComplementNB. I replaced the Multinomial Naive Bayes model that we had been using, and immediately our results improved drastically.

### Random Forest

After finding this model, it seemed as though the Random Forest classifier had significant promise; like ComplementNB, it is resistant to skewed datasets. Upon first setting up the Random Forest classifier, I found that using the same features as with ComplementNB enabled the classifier to achieve remarkable results relatively equal to the tuned ComplementNB model.

When I added hyperparameter tuning (and waited several hours for it to complete), the RandomForestClassifier achieved *remarkable* results on the never-before-seen testing set.

## Data Preparation

Because of the nature of our data, there were a lot of profane, obscene, racially-charged, and otherwise unpleasant words that almost-certainly determine some categories of toxicity in a comment. We chose to include counts for those particular words as a feature, as a comment that contains a lot of racial slurs or obscenities will almost certainly belong to the `'identity_hate'` and `'obscene'` categories.

In addition to these word counts, each comment was tokenized, and English stop words were removed. The term frequency/inverse document frequencies were calculated for the comment, and then the output, concatenated with the aforementioned "bad word counts", were fed to the model.

# Results

## Naive Bayes

The results achieved by the Naive Bayes model served as our baseline for evaluation. A hyperparameter-tuned Naive Bayes model yielded the following results on our testing set:

```
TESTING RESULTS:
Hamming loss (lower is better): 0.018717895691155242
Jaccard similarity (higher is better): 0.9440518640605122

              precision    recall  f1-score   support

       toxic       0.85      0.74      0.79     15294
severe_toxic       0.38      0.77      0.51      1595
     obscene       0.80      0.85      0.82      8449
      threat       0.59      0.13      0.21       478
      insult       0.71      0.74      0.72      7877
identity_hate      0.44      0.41      0.42      1405

   micro avg       0.75      0.74      0.74     35098
   macro avg       0.63      0.61      0.58     35098
weighted avg       0.77      0.74      0.75     35098
 samples avg       0.06      0.07      0.06     35098
```

## Random Forest Classifier

The results achieved by the Random Forest classifier (after hyperparameter tuning) were stellar, and by far the best results achieved of any of the models.

```
TESTING RESULTS:
Hamming loss (lower is better): 0.008330669942115631
Jaccard similarity (higher is better): 0.9736499739927681

              precision    recall  f1-score   support

       toxic       0.93      0.87      0.90     15294
severe_toxic       0.88      0.64      0.74      1595
     obscene       0.93      0.90      0.92      8449
      threat       0.96      0.61      0.75       478
      insult       0.88      0.84      0.86      7877
identity_hate      0.95      0.62      0.75      1405

   micro avg       0.92      0.85      0.88     35098
   macro avg       0.92      0.75      0.82     35098
weighted avg       0.92      0.85      0.88     35098
 samples avg       0.08      0.08      0.08     35098
```

# Analysis & Observations

We've made remarkable progress since the beginning of this project. One of the aspects that we hope to take a closer look at soon is including derived information about the texts as features, such as how long the comment is (in characters), the average word length, number of capital letters, etc. It seems as though information *about* the comment, not just the comment itself, is important to achieving strong results with our model.