# Automated Hate Speech and Offensive Language Detection

Team: Cobra

Tingyan Xiang(tx443), Nan Su(ns3783)

Tianyu Wang(tw1682), Yueqiu Sun(ys3202)

December 2017

# 1 Introduction

## Business Understanding

Many tweets are posted every day and the hate and offensive language occurs more than ever. It is usually very crucial to distinguish different types of abusive language. By classifying tweets, we can know people's attitude to certain news, celebrities and events in twitter very quickly. For twitter administrators, they can monitor and filter out tweets with extreme language more efficiently based on classifying. For instance, a user can report any tweet with abusive language. These tweets can be classified first before they go to the administrators, which improves the efficiency of the monitor process. For twitter users, there can be a new function to filter these abusive tweets which they may not want to see.

A key challenge for automatic hate-speech detection on social media is the separation of hate speech from other instances of offensive language. Lexical detection methods tend to have very low precision because they classify all messages containing particular terms as hate speech and previous work using supervised learning has failed to distinguish between the two categories. Many types of abusive language are often conflated.

To conclude, this is a classic multi-class classification data mining problem: We use a sample of these tweets with three categories: those containing hate speech, only offensive language, and those with neither. We transform tweet contents into features by using natural language processing, train multi-class classifiers to distinguish between these different categories and finally, evaluate models and choose one with optimal results.

## Data Understanding

We found the data in an open-source Github website. The data has also been used by some published papers which means it is reliable to some extent so fortunately, we do not need extra time to validate it.

The contributor collected tweets that contained terms in the Hatebase.org lexicon and labeled a sample of 25k of these into three categories to distinguish between hate speech(language that is used to express hatred towards a targeted group or is intended to be derogatory, to humiliate, or to insult members of the group), offensive language that does not meet this definition, and non-offensive language. Only tweets where two or more human coders agreed are used. The number of users who judged the tweet to be hate speech or offensive or neither offensive nor hate speech are given. The class label is defined for majority of users: 0 for hate speech, 1 for offensive language and 2 for neither.

The data are stored as a CSV and contains 5 columns:

| | count | hate_ | offensive_ | neither | class | tweet |
|---|---|---|---|---|---|---|
| 0 | 3 | 0 | 0 | 3 | 2 | !!! RT @mayasolovely: As a woman you shouldn't complain about cleaning up your house. & as a man you should a |
| 1 | 3 | 0 | 3 | 0 | 1 | !!!!! RT @mleew17: boy dats cold...tyga dwn bad for cuffin dat hoe in the 1st place!! |
| 2 | 3 | 0 | 3 | 0 | 1 | !!!!!!! RT @UrKindOfBrand Dawg!!!! RT @80sbaby4life: You ever fuck a bitch and she start to cry? You be confused as sh |
| 3 | 3 | 0 | 2 | 1 | 1 | !!!!!!!! RT @C_G_Anderson: @viva_based she look like a tranny |
| 4 | 6 | 0 | 6 | 0 | 1 | !!!!!!!!!!!! RT @ShenikaRoberts: The shit you hear about me might be true or it might be faker than the bitch who told it to |
| 5 | 3 | 1 | 2 | 0 | 1 | !!!!!!!!!!!!!!!!"@T_Madison_x: The shit just blows me..claim you so faithful and down for somebody but still fucking with ho |
| 6 | 3 | 0 | 3 | 0 | 1 | !!!!!!"@__BrighterDays: I can not just sit up and HATE on another bitch .. I got too much shit going on!" |
| 7 | 3 | 0 | 3 | 0 | 1 | !!!!“@selfiequeenbri: cause I'm tired of you big bitches coming for us skinny girls!!” |
| 8 | 3 | 0 | 3 | 0 | 1 | " & you might not get ya bitch back & thats that " |
| 9 | 3 | 1 | 2 | 0 | 1 | " @rhythmixx_ :hobbies include: fighting Mariam"<br>bitch |
| 10 | 3 | 0 | 3 | 0 | 1 | " Keeks is a bitch she curves everyone " lol I walked into a conversation like this. Smh |
| 11 | 3 | 0 | 3 | 0 | 1 | " Murda Gang bitch its Gang Land " |

Figure 1: sample data

count = number of CrowdFlower users who coded each tweet (min is 3, sometimes more users coded a tweet when judgments were determined to be unreliable by CF).

hate_speech = number of CF users who judged the tweet to be hate speech.

offensive_language = number of CF users who judged the tweet to be offensive.

neither = number of CF users who judged the tweet to be neither hate nor offensive.

class = class label for majority of CF users. 0 - hate speech 1 - offensive language 2 - neither

Taking a look at the distribution of the count column and our target variable, we can observe that most tweets are labeled by three people and the data is very imbalanced. As a matter of fact, there are 19190 tweets which are identified as offensive. The number decreases to 4163 and 1430 for neither and hate speech respectively. (See figure: class distribution)

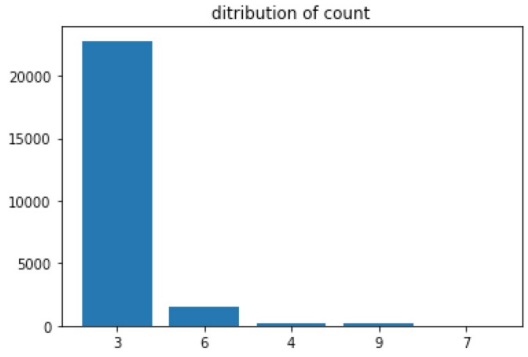


Figure 2: distribution of count

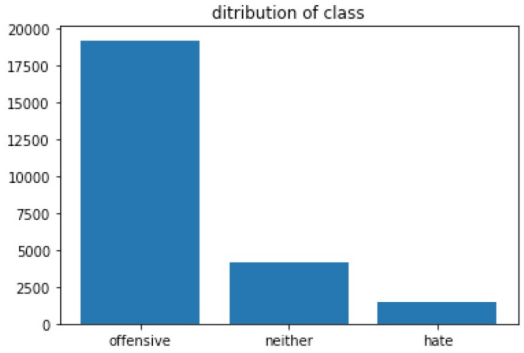


Figure 3: distribution of class

## 2 Data Preparation

Our target variable is a categorical variable and denotes which class each tweet belong to. And there are three classes: 0 denotes that a tweet contains hate language; 1 denotes that a tweet do not contain hate language but contain offensive language; 2 denotes that a tweet includes neither hate nor offensive language.

Machining learning models are not able to deal with text directly, so we need to transform text of each tweet into numerical or categorical features. Tweets are highly unstructured and not formal writing including emoij and abbreviation, so the first step is to clean text of tweets for tokenization. We converted text of each tweet into a list of words through tokenization and further cleaning, and then transformed the word list of each tweet into a feature vector based on the bag of words model.

In the process of cleaning data, we keep information of some objects and words, like counts of URLs, Mentions, Retweets, and the number of all words, even though these objects and words are removed in the process.

Steps of processing tweet data:

- Remove URLs and emoij using regular expressions
- Tokenize text based on space characters and convert words into lowercase
- Remove words beginning with Mention (@)
- Convert contractions to formal writing based on a dictionary we build, and then remove punctuations
- Remove words with length less than 2 and words that are not alphabetic
- Remove stop words. For the stop word list, add some common words in tweets, like 'rt' and 'ff', and remove negation words which may be important in bigram features.
- Stem and lemmatize words to convert inflectional forms and derivational forms of words to base forms.

In terms of extracting features, we can choose n-gram, like unigram and bigram, and different modes including TFIDF, count and binary. Besides, we can use parts of speech of each word in the same way. We also take advantage of some other features. First, there are some statistics of text including the number of words, characters, syllabus, URLs, hashtags and mentions; second, Flesch-Kincaid grade level and Flesch readability ease both are used to measure readability of text; third, sentiment analysis indicators based on VADER are used to evaluate how positive or negative a piece of text is.

Please note that we may not use all the features and different features can be used for different models.

# 3 Modeling and Evaluation

We used three different models to fit the data: Logistic Regression(LR), Random Forest(RF) and Support vector machine(SVM). Models may perform differently based on different data, and pros and cons are as follows:

LR:

Pros:

- Provides probabilities for outcomes
- Multi-collinearity is not really an issue and can be conquered by regularization to some extent
- Low variance

Cons:

- High bias
- Does not perform well when feature space is too large
- Sensitive to outliers

SVM:

Pros:

- Performs similarly to logistic regression when linear separation
- Performs well with non-linear boundary depending on the kernel used
- Handle high dimensional data well

Cons:

- Computational inefficient with large number of observations
- Can be tricky to find appropriate kernel sometimes

RF:

Pros:

- Reduces variance in comparison to regular decision trees
- Computational efficient on large data sets.
- Can handle a large amount of features without feature deletion.
- Generates an internal unbiased estimate of the generalization error as the forest building progresses.
- Has methods for balancing error in class.

Cons:

- Not as easy to visually interpret
- Does not reduce variance if the features are highly correlated

We use confusion matrix as our evaluation framework and default random forest as a baseline model.

## Random Forest

Starting with basic features: 1-gram binary words, 1-gram binary properties of words, term frequency–inverse document frequency, number of words and characters etc, without changing any default setting, after fitting random forest model, our confusion matrix is as follows (Figure 4):

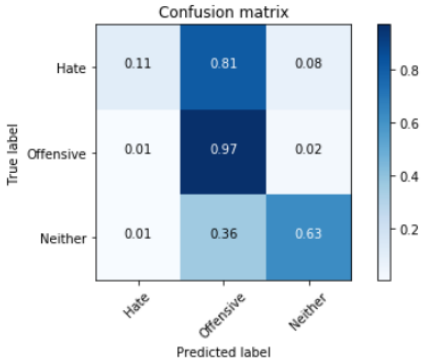


Figure 4: Basic features with default value

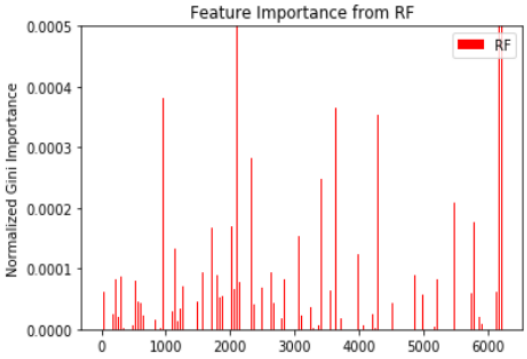


Figure 5: Feature Importance

It can be observed that the default random forest fitted well on offensive class, but had poor performance for the hate class. This model could not separate the hate speech from offensive speech accurately.
We thought of two possible reasons:

- The number of features was too large, which might make random forest models unstable to some degree.
- The data was imbalanced and the algorithm might be biased towards the majority class because our loss function did not take the data distribution into consideration.

For the first possible reason, we plotted a feature importance graph(Figure 5) and a corresponding confusion matrix(Figure 6):

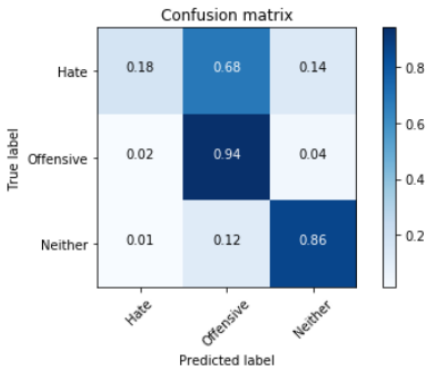


Figure 6: Basic model with feature selection

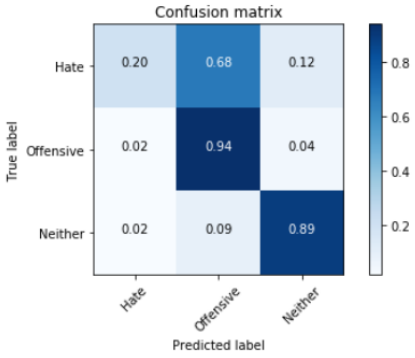


Figure 7: Model with feature selection and Parameter adjusting

After choosing features with feature importance bigger than 0.05, we adjusted the parameters by using grid-searching. When we set the number of trees to 70, the criterion to gini and set the number of maximal features in each tree to square root of total features, the model performs the best (Figure 7). Based on results above, we realized that feature selection is not the key point for model improvement. And we needed to move to the second possible reason.

There are several ways to deal with imbalanced data. Random forest models already took bagging of models into consideration, and we also took advantage of three other ways, over-sampling, under-sampling and combination.

The most common technique for oversampling a data set is known as SMOTE: firstly take a sample from the data set, and consider its k nearest neighbors (in feature space). To create a synthetic data point, take the vector between one of those k neighbors and the current data point. Multiply this vector by a random number x which lies between 0 and 1. Finally add this vector to the current data set to create new and synthetic data set. Then, after using SMOTE to our data set, we got Figure 8

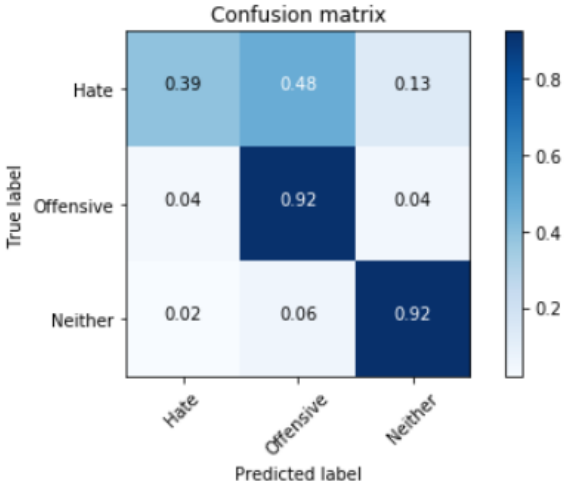


Figure 8: SMOTE

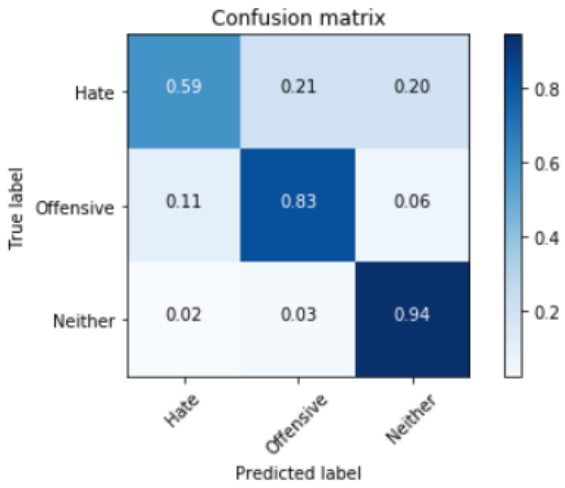


Figure 9: SMOTE+ENN

After using SMOTE, there was a significant improvement in our model. It could be inferred that the poor fitness might be a result of data imbalance. We then tried combination of over- and under-sampling as well, which provided a even better result. (Figure 9)

Although the prediction of hate speech was still not perfect, it had been greatly improved after a series of adjustments.

## Logistic Regression

After finishing random forest, we wondered if we could further improve the prediction by adding assumptions. So next, we assumed that the hate speech detective model is linear. Thus, we wanted to fit the model with Logistic Regression.

To apply Logistic Regression model on a multi-class classification problem, there are various parameters we can adjust to find the optimal candidate. For example, sk-learn package gives an argument "multi_class" which can be set as "multinomial" or "ovr". If the option chosen is "multinomial", the loss minimized is the multinomial loss fit across the entire probability distribution. If the option chosen is "ovr", the model fits one classifier per class. For each classifier, the class is fitted against all the other classes. In addition to its computational efficiency (only n_classes classifiers are needed), one advantage of this approach is its interpretability. "ovr" is the most commonly used strategy for multi class classification and is a fair default choice. After trying both options, we decided to use "ovr", which perform slightly better and much faster.

Things can be tricky for the process of choosing features and tuning parameters because there are tons of options to make. The procedure should be like: first we need to decide what exact features we would like to use, like n-gram, should they be binary, either "tf" or "tfidf" should be chosen, the lower bound of the word length etc. Notice that we must choose these features not only in term of words but also in terms of the property of words. Then, ideally, for each combination of these parameters, we also need to do grid searching to find the best C for the Logit model, which is highly time consuming. Therefore we attempted to adjust at most two options including the feature options and model parameters, and leave some of the options with default value instead of making adjustments to all options at the same time. When one option gives better performances than the other in most cases, we will not change it when we continue to adjust other options.

Therefore, at first we tried both "tf" and "tfidf", with binary option set to one and zero, respectively. The combination of "tf" and "binary" provided better result generally. For the first model, we used basic features including binary 1-gram words and binary 1-gram properties of words. After grid search, the parameter C =1 produced the best model with log-loss value of 0.37730. However due to the large amount of sparse features, the algorithms we used to solve the LR problem failed to converge.(Figure 10) Then we attempted to use Logistic Regression with L1 norm to reduce the number of features so that this issue could be solved.

Setting c(Here the c was the parameter for feature selection) to 0.01 and 0.1 helped shrink the feature size to hundreds and surprisingly, the performance became better. So we applied grid searching on a sequence of c and the parameter C = 10 and c = 0.3 produced the best model with log-loss value of -0.36185. With the chosen features and model, we can plot confusion matrix on test set as following:(Figure 11)

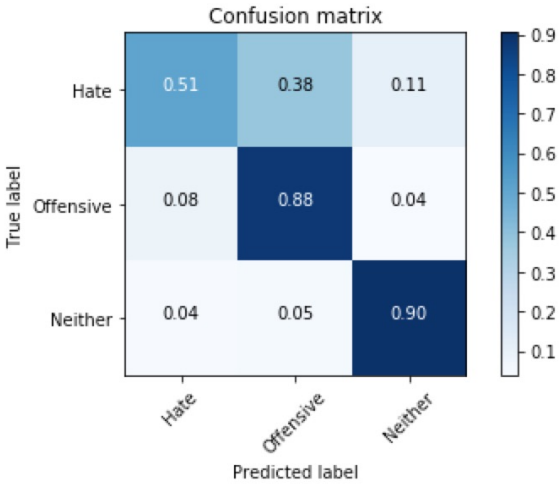


Figure 10: basic features without feature selection (fail to converge)

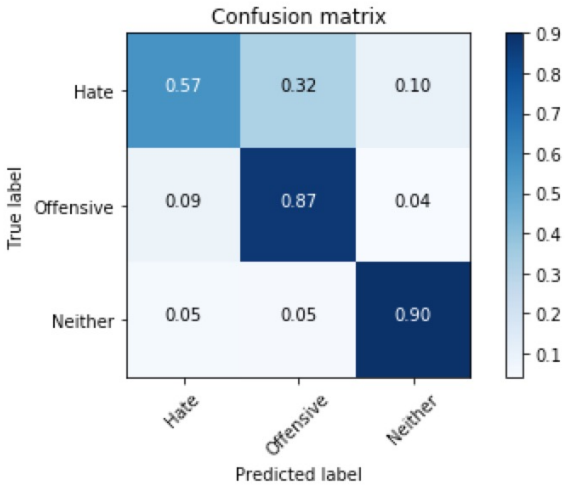


Figure 11: basic features with feature selection

In case that the 1-gram features were not enough, we then wanted to add more features. Considering the time cost of changing feature options one by one, we determined to use all the features we have: 3-gram for both words and property of words. Now that all features were used, we also applied L1 norm as a method of feature selection to, not only prevent the algorithm not converging again, but also improve the efficiency significantly. After grid searching C for the model and adjusting the parameter c of feature selection. We had the best log-loss value of -0.35131 with C=10 and c=0.25. The number of features decreased to nearly one thousand. With chosen features and model, we can plot confusion matrix on test set as follows:(Figure 12)

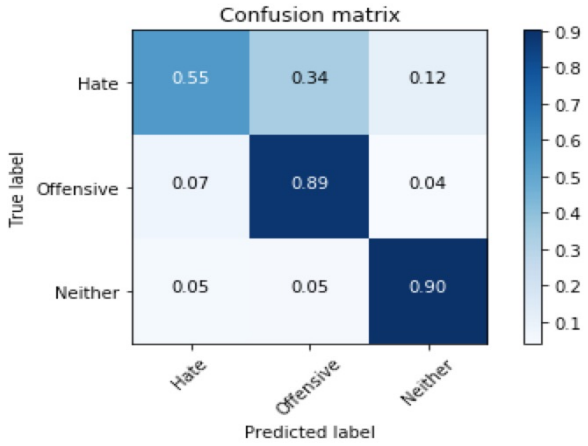


Figure 12: all features with feature selection

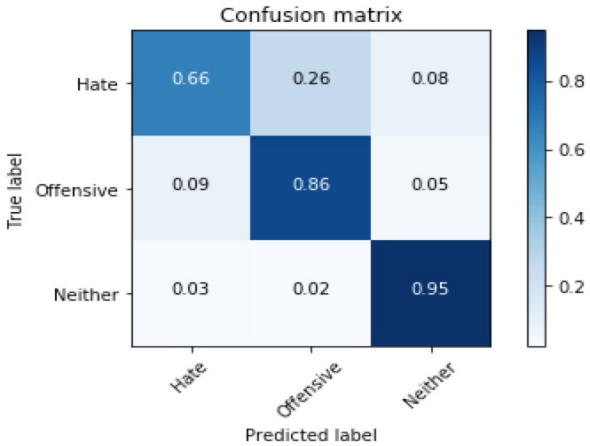


Figure 13: fine tuning

It can be seen from the figure that adding more features did not improve the performance of our model. As a matter of fact, we tried different kinds of combinations of features. After finding the best in-sample

model by parameter adjustments for both feature selection and model selection, the results appeared to be very similar. In general, feature selection by utilizing Logistic Regression with L1 norm was very helpful boosting the performance.

Finally we made use of fine tuning and set class weight option to 'balanced'. This means that the mis-classification cost for the minority class increased. The performance became better with approximately three hundred features left. The confusion matrix is shown above.(Figure 13)

## Support Vector Machine

Being aware of the high dimension of the feature space, we used Support Vector Machines to build a multi-class classifier to distinguish between tweets with hate languages and offensive languages and neither of the two. SVM also gives us the flexibility to choose different kernels. However, it is crucial to tune the regulation terms to optimize the model.

We first used a Logistic Regression with L1 regulation to reduce the dimension of the data. We chose from C=0.01 and C=0.1, and C=0.01 turned out to optimize the model results. Using L1 regulation with C=0.01 dramatically reduced 7131 features to 37 features.

It is worth noting that tuning class weight is crucial in building SVM in this case. The data was un-balanced, that is to say, the number of tweets with offensive language outweighed the other two categories, especially twitters with hate language. When the class weight was set to be default(none), SVM tended to classify almost all the tweets to the offensive language categories as it minimize the loss function, which was the last thing we want.

Setting the class weight to be balanced increased the price to mis-classify twitters with hate language, and increased the percentage of true positive for it.

The baseline SVM model with rbf kernal,L2 regulation, C=10, balanced class weight and unigram and tfidf features produced the confusion matrix shown as figure 14:

We implemented grid search to optimize the following hyper-parameter: kernel(rbf or sigmoid or linear or poly), regulation type(L1 or L2), C, and class weight(balanced or none). And we used iteration to select the features that could optimize the model performance from the features set(n-grams: 1 gram, 2 gram, modes: binary, count, frequency, tfidf) We tested each model using 5-fold cross-validation, holding out 10% of the samples for evaluation to prevent over-fitting. After using grid search to iterating hyper-parameters and different sets of features, we found that the model with linear kernel, L2 regulation, C=1, balanced class weight and uni-gram and binary features performed better than the models with other combinations of hyper-parameters and features.

We trained the best performing SVM model using the entire training data and used it to predict the labels for the test data. And we obtained the following confusion matrix shown in figure 15,

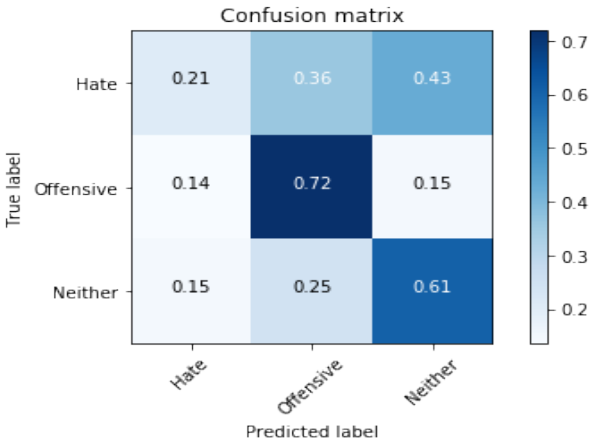


Figure 14: Baseline SVM model

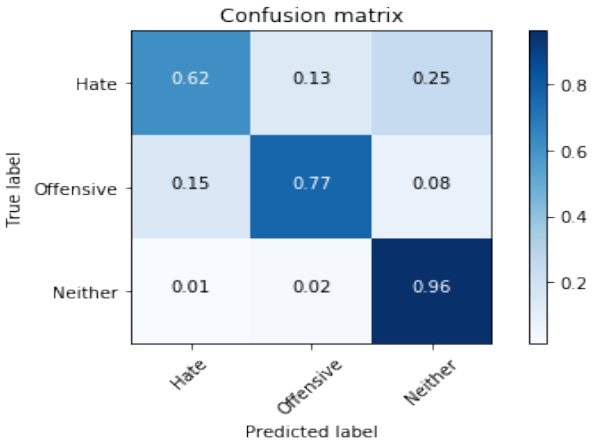


Figure 15: Best performing SVM model

The recall score was 0.96 for the Neither class, 0.77 for the Offensive class, and 0.62 for the Hate class. We can see that most of the mis-classifications occur in the right top grid of the matrix, suggesting that the model performed relatively poorly in distinguish Hate class and Neither Class.

# 4 Conclusion

## Deployment

In this part, we apply our classification model on a new data set from Twitter. We draw thousands of tweets through Twitter API and filter out some duplicates generated by re-tweeting and copying. We obtain 3727 tweets based on key words, like "fags" since these tweets are likely to contain hate speech or offensive language. After classifying these tweets with Logistic Regression classifier, we get 605 records labeled as hate speech and 302 records labeled as offensive speech.

First we take a look at the tweets identified with high probability of hate class, of which most are actually hate speech, like "Nigga you'll die gay" and "Gay niggas not allowed to make snow angels, god dont even fwu foh"; some are offensive speech, like "You don't want me to say fag but you making racist comment about my dick size is okay? lol why are some gay people the worst?"; few are neither, like "RT @ConradSW19: Gay people didn't kill millions of Jews and start a world war. Ffs. You choose to hate you don't choose to love. THIS IS NO...".

In general, those tweets which are labeled as offensive class are mostly offensive speech. And the error for neither class is negligible, as well. In terms of hate speech class, there are still some offensive tweets and

tweets that are neither hate or offensive, but more than 60% true hate speech is included. And based on this classification, we can find out most hate speech more efficiently.

As is shown above, twitter can take advantage of our data mining approach to monitor tweets with hate speech more efficiently. For example, Twitter receives tons of tweets reported by users which are potentially hate or offensive speech to certain people or group. Twitter can classify these reported tweets first based on our classifier and then focus on the tweets which are classified as hate speech. This classification model actually narrows down the monitoring range of most hate speech in tweets for administrator.

## Improvement

**More data** When applying our model on a real industrial scenario, it is hard to evaluate how this model performs since real labels are unknown all the time. Twitter can invite some users to label tweets by simply clicking "hate" or "offensive" when looking through their twitter page daily. Based on labeled data, Twitter is able to evaluate and thus improve the model. When it comes to application, the model should be updated periodically with tons of new tweets being generated because they can lead to new features, like new prejudiced terms. As a result, the model can always change.

**Evaluating model based on objectives** A model can be evaluated based on different objectives. A single model cannot always achieve its best performance no matter what the business objective is. When a new different business scenario occur, the firm should change the evaluation metric with it. For example, if Twitter wants to add a filter function for the users. An evaluation metric which weights precision more than recall might be more useful since the user experience will be severely undermined if the wrong twitter is filtered.

## Ethical consideration

There is no absolute criterion to judge whether a tweet is hate speech, offensive speech or neither and labels depend on certain people's opinion. Ethical problems may exist, for example, if Twitter forces to filter all the tweets that they think may include hate speech or offensive language and leaves the rest to the users, freedom of speech can be an issue. Or when an African American posted a tweet which contains the word "nigger", it is difficult to determine whether this is a hate speech.

# References

https://www.analyticsvidhya.com/blog/2014/11/text-data-cleaning-steps-python

http://www.vikparuchuri.com/blog/natural-language-processing-tutorial

https://machinelearningmastery.com/deep-learning-bag-of-words-model-sentiment-analysis

http://scikit-learn.org/stable/tutorial/text_analytics/working_with_text_data.html

https://medium.com/towards-data-science/machine-learning-nlp-text-classification-using-scikit-learn-python-and-nltk-c52b92a7c73a

http://t-redactyl.io/blog/2017/04/using-vader-to-handle-sentiment-analysis-with-social-media-text.html

Davidson, Thomas, Dana Warmsley, Michael Macy, and Ingmar Weber. "Automated Hate Speech Detection and the Problem of Offensive Language." arXiv preprint arXiv:1703.04009 (2017).

## Contributions

Tingyan Xiang(tx443): Data Preparation; Feature Selection; Conclusion.

Nan Su(ns3783): Introduction; Modeling and Evaluation LR Part. Conclusion.

Tianyu Wang(tw1682): RF and the rest Part of Modeling and Evaluation.

Yueqiu Sun(ys3202): Data Preparation; Modeling and Evaluation SVM Part