Hate Speech Identification

Data Mining Final Project

Jinmei Zheng
University of Pittsburgh
jiz155@pitt.edu

Olivia Keirn
University of Pittsburgh
odk1@pitt.edu

ABSTRACT

Detecting hate speech and offensive language in social media is becoming more and more relevant as time passes. While most research does not distinguish between hate speech and offensive language, we aim to do just that. Previous work from Davidson et al. [2] tried to do the same, but their model did a poor job classifying hate speech tweets — most likely being mistaken as offensive language.

We used the same data from Davidson et al. [2], which consists of just under 25,000 tweets labeled as hate speech, offensive language, and neither hate speech nor offensive language. For each tweet, we did text mining such as deleting tweet information, removed stop words, and did stemming on the remaining words. Along with creating a term-document matrix to use as features, we also added features — such as if the tweet has a negating term (no/not) and the ratio of 'bad words' to the total number of words in each tweet.

We then created a logistic regression baseline model using our term-document matrix — choosing to only use the top 1,000 terms as features and then have the `class` variable be our response variable. Next, we use a logistic model again, but tune the parameters in order to get better results. We also try a Linear SVM model and two Neural Network models — CNN and RNN. Our RNN neural network ends up outperforming the benchmark results from Davidson et al. [2], giving us an even better way to classify tweets as hate speech, offensive language, or neither of those two.

KEYWORDS

Hate speech, text mining, feature selection, stemming, classification, prediction

ACM Reference Format:
Jinmei Zheng and Olivia Keirn . 2018. Hate Speech Identification: Data Mining Final Project. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

For our final project, we will be determining whether or not a tweet contains hate speech. We chose this problem because as social media grows, so do the number of posts. While these platforms are meant to be a place for sharing thoughts, ideas, and life, not all posts reflect this and contain offensive messages. The ability to automatically determine a post with hate speech can be useful so it can be reported and dealt with accordingly by the platform. We will be using the `hate speech and offensive language` dataset from data.world to solve this problem.

2 RELATED WORK

Hate speech itself is vague, hard to be distinguished from offensive language and has no formal definition. However, a common consensus around the definition of hate speech exists to some extent, suggested by Jacobs and Potter [4] and Walker [10]. Recently, Davidson et. al. [2] propose a more precise definition of hate speech: "hate speech is language that is used to expresses hatred towards a targeted group or is intended to be derogatory, to humiliate, or to insult the members of the group."

Most of related works classify hate speech without distinguishing offensive language [1, 8, 12]. Kwok and Wang [5] propose to use bag-of-words to extract features from the tweets. However, bag-of-words features are extremely sensitive to offensive words, which easily results in a high recall but low precision. In other words, bag-of-words are not good at distinguishing the offensive language and hate speech. To solve the problem, syntactic features are introduced and have been demonstrated a better identification [3, 8, 11].

The most related work to our project is Davidson et al. [2], where they explicitly define the difference from hate speech to offensive language and treat this difference as the major challenges. Davidson et al. collect a dataset of tweets and use crowdsourcing to label tweets into three classes: hate speech, only offensive language, and others. Serving as a benchmark, they carefully design features including both bag-of-words and syntactic features and propose to use logistic regression with l2 regularization as classifiers. However, 40% of hate speech are still not detected in their work. In our project, we will build upon it, explore the techniques they mentioned, try more recent models, and potentially outperforms it.

3 DATASET

3.1 Data Collection

We used the same data that Davidson et al. [2] collected and used for their research. They used the Twitter API to find tweets that contained words and phrases considered to be hate speech — which were collected by `Hatebase.org`. Of these tweets, 25,000 were randomly sampled to be used for classification. Crowdsourcing, through CrowdFlower, was used to create the ground truth classification for each tweet. At least three people voted whether a tweet contained hate speech, offensive language, or neither. Finally, the class was determined by what the majority of CrowdFlower users perceived the tweet as.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

The final dataset consists of seven variables — `X, count,class, hate_speech, offensive_language, neither, and tweet` — and 24,783 instances of potential hate speech tweets. The variable `X` is a unique identifier for each instance. The variable `count` is the number of CrowdFlower users who read and voted on each tweet, `hate_speech` is the number of CrowdFlower users who perceived the tweet as hate speech, `offensive_language` is the number of users who thought the tweet was offensive, and `neither` is the number of users who judged the tweet as neither offensive nor hate speech. `class` is the classification given to each tweet — 0 means the tweet was labeled as hate speech, tweets voted as offensive were labeled as 1, and tweets voted as neither were labeled 2.

For our project, we only needed the tweet and classification — leaving us to drop the rest of the variables.

3.2 Data Characteristics

From Figure 1, we can see that of all 24,783 tweets, only 1,430 (5.77%) are labeled as hate speech and 4,163 (16.80%) are considered as neither hate speech nor offensive. Most tweets — 19,190 (77.42%) — are classified as offensive. The distribution of classes is clearly not even and doesn't line up with a study of Twitter that claims 11.6% of tweets are flagged as hate speech. According to Davidson et al. [2], however, this is most likely caused by how strict their criteria was for being labeled as hate speech.

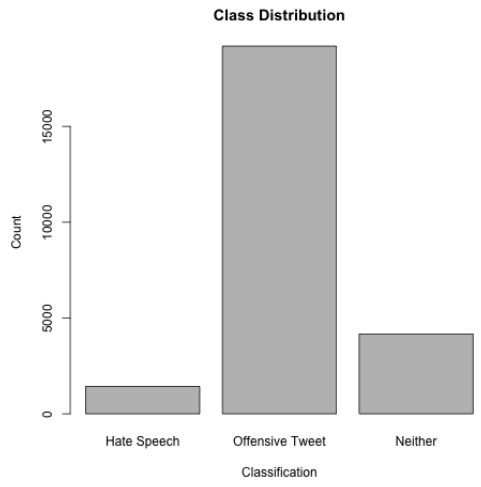

Figure 1: Class Distribution of Tweets

Before starting any text mining on the tweets, we created a new variable `neg` to add to the data frame. This variable marks if a tweet uses the words "no" or "not" — 1 if the tweet contains either string and 0 if it does not. We decided that this was necessary because these words can a positive tweet and vice versa. For example, consider the phrase "no more school shootings". This would most likely be classified as neither hate speech or offensive language. After text mining though (described in a later section), removing stop words would leave the phrase as such: "more school shootings". This tweet would be marked as either offensive language or hate speech now. So keeping the negation information can be helpful for tweets which had previously been incorrectly classified.

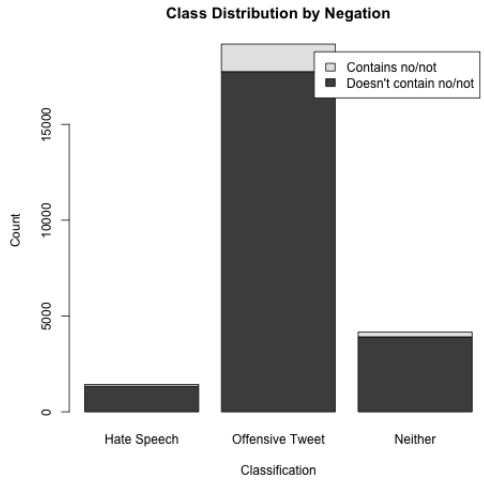

Figure 2: Class Distribution of Tweets by Negation

We can see in Figure 2 that not many tweets contain the words "no" and "not". 6.64% of tweets labeled as hate speech, 7.49% of tweets labeled as offensive language, and 6.0% of tweets labeled as neither contain a negation. As stated previously though, this information can be helpful to correctly classify tweets that have been classified incorrectly.

3.3 Original Tweets

Since we only have the tweet as a variable, text mining is crucial to create features for our model. Originally, tweets looked like the following:

> !!! RT @mayasolovely: As a woman you shouldn't complain about cleaning up your house. & as a man you should always take the trash out...
>
> "@Almightywayne__: @JetsAndASwisher @Gook____ bitch fuck u http://t.co/pXmGA68NC1" maybe you'll get better. Just http://t.co/TPreVwfq0S
>
> "@CarelessOne92: Leafs better win this damn game so I can go riot and shit #EarlyChristmas" you better start looting my nig

From these, we noticed that some parts of the tweet weren't necessary when it came to distinguishing if a tweet has offensive language or is hate speech.

3.4 Text Mining

3.4.1 Tweet Information. First, we dropped the `RT and @` along with the letters that follow the `@`. Next, we dropped URLs that were links to either images or articles. This was all done using general expressions. We decided that knowing who tweeted, if it was retweeted, and if there were links weren't crucial to determine to nature of the tweet.

3.4.2 Text Removal. We then removed all capital letters (and replaced them with lowercase numbers), the remaining punctuation, and all of the numbers. This left us with plain text with only lowercase words. After, English stop words were removed. Stop words are words that are common and neither add nor detract meaning from a sentence, hence why we decided to drop them.

3.4.3 Stemming. The final step to text mining had to do with dropping prefixes and suffixes to words and just keeping the base. This is called stemming. This allows for words with the same base to be counted only once in the term document matrix.

3.5 Tweets After Text Mining

The following show the previous tweets after text mining:

> woman shouldnt complain clean hous man alway take trash
>
> bitch fuck u mayb youll get better just
>
> leaf better win damn game can go riot shit early-christma better start loot nig

3.6 Term Document Matrix

After this text mining, we were able to make a massive term-document matrix. This shows how many times a term (word) is in each document (tweet). Of all 24,783 tweets there are 16,750 distinct words. It is a sparse matrix, but some words show up more than others either across tweets or within the same tweet.

3.7 Additional Features

After the text mining and creating the term-frequency matrix, we next counted how many words were left in the tweet by summing across rows for each document. Next, using a list of offensive and profane words [9], we compared the terms in the matrix to the words on the list and counted how many 'bad words' were in each tweet. With these counts, we were able to find the ratio of bad words to total words in each tweet and use that as a feature.

We predicted that offensive language tweets would have a higher ratio than neither offensive language nor hate speech tweets and that hate speech tweets would have an even higher ratio than the others. From Figure 3, we can see that our predictions were correct. While they all range from 0.0 - 1.0, the medians for hate speech tweets is 0.25, offensive language tweets is 0.2, and neither tweets is 0.0. Over 75% of the ratios for neither tweets are lower than the majority of the offensive language and hate speech tweets.

4 BASELINES

4.1 Feature Extraction

We utilize a naiive implementation—bag of words—as features. Due to the huge number of terms compared to the number of documents, we count the document frequency of every term and conduct experiments using respectively (1) the most frequently used 1000 terms, (2) the least frequently used 1000 terms, and (3) the combination of these two. The intuition behind using the least frequently used words is that such words are typically aligned with a specific scenario/class, which potentially helps to classify tweets. Experiments demonstrate that using the most frequently used 1000 terms

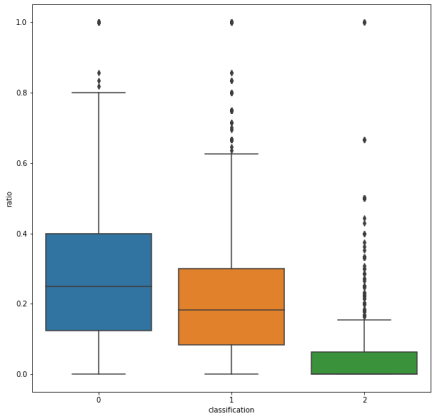

Figure 3: Bad Word Ratio per Class

achieves the best performance. The confusion matrices of each features are summarized in Figure 4.

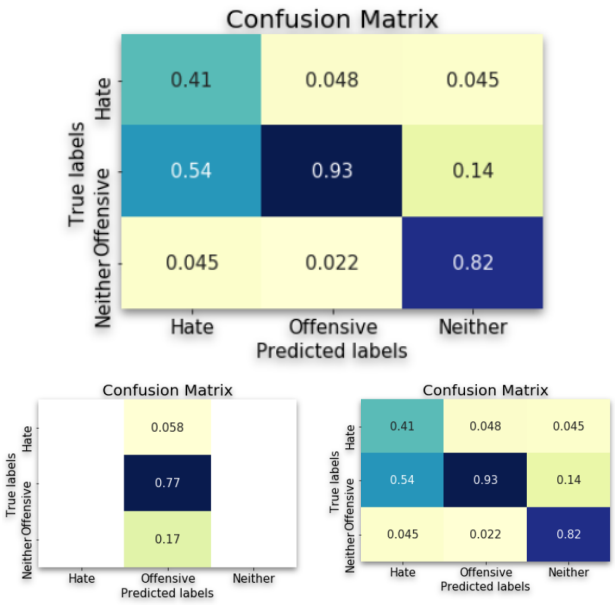

Figure 4: Confusion matrices of various features. (Top) using the most frequently used 1000 terms. (Bottom-left) using the least frequently used 1000 terms. (Bottom-right) using the both. The numbers in confusion matrices are precision.

4.2 Logistic Regression

We use logistic regression model as our baseline model. Logistic regression model is a classifier normally used to classify the binary response variable. Logistic regression is a predictive analysis like

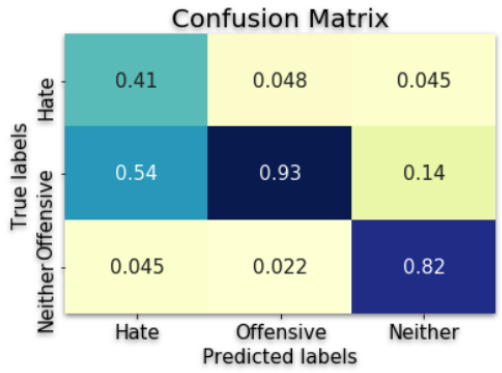

Figure 5: Confusion matrix of baseline method.

	precision	recall	f1-score	support
Hate	0.41	0.20	0.27	359
Offensive	0.93	0.95	0.94	4800
Neither	0.82	0.89	0.85	1038

Table 1: Quantitative Evaluation of baseline method.

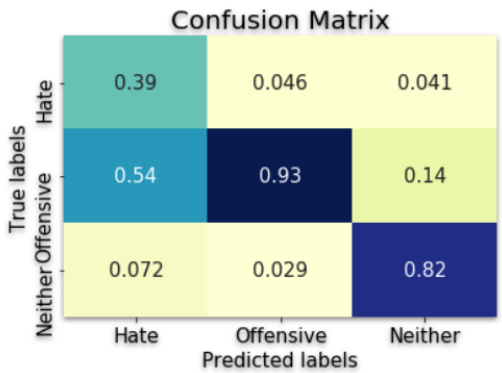

Figure 6: Confusion matrix of logistic regression method with C=10.

	C=0.01	C=0.1	C=1	C=10	C=100
Correct Hate	2	44	73	92	92
F1-score(Hate)	0.01	0.20	0.27	0.31	0.30
Accuracy	0.8821	0.8984	0.8946	0.8868	0.8862

Table 2: Evaluation of Logistic Regression Model with C

other regression analysis. It used to describe the data and find the relationship between dependent response variables and independent predictors. We use logistic regression model to predict our three class response variable by using `multi-class='multinomial'` and `solver='lbfgs'` as the parameters for the model.

For data splitting, we use the package `train_test_split` from `sklearn.model_selection` to split the data into training set and testing set. The proportion for training set is 75% of total data. The proportion of testing set is 25% of total data. In order to keep unification, we set `random_state=0`. We use the training data to train the logistic regression model and use the testing data to test the model and get the performance.

For our baseline model, the accuracy is 0.8946 included the correctly labeled `Hate, Offensive, Neither`. For `Hate`, there are 73 cases are true labeled. For `Offensive`, there are 4550 cases are true labeled. For `Neither`, there are 920 cases are true labeled. Check Figure 5. From the table 1,the precision ,recall and f1-score of class `Hate` are lower than other classes(`Offensive`, `Neither`). The class `Offensive` has the best precision, recall and f1-score. This result shows that the detection of hate speech is not accurate for the baseline model. In next work, our models will focus on improving the accuracy of `hate` class.

5 OUR METHOD I: PARAMETER TUNING

In this part, we use the same data as baseline model. We using only first 1000 sorted document frequency terms as the predictors. For a better performance, we tune the parameter `C`, the inverse of regularization strength in logistic regression. A smaller `C` refers to a stronger regularization while a larger `C` refers to a weaker regularization. We train logistic regression classifiers with `C` as 0.01, 0.1, 1, 10, 100 respectively.

Check the table 2. With `C=10` for logistic model has the best performance on `Hate` cases. The `f1-score` of `C=10` increase to 0.31. Even though the result is not good, the performance is getting better. From this result, we will set `C=10` for all logistic regression models.

Check Figure 6. This is the confusion matrix of logistic regression model with `C=10`. For `Hate` class, there are 92 cases correctly labeled. It increases 26% from the baseline model. The total accuracy of this model is 0.8868, similar with baseline model.

6 OUR METHOD II: SVM

In this experiments, we first augment the features used in baseline by additional ones, described in Section 3.7. Then we utilize Support Vector Machine (SVM) to fulfill this three-class recognition problem. Specifically, due to the time complexity concern, we employ a linear SVM instead of computing kernels. Further, we tune the parameter `C`, the penalty weight of regularization. Different from the logistic regression, a smaller `C` in LinearSVM refers to weaker regularization while a larger `C` refers to a stronger regularization penalty. The number of correct predictions in class Hate, the F-1 score in class Hate and the prediction accuracy using different `C` are summarized in Table 3 and the confusion matrix of the best model is visualized in Figure 9.

	C=15	C=20	C=25	C=28.33	C=300
Correct Hate	82	87	89	91	82
F1-score(Hate)	0.29	0.31	0.31	0.32	0.30
Accuracy	0.889	0.890	0.892	0.892	0.892

Table 3: Quantitative evaluation of SVM Model with various parameters.

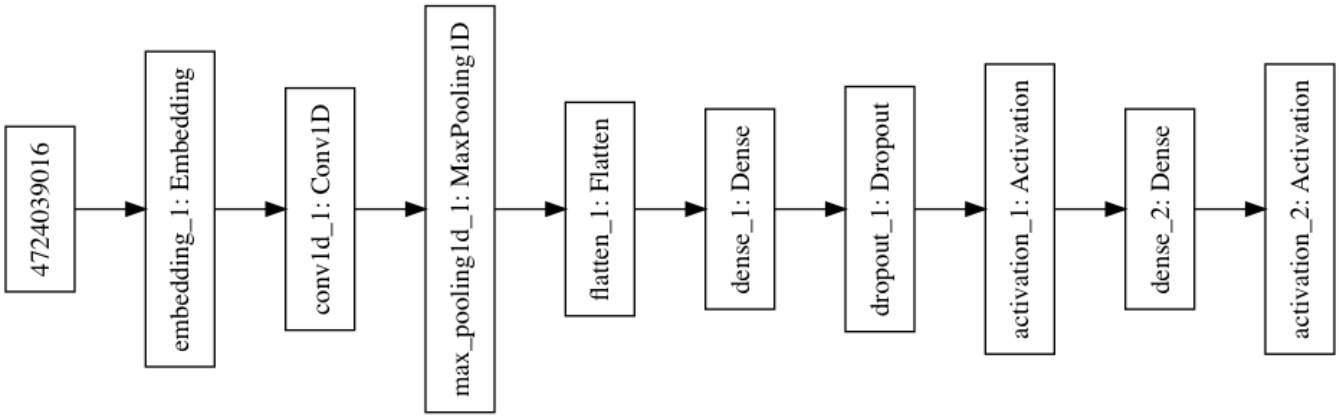

Figure 7: Architecture of convolutional neural network.

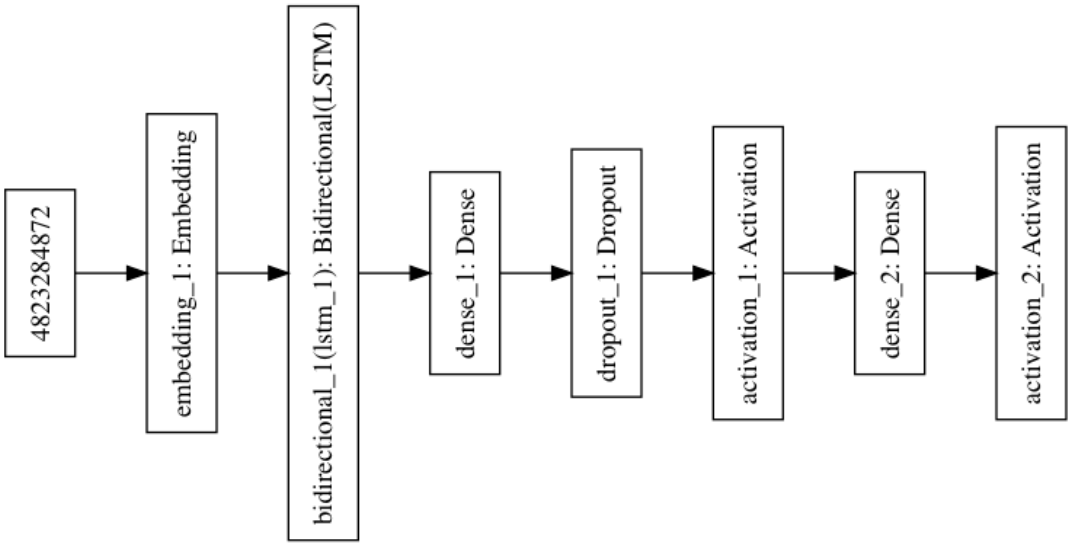

Figure 8: Architecture of Recurrent neural network.

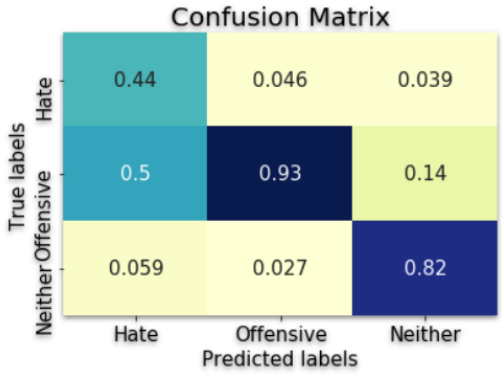

Figure 9: Confusion matrix of SVM with C=28.333.

7 OUR METHOD III: NEURAL NETWORK

Due to the low precision and recall on class Hate, we hereby explore the utility of neural network in our hate speech recognition problem. Different from image applications, it is problematic to directly feed tweets into neural networks. Therefore, we starts from applying word embedding to tweets.

7.1 Word embedding

Word embedding [7] is to build a mapping from language words to a high dimensional space, where each point represents a word. Based on word embedding, each word can be represented by a vector—the coordinates of feature space–and further a sentence can be represented by a matrix, the concatenation of all word vectors. Since neural networks require a consistent dimension of features, we pad zeros for short sentences and clip words for long sentences. Training a word embedding model is beyond the scope of this

project. Thus we download a model trained on twitters [6] and apply it to our data for feature extraction.

7.2 Convolutional Neural Network

We then establish a convolutional neural network to a map from the created word embedding to a vector of three elements, indicating the possibility of each class given the input. We conduct experiments with different architectures from shallow network work to deep network, from smaller kernel to larger kernel convolution, and from smaller number of channels to larger ones. Experiments demonstrate it achieves the best performance when sequentially passing (1) a single layer convolution with kernel size 7, 20 channels, (2) a max pooling layer, (3) a dense layer with 10 units, (4) a dropout layer with rate 0.5, (5) ReLU activation, (6) a dense layer mapping to three units followed by softmax activation. Our final architecture is visualized in Figure 7.

7.3 Recurrent Neural Network

Besides convolutional neural network, we also utilize a recurrent neural network containing a bidirectional Long Short-Term Memory (LSTM) layer to enhance language understanding in a more semantic way. After trying various architecture, our final architecture is (1) bidirectional LSTM layer with 20 units, (2) a dense layer with 8 units, (3) a dropout layer with rate 0.4, (4) ReLU activation, and (5) a dense layer with 3 units followed by a softmax activation. The architecture is visualized in Figure 8.

7.4 Loss Function and Implementation Details

Since our problem is inherently a three-category classification instead of binary classification, we utilize a categorical cross-entropy loss. As for training, we train CNN for 30 epochs and RNN for 10 epochs because RNN is slower. For training algorithm, both neural network are trained by Adam optimizor with exponentially reduced learning rate. We implement these two neural networks using Keras.

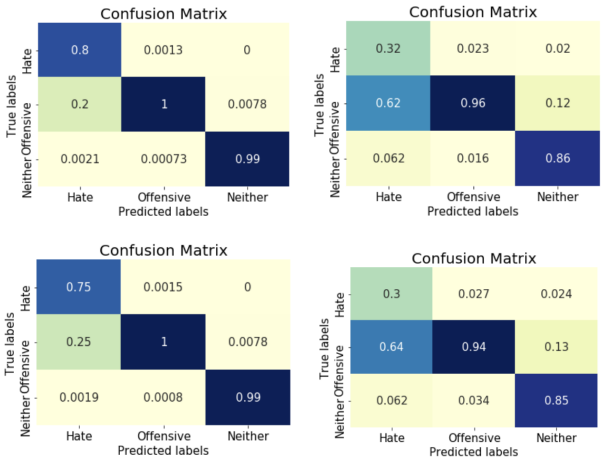

Figure 10: Confusion matrix of neural networks. (Top-left) CNN on training set. (Top-right) CNN on test set. (Bottom-left) RNN on training set. (Bottom-right) RNN on test set.

To solve the problem of unbalanced data distribution along different classes, we apply corresponding sample weight to different classes.

7.5 Experiments

We split overall data into training and test set with split ratio 0.2. Confusion matrix of both neural network on training and test set are summarized into Figure 10. The F-1 score on class Hate are 0.88 for CNN on training and 0.42 on test; 0.85 for RNN on training and 0.38 on test. Experiments demonstrate that CNN achieves the highest precision and RNN achieves the highest recall in our all experiments.

To compare our RNN model to the benchmark proposed along with the dataset [2], we follow the same training and test procedure and show the computed confusion matrices in Figure 11. One can see that our modern RNN dramatically outperforms the benchmark.

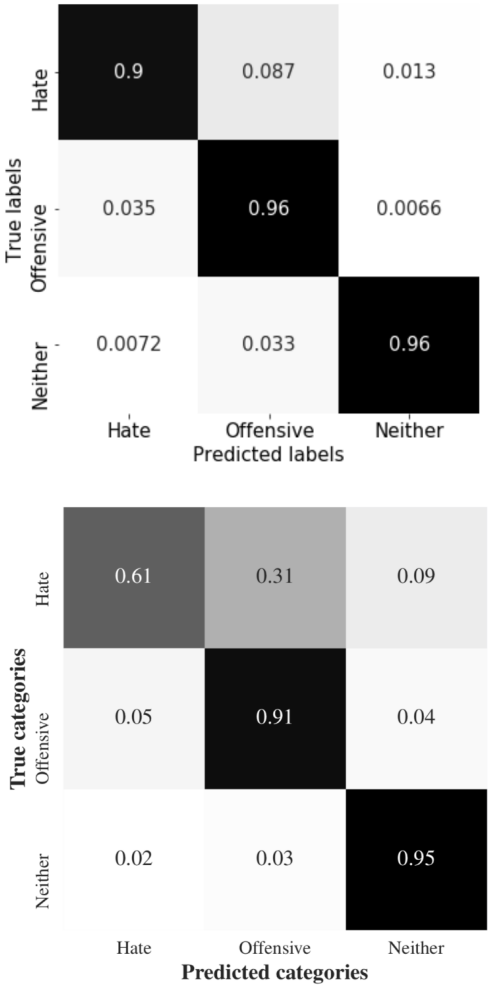

Figure 11: Confusion matrix of our proposed RNN (top) against the benchmark [2] (bottom).

8 RESPONSIBILITY

The project is a co-work between Olivia Keirn and Jinmei Zheng. The contributions of each author are:

- **Olivia Keirn**: data analysis, text mining, creating additional features, and feature extraction.
- **Jinmei Zheng**: word embedding, training and testing classifiers including logistic regression, support vector machine and neural networks.

9 DISCUSSION

Our solution to the problem of hate speech recognition exceeds the performance of the benchmark solution found by Davidson et al. [2]. Using neural networks, especially a recurrent neural network, using word embedding significantly increases our results.

One of the next steps to continue this project would be to pursue more feature engineering. Most tweets on Twitter use emoji's. Being able to incorporate the connotation of an emoji — angry face, smiley face, crying emoji — in a tweet may be able to help classify tweets better. Also, finding another list of 'bad words' that is geared towards Twitter could be useful to find common bad words found in regular tweets vs. bad words commonly found in flagged/reported tweets.

Another possible extension to our project would be to pull a new dataset from Twitter to try our classifier on. This way, we can see how well our classifier works on different data and decide if more adjustments need to be made. Since tweets are relatively short due to the character limit, this project would be interesting on Facebook posts since there are no limitations on length. Because of the lack of length limitation, the feasibility of this task is called into question. Our dataset of 25,000 tweets created 16,750 features, so growing the number of terms could become difficult. Testing across platforms would be the goal however, because having a versatile classifier would be the most useful for social media users and administrators today.

10 CONCLUSION

Our goal for this project was to be able to classify a tweet as hate speech, offensive language, or neither. While previous authors have tried this, we wanted to create a new approach that performed even better on social media posts. The text mining performed helped to cut down the term document matrix significantly, but still keep all of the relevant information we need in order to classify a tweet. We added two new features as well — negating term and pulling out the bad words in each tweet to find a ratio of bad words to total words in a tweet. After this, we were able to build classifiers to identify hate speech and offensive language, or neither.

We utilized two types of classifiers to solve the problem of hate speech recognition: one is conventional classifiers (e.g. SVM, logistic regression) and the other is modern neural networks. For conventional classifiers, we extracted bag-of-words features via term frequency, trained logistic regression and SVM, and tuned parameters. For neural network, we extracted features via word embedding learned from twitter data and trained a convolutional neural network and a recurrent neural network. Experiments demonstrate the superior performance of our proposed models. It is worth noting that our RNN dramatically outperforms the benchmark of the hate speech dataset.

REFERENCES

[1] Pete Burnap and Matthew L Williams. 2015. Cyber hate speech on twitter: An application of machine classification and statistical modeling for policy and decision making. *Policy & Internet* 7, 2 (2015), 223–242.

[2] Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. *arXiv preprint arXiv:1703.04009* (2017).

[3] Njagi Dennis Gitari, Zhang Zuping, Hanyurwimfura Damien, and Jun Long. 2015. A lexicon-based approach for hate speech detection. *International Journal of Multimedia and Ubiquitous Engineering* 10, 4 (2015), 215–230.

[4] James B Jacobs, Kimberly Potter, et al. 1998. *Hate crimes: Criminal law & identity politics*. Oxford University Press on Demand.

[5] Irene Kwok and Yuzhou Wang. 2013. Locate the Hate: Detecting Tweets against Blacks.. In *AAAI*.

[6] Quanzhi Li. 2017. Word Embedding Data Sets Learned from Tweets and General Data. https://doi.org/10.5281/zenodo.581402

[7] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. 3111–3119.

[8] Leandro Araújo Silva, Mainack Mondal, Denzil Correa, Fabrício Benevenuto, and Ingmar Weber. 2016. Analyzing the Targets of Hate in Online Social Media.. In *ICWSM*. 687–690.

[9] Louis von Ahn. [n. d.]. Offensive/Profane Word List. https://www.cs.cmu.edu/~biglou/resources/

[10] Samuel Walker. 1994. *Hate speech: The history of an American controversy*. U of Nebraska Press.

[11] William Warner and Julia Hirschberg. 2012. Detecting hate speech on the world wide web. In *Proceedings of the Second Workshop on Language in Social Media*. Association for Computational Linguistics, 19–26.

[12] Zeerak Waseem and Dirk Hovy. 2016. Hateful symbols or hateful people? predictive features for hate speech detection on twitter. In *Proceedings of the NAACL student research workshop*. 88–93.