VDGOOD Journal of Computer Science Engineering

Detection of Hate Speech and Offensive Language in Twitter Using Sentiment Analysis

Harish Krishna K[a*], Thirunayan Manikantan R[b], Aravind Kumar D[c], Amirthavalli R[d],

[a,b,c,]*UG Scholar, Department of Computer Science and Engineering, Velammal Engineering College, Chennai, India.*

[d]*Assistant Professor, Department of Computer Science and Engineering, Velammal Engineering College, Chennai, India*

VDGOOD use only: Received 14-Mar-2020; revised 30-Mar-2020; accepted 20-Apr-2020

Abstract

The exponential growth of social media such as Twitter and other community forums has revolutionized communication and content publishing, but is also increasingly exploited for the propagation of hate speech and the organization of hate-based activities. The anonymity and mobility standard afforded by such media has made the breeding and spread of hate speech – eventually leading to hate crime – effortless in a virtual world that are beyond the realms of law enforcement. Existing methods in the detection of hate speech primarily cast the problem as a supervised document classification task. These can be divided into two categories: one relies on manual feature engineering that are then consumed by algorithms such as SVM, Naive Bayes, and Logistic Regression (classic methods); the other represents the more recent deep learning paradigm that employs neural networks to automatically learn multilayers of abstract features from raw data (deep learning methods). In this method we show that it is a much more challenging task, as our analysis of the language in the typical datasets shows that hate speech lacks unique, discriminative effects and therefore is found in the 'long tail' in a dataset that is difficult to discover. We then propose Deep Neural Network (DNN) structure as a unique feature extractors that are particularly effective for capturing the semantics of hate speech. Our methods are tested on the largest collection of hate speech datasets based on Twitter, and are shown to be able to outperform state of the art by up to 6 percentage points in macro-average F1, or 9 percentage points in the more challenging case of identifying hateful content. As a proxy to quantify and compare the linguistic characteristics of hate and non-hate Tweets, we also propose to study the 'uniqueness' of the vocabulary for each class. © 2020 VDGOOD Professional Association. All rights reserved

Keywords: Classic Methods; DNN; Detection of hate speech and offensive language in Twitter; Sentimental Analysis.

1. Introduction

We have chosen to work with twitter as we feel it is a better approximation of public sentiment as opposed to conventional internet articles and web blogs. The reason is that the amount of relevant data is much larger for twitter, as compared to traditional blogging sites. Moreover the response on twitter is more prompting and also more general (since the number of users who tweet is substantially more than those who write web blogs on a daily basis). Sentiment analysis of public is highly difficult in macro-scale socioeconomic phenomena like predicting the stock market rate of a particular firm. This could

be done by analyzing overall public sentiment towards that firm with respect to time and using economics tools for finding the correlation between public sentiment and the firm's stock market value. Firms can also estimate how well their product is responding in the market, which areas of the market is it having a favorable response and in which a negative response (since twitter allows us to download stream of geo-tagged tweets from a particular locations. If firms can able to get this information they can analyze the reasons behind geographically differentiated response, and so they can market their product in a more optimized manner by looking for appropriate solutions by establishing suitable market segments. Predicting the results of the political elections and polls is also an emerging application to sentiment analysis. One of a study was conducted by Tumasjan et al. In Germany for predicting the result of the federal elections in which concluded that twitter is such a good reflection of offline sentiment analysis .

2. Data Description

The data given is in the form of a comma-separated values in other words CSV files that has tweets and their corresponding sentiments. The training datasets in the csv file is of the type format tweet_id, sentiment, tweet where the tweet_id is a unique integer identifying the tweet, sentiment is either 1(positive) or 0(negative), and tweet is put inside in " ". Similarly, the training dataset is a csv file of type tweet_id, tweet. The dataset is a collection of words, emoticons, symbols, URLs and references to people. Words and emoticons provide to predicting the sentiment, but URLs and references to people don't. Therefore, URLs and references can be ignored. The words are also a mixture of misspelled words,words with many repeated letters and extra punctuations. The tweets, thereby, need to be pre-processed to standardize the dataset .The provided training and test dataset have 5000 and 1000 tweets respectively. Preliminary and the statistical analysis of the contents of datasets, after preprocessing might change.

2.1. Pre-processing

Raw tweets taken from the twitter generally result in a noisy dataset. This is due to the casual nature of people's usage of social media. Tweets have certain special characteristics such as re-tweets, emoticons, user mentions, etc. which have to be suitably extracted. Therefore, the raw twitter datasets has to be normalized to create a dataset in which it can be easily

learned by various classifiers. We have applied an extensive number of pre-processing procedures for standardizing the dataset and reduce its size. We first do some general pre- processing of data set tweets which are.

Convert the tweet to lower case. Replace 2 or more dots (.) with space.

Remove the spaces and quotations from the ends of tweet datasets. Change two or more spaces with a single space.

2.2. URL

Most of the users often share hyperlinks to other webpages in their tweets and comments . Any particular URL is not important for text divisions as this would lead to very sparse features. Therefore, we replace all the URLs in tweets with the word URL. The regular expression used to match URLs is ((www\.[\S]+)|(https?://[\S]+)).

2.3. User Mention

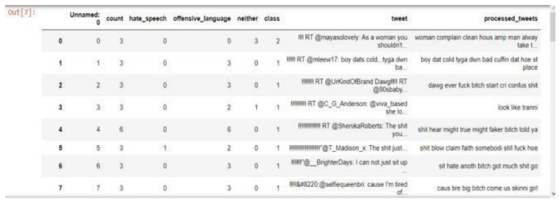
Every twitter account users will have a handle related with them. Users are most often mentioning the other users in their tweets by a at symbol eg:@handle. We will remove all account with the word USER_MENTION. The regex used to match account mention with @[\S]+. Make sure that the symbols you use in the equation has been defined previously or after the equation.

2.4. Emoticons

Users often use emotions to express. It is impossible to exhaustively match all emoticons used on social media. However, we try match some most commonly used emoticons which are used very frequently. We replace the matched emoticons to EMO_POS or EMO_NEG depending upon on it's a positive or a negative emotion.

2.5. Hashtag

Hashtags are unspaced phrases prefixed by the hash symbol (#) which is frequently used by users to mention a trending topic on twitter. We replace all the hashtags with the words with # symbol . For example, #hello is replaced by hello. The regex used to match hashtags is #(\S+).



Out[7]:

	Unnamed: 0	count	hate_speech	offensive_language	neither	class	tweet	processed_tweets
0	0	3	0	0	3	2	!!! RT @mayasolovely: As a woman you shouldn't...	woman complain clean hous amp man alway take t...
1	1	3	0	3	0	1	!!!!! RT @mleew17: boy dats cold...tyga dwn ba...	boy dat cold tyga dwn bad cuffin dat hoe st place
2	2	3	0	3	0	1	!!!!!!! RT @UrKindOfBrand Dawg!!!! RT @80sbaby...	dawg ever fuck bitch start cri confus shit
3	3	3	0	2	1	1	!!!!!!!!! RT @C_G_Anderson: @viva_based she lo...	look like tranni
4	4	6	0	6	0	1	!!!!!!!!!!!!! RT @ShenikaRoberts: The shit you...	shit hear might true might faker bitch told ya
5	5	3	1	2	0	1	!!!!!!!!!!!!!!!!!!"@T_Madison_x: The shit just...	shit blow claim faith somebodi still fuck hoe
6	6	3	0	3	0	1	!!!!!!"@__BrighterDays: I can not just sit up ...	sit hate anoth bitch got much shit go
7	7	3	0	3	0	1	!!!&#8220;@selfiequeenbri: cause I'm tired of...	caus tire big bitch come us skinni girl

Fig.1. Data set of twiter hate speech

3. Feature Extraction

We extract 2 types of features, namely unigrams and bigrams. We produce a frequency distribution among unigrams and bigrams present in the dataset and choose top N
unigrams and bigrams for our analysis.

```
Out[3]: <matplotlib.axes._subplots.AxesSubplot at 0x1fc8aee7f60>
```

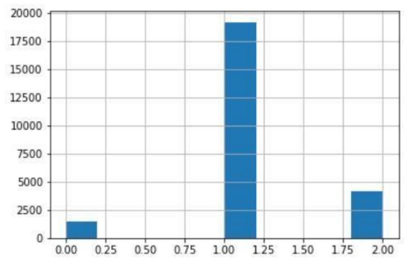


Fig.2. Insight of the data set in offencive language

A. Unigrams

Probably the simplest and the most commonly used features for text classification is the presence of single words or tokens in the the text. We, therefore, use top N words from these to create our own word sets where N is 15000 for sparse vector classification and 90000 for dense vector classification. The frequency distribution follows Zipf's law which states that in a large sample of words, the frequency of a word is inversely proportional to its rank in the frequency table. This can be seen by the fact that a linear trendline with a negative slope fits the plot of log(Frequency) vs. log(Rank).

B. Bigrams

Bigrams are word pairs in the dataset which occur in succession in the corpus. These features are a good way to model negation in natural language like in the phrase – This is not good. We therefore use only top 100000 bigrams from these to create our words list.

4. Feature Representation

After extracting the unigrams and bigrams, we represent each tweet as a feature vector in either sparse vector representation or dense vector representation depending on the classification method.

A. Dense Vector Representation

For dense vector representation we use a word sets of unigrams of size 90000 i.e. the top 90000 words in the dataset. We assign an corresponding integer index to each word depending on its rank (starting from the index 0) which means that the most common word is assigned the number 10, the second most common word is assigned the number 11 and so on. Each tweet is then represented by a vector of these indices which is a dense vector.

B. Naive Bayes

Naive Bayes is a simple that can be used for text classification. In this model, the class c^ is assigned to a tweet t, where In the formula above, fi represents the i-th feature of total n features. P(c) and P(fi|c) can be obtained through

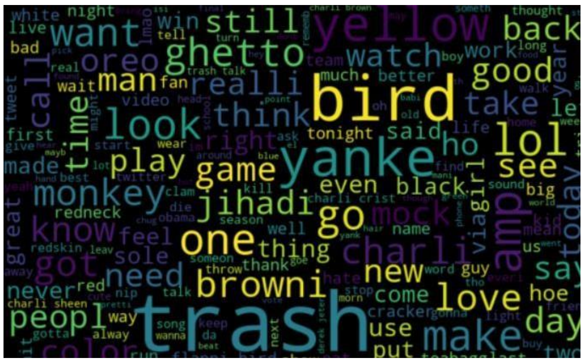


Fig.3. Representation of the hate words in bigraph

maximum likelihood estimates.We used MultinomialNB from sklearn.naive_bayes package of scikit-learn for Naive Bayes classification. We used Laplace smoothed version of Naive Bayes with the smoothing parameter set to its default value of 1. We found that presence features outperform frequency features because Naive Bayes is built to work best on integer features in data sets rather than traditional features.

C. Decision Tree

Decision trees are a classifier model that compares each node of the tree is a test on the data set, and its children represent the outcomes. The leaf nodes represents the final classes of the data points.For each node in the tree the best test condition or decision has to be taken. We use the GINI factor to decide the best split. We use the Decision Tree Classifier in sklearn.tree package given by scikit-learn to build our representation . GINI can be used to evaluate the split of every node and the best split is considered . The model has been performing a bit better using the presence this feature compared to frequency. Also using unigrams didn't make any significant improvements. The best accuracy achieved using decision trees was 68.1%.

D. Random Forest

Random Forest is the most popular learning algorithm for classification and regression. Random Forest creates a multiple and multitude of decision trees classifies based on the aggregated decision corresponding to the trees. For a set of tweets x1; x2; : : : xn and their respective sentiment labels y1; y2; : : :n bagging repeatedly selects a random sample (Xb, Yb) with replacement. Each divisionof tree fb is trained by random sample (Xb, Yb) where b ranges from 1 : : : B. Finally, the decisions are based on the result of the B trees.We implemented random forest algorithm by using Random Forest Classifier in sklearn.ensemble given by scikit-learn. We tested using 10 estimators (trees based) using both presence system and frequency features. Presence features updates performed better than frequency but the improvement was not substantial.

E. SVM

SVM, aka support vector machines, is a non-probabilistic binary linear classifier in the data set . For a training set of points (xi; yi) where x is the feature vector and y is the class, we want to find the maximum-margin hyperplane that divides the points with yi = 1 .

F. Convolutional Neural Network

We used keras with TensorFlow in the backend to implement the (CNN) Convolutional Neural Network model. We used the Dense Vector Representation (DVR) of the tweets to train our CNN models. We used a vocabulary of top 5000 words from the training dataset. We represent each word in our vocabulary with an integer index from 1 : : : 5000 where the integer index represents the rank of the word in the dataset. The integer index 0 is reserved for the special padding word. Further each of these 5000+1 words is represented by a 200 dimensional vector. The first layer of our models is the Embedding layer which is a matrix of shape (v + 1) d where v is vocabulary size (=5000) and d is the dimension of each word vector (=200). We initialize the embedding layer with random weights from N (0; 0:01). Each row of this embedding matrix represents represents the 200 dimensional word vector for a word in the vocabulary. For words in our vocabulary which match GloVe word vectors provided by the StanfordNLP group, we seed the corresponding row of the embedding matrix from GloVe vectors. Each tweet i.e. its Dense Vector Representation (DVR) is padded with 0s at the end until its length matches to max_length which is a parameter we tweak in our experiments. We also conducted experiments using SGD+ Momentum weight updates and found out that it takes longer (100 epochs) to converge compared to validation accuracy equivalent to Adam. We ran our model upto 10 epochs. Using the Adam weight update scheme, the model converges very fast (4 epochs) and begins to overfit badly after that. We, therefore, use models from 3rd or 4th epoch for our results.

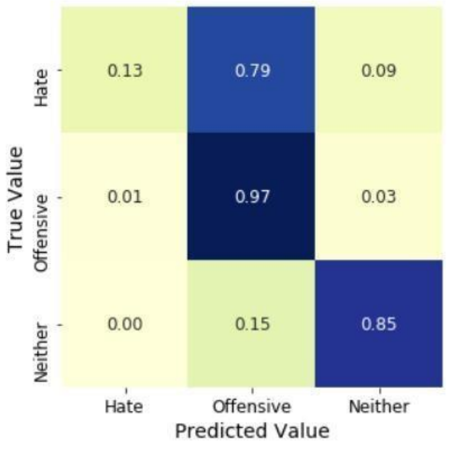
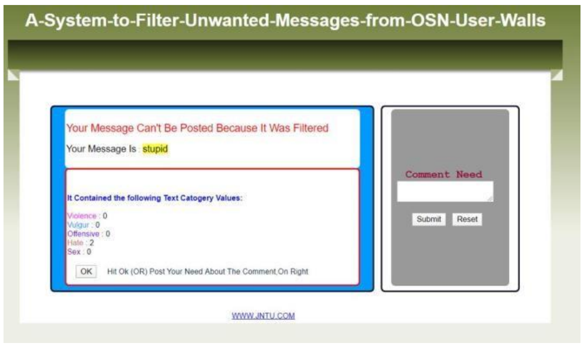


Fig.4. prediction ratio of the hate speech

5. Conclusion

A. Summary of achievements

The provided tweets were a mixture of words, emoticons, URLs, hastags, user mentions, and symbols. Before training the we pre-process the tweets to make it suitable for feeding into models. We implemented several machine learning algorithms like Naive Bayes, Maximum Entropy, Decision Tree, Random Forest, SVM, Recurrent Neural networks and Convolutional Neural Networks to classify the polarity of the tweet. We used two types of features namely unigrams and bigrams for classification and observes that augmenting the feature vector with bigrams improved the accuracy. Once the feature has been extracted it was represented as either a sparse vector or a dense vector. It has been observed that presence in the sparse vector representation recorded a better performance than frequency. Neural methods performed better than other classifiers in general. Our best LSTM model achieved an accuracy of 83.0% on Kaggle while the best CNN model achieved 83.34%. The model which used features from our best CNN model and classifies using SVM performed slightly better than only CNN. We finally used an ensemble method taking a majority vote over the predictions of 5 of our best models achieving an accuracy of 83.58%.


A-System-to-Filter-Unwanted-Messages-from-OSN-User-Walls

Your Message Can't Be Posted Because It Was Filtered

Your Message Is : stupid

It Contained the following Text Catogery Values:

Violence : 0
Vulgur : 0
Offensive : 0
Hate : 2
Sex : 0

OK Hit Ok (OR) Post Your Need About The Comment On Right

Fig. 5. The detection of hate speech

References

[1] Albert Biffet and Eibe Frank. Sentiment Knowledge Discovery in Twitter Streaming Data. Discovery Science, Lecture Notes in Computer Science, 2010, Volume 6332/2010, 1-15, DOI: 10.1007/978-3-642-16184-1_1

[2] Alec Go, Richa Bhayani and Lei Huang. Twitter Sentiment Classification using Distant Supervision. Project Technical Report, Stanford University, 2009.

[3] Alexander Pak and Patrick Paroubek. Twitter as a Corpus for Sentiment Analysis and Opinion Mining. In Proceedings of international conference on Language Resources and Evaluation (LREC), 2010.

[4] Andranik Tumasjan, Timm O. Sprenger, Philipp G. Sandner and Isabell M. Welpe. Predicting Elections with Twitter: What 140 Characters Reveal about M. Young, The Technical Writer's Handbook. Mill Valley, CA: University

[8] Hatzivassiloglou, V, & McKeown, K.R. Predicting the semantic orientation of adjectives. In Proceedings of the 35th Annual Meeting of the ACL and the 8th Conference of the European Chapter of the ACL, 2009.

[9] Johann Bollen, Alberto Pepe and Huina Mao. Modelling Public Mood and Emotion: Twitter Sentiment and socio- economic phenomena. In Proceedings of AAAI Conference on Weblogs and Social Media (ICWSM), 2011.

[10] Luciano Barbosa and Junlan Feng. Robust Sentiment Detection on Twitter from Biased and Noisy Data. In Proceedings of the international conference on Computational Linguistics (COLING), 2010.

[11] Peter D. Turney. Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews. In Proceedings of the Annual Meeting of the Association of Computational Linguistics (ACL), 2002.

[12] Rudy Prabowo and Mike Thelwall. Sentiment Analysis: A Combined Approach. Journal of Infometrics, Volume 3, Issue 2, April 2009, Pages 143-157, 2009.
Sentiment in Microblogs. In Proceedings of Empirical Methods on Natural Language Processing (EMNLP), 2011.