Improving twitter hate speech detection using graph-based features

Sudhanshu Ranjan
Indian Institute of Technology
Guwahati
sudhanshu.ranjan@iitg.ac.in

Ankit Kumar Singh
Indian Institute of Technology
Guwahati
ankit.ks@iitg.ac.in

Abstract

Hate speech detection in social media has gained a lot of attention. A large number of deep learning techniques have been used for hate speech detection. Most of these techniques train the model to look at a sentence and make prediction. Text-gcn was a recently proposed model that works by creating a corpus graph and the architecture allows document document interaction in order to make better predictions. We show the features learned from text-gcn can improve the performance of the baseline models.

1 Introduction

Hate speech detection in social media is an important task as there are large a large number of instances of people using hateful and offensive speech online. Though most of the social media platform have rules that prohibit use of hate speech, automating its detection is a difficult task.

A large number of methods that have been applied to the problem rely either on TF-IDF features based on n-grams generated from corpus, or on convolutional neural networks or recurrent neural networks. Most of these networks rely only on the documents and the word embedding to predict the output. Graph Convolutional network (Kipf and Welling, 2017) was a recent network proposed that improved significantly the performance of neural networks on graphs. Based on GCN, text-GCN (Zhang and Luo, 2018) was proposed that achieved benchmark results on multiple text classification datasets. The model jointly learns the embeddings for word and the documents using the labels of the documents. The text-GCN model is different from CNNs or RNNs because a two layer text-GCN network will allow the interaction between two documents and hence has the capability to learn features different from RNN or CNN which don't explicitly allow the interactions between two documents.

The paper focuses on the task of hate speech detection. We use convolutional neural network and GRU as baseline models for the task. The paper shows that using corpus-graph based features can improve the performance of the baseline models.

2 Related methods

Logistic regression Davidson et al. (2017) used unigram, bigram, trigram weighted by TF-IDF, combined them with syntactic features like POS tag and used Flesch-Kincaid Grade Level and Flesch Reading Ease scores to detect quality of tweet. They also used sentiment score of each tweet, features like URLs count and retweet counts, and then used Logistic Regression(LR) with L2 regularization to train a classifier. Previosuly, Waseem and Hovy (2016) had also used a logistic regression based on n-gram features.

Neural networks Park and Fung (2017) used a convolutional neural network for the task. They used a character level CNN, a word level CNN and a hybird CNN that uses both of them. For character, they use one-hot encoding and for word, they use a 300-dimensional word2vec embedding. Zhang et al. (2018) used combination of convolutional neural networks and GRU. Founta et al. (2018) used a GRU to improve upon previous methods. The paper also proposed a variant where it used the metadata like network of user to obtain obtain better performance. Kshirsagar et al. (2018) proposed TWEM, a simpler model where they used word embedding, max pooling and average pooling to improve upon previous models.

3 Methodology

Our model combines the sentence embedding obtained from Text-GCN with those of CNN/GRU.

3.1 Text-GCN

The model works by creating a weighted graph from the corpus and then train a Graph Convolutional Network on top of it. The nodes of the graph are the words and the documents and the edges are links between two words and also between word-document. Let E be the edge matrix that contains the weight of edge between the nodes.

$$E_{ij} = \begin{cases} PMI(i,j) & i,j \; are \; words \\ TF-IDF_{ij} & i \; is \; document, j \; is \; word \\ 1 & i = j \\ 0 & otherwise \end{cases}$$

where

$$PMI(i,j) = log\frac{p(i,j)}{p(i)p(j)}$$

$$p(i,j) = \frac{\#W(i,j)}{\#W}$$

$$p(i) = \frac{\#W(i)}{\#W}$$

where W(i) denotes the number of sliding windows that contain word i, W(i, j) denotes the number of sliding windows that contain both word i and j and W denotes the total number of sliding windows. The edges with negative weights are discarded. The original paper after building the graph used a two layer GCN and then fed it to the final softmax layer. The activation used is ReLU network. During experimentation, we found using a three layer GCN-network gave better results. We use the cross entropy loss. To prevent over fitting, we use dropout after each layer.

$$H^{(1)} = ReLU(\tilde{E}XW_0)$$
$$H^{(2)} = ReLU(\tilde{E}H^{(1)}W_1)$$
$$Z = softmax(\tilde{E}H^{(2)}W_2)$$

where $\tilde{E} = D^{-\frac{1}{2}}ED^{-\frac{1}{2}}$

In general X is the feature matrix. Here, we keep it as identity matrix, i.e. $X = I$.

The three layer network allows message passing between nodes that are three layers away which allows interaction between two document nodes.

3.2 CNN

For CNN, we use a word level static CNN (Kim, 2014). For word representation, we use 100-dimensional Glove word embedding (Pennington et al., 2014) that were explicitly trained on twitter.

Categories	Count
hate	1430
offensive	4163
none	19190

Table 1: Label wise count

3.3 GRU

Similar to CNN, we use a word level static GRU (Chung et al., 2014) with 100-dimensional Glove word embedding.

The final model utilizes the sentence embedding learned from Text-GCN. This embedding is concatented with the embedding learnt from CNN/GRU and then passed to the final softmax layer. The architecture for this can be seen in figure 1.

4 Dataset

We use the dataset from Davidson et al. (2017). The dataset contains tweets from three categories hate, offensive, neither. The count of each category is summarized in table 1. We split the dataset into train , development and test with ratio 90%, 10% and 10%. For hyper parameter tuning, we use the development set and report the results on the test set.

5 Training details

For training, we initially trained the model for CNN/GRU and text-GCN separately. For CNN, we keep the filter size as 3, 4, 5 and number of filters as 100. For GRU, we keep the number of hidden units as 100. Using the parameters obtained by training, we then concatenate the final layer outputs from both the models and then train the network again. For regularization, we use dropout. For text-GCN, dropout is applied at each layer of text-GCN, whereas for CNN/GRU the dropout is applied only before the final layer. We use Adam optimizer (Kingma and Ba, 2014) and set learning rate to 0.001 and dropout parameter to 0.5. We train all the models for 30 epochs.

6 Results

The results are shown in the table 2. Using corpus graph based features improved the result of both the baseline models. To establish the significance, we use the t-test and report values with significance level $p < 0.05$.

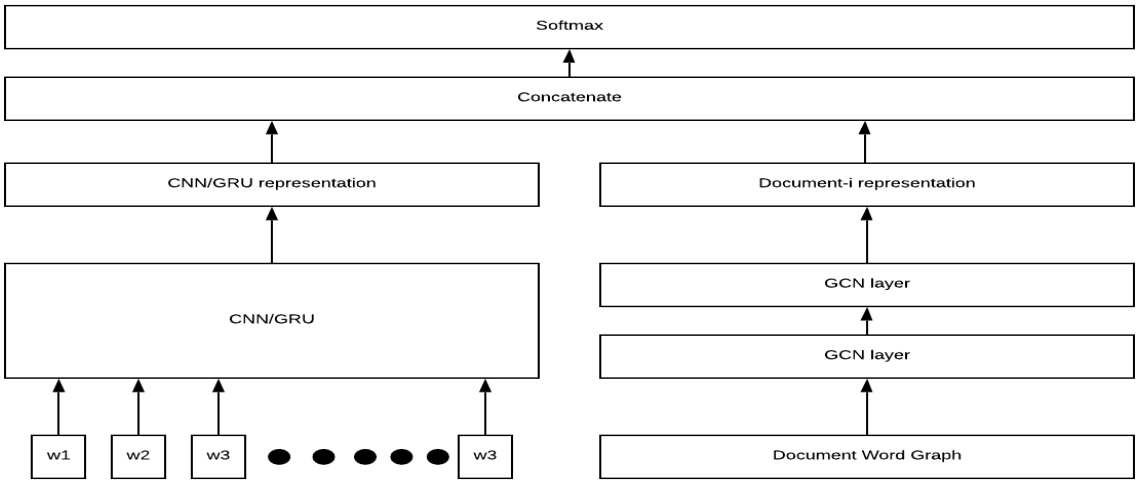

Figure 1: Architecture of the model.

Model	F1 score
Text-GCN	83.11
CNN	89.45
GRU	89.55
CNN + CG	**89.79**
GRU + CG	89.63

Table 2: Results using corpus graph. We show the results of baseline Text-GCN, CNN and GRU and the improvements obtained by using Corpus Graph along with them dentoed as CNN+CG and GRU+CG.

7 Future work

As pointed out in Davidson et al. (2017), our model also suffers from the problem of distinguishing between offensive and hate speech. Secondly, we need better models that focus on small length texts since the average length of tweet is approximately nine words which is way less then usual sentence length of other datasets used for benchmarking sentence classification task.

References

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language.

Antigoni-Maria Founta, Despoina Chatzakou, Nicolas Kourtellis, Jeremy Blackburn, Athena Vakali, and Ilias Leontiadis. 2018. A unified deep learning architecture for abuse detection.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*.

Rohan Kshirsagar, Tyrus Cukuvac, Kathy McKeown, and Susan McGregor. 2018. Predictive embeddings for hate speech detection on twitter. In *Proceedings of the 2nd Workshop on Abusive Language Online (ALW2)*, pages 26–32, Brussels, Belgium. Association for Computational Linguistics.

Ji Ho Park and Pascale Fung. 2017. One-step and two-step classification for abusive language detection on twitter.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

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*, pages 88–93, San Diego, California. Association for Computational Linguistics.

Z. Zhang, D. Robinson, and J. Tepper. 2018. Detecting hate speech on twitter using a convolution-gru based

deep neural network. In *Proceedings of the 2018 Extended Semantic Web Conference*.

Ziqi Zhang and Lei Luo. 2018. Hate speech detection: A solved problem? the challenging case of long tail on twitter.