Hate Speech Detection on Twitter

Jainul N. Vaghasia
Department of Computer Science
University of Washington
Seattle, WA 98105
jnv3@cs.washington.edu

Abstract

We explore a neural network based method for classifying the presence of hate speech in a tweet. The network is based on word embeddings and average pooling, and produces state-of-the art results on offline model. The method when combined with resampling of subset of past training data performs well under severe class imbalance in online setting. With correct queue size, we obtain robust behavior against moderate concept drift in the stream as well.

1 Introduction

Cyberbullying has become a rather common incident in that Duggan and Smith [2013] found that, as of 2013, 73% of people had witnessed harassment online, and a full 40% of people had experienced harassment directly. Such online harassment often includes posts with sexually violent language, threats, hate speech and degrading racist terms. Being able to identify cases of toxic tweets can therefore help in tackling this problem at its core. To this end, we frame the following binary classification problem: given a tweet, we wish to determine whether a tweet is offensive or not. There is no agreed upon definition of "offensive" owing to the ambiguities in human communication, but we use the following specific definition provided by Davidson et al. [2017]: "language that is used to express hatred towards a targeted group or is intended to be derogatory, to humiliate, or to insult the members of the group."

We then approach this problem with two different perspectives. First, we handle the tweets in an offline manner in which tweets are provided in a batch beforehand. We optimize our model against class imbalance arising from the fact that offensive tweets are expected to be relatively fewer than non-offensive tweets. Second, we adapt our model for online learning where tweets arrive at different time steps such as when a reviewer flags a tweet as inappropriate. In doing so, we address the issue of class imbalance in the online setting while using minimal storage space. Furthermore, we address the phenomenon of concept drift which is defined as the variation of the underlying distribution of tweet classes with time.

All code including backprop was written on basic libraries for instruction purposes and is available here: `https://github.com/JainulV/Hate-Detection-Twitter`.

2 Related Work

Many efforts have been made to classify hate speech using data scraped from online forums. Waseem and Hovy [2016] and Davidson et al. [2017] applied logistic regression with engineered features such as n-grams, sentiment and tweet metadata on Sexism/Racism and HATE datasets respectively. Founta et al. [2018] implemented two deep neural networks trained using transfer learning on tweet data and user metadata. Shen et al. [2017] incorporate the use of simple word embeddings whose model we modify to use transformed word embeddings model (TWEM).

32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, Canada.

Table 1: Dataset Summary

Name	Labels and Counts		Total
HATE	Hate Speech	Not Hate Speech	
	$1,430$	$23,353$	$24,783$
HAR	Harassing	Not Harassing	
	$5,285$	$15,075$	$20,360$

In the realm of online learning, little work deals with class imbalance and concept drift each. Previous works of Zhao et al. [2018] have modified the loss function to factor in weights of each class and perform a cost-sensitive online gradient descent. The problem is we need to know the class imbalance severity beforehand and since the cost remains static, it cannot deal with concept drift. Other strategies include oversampling and time decayed class size metric (Wang et al. [2015]) which either require abundant data or are static. For concept drift, a common strategy is to use a sliding window over the data (Hoens et al. [2011]). The drawback of this procedure is it does not handle class imbalance well.

This project seems to be the first attempt to address online learning problems of class imbalance and concept drift in hate speech detection on social networks.

3 Datasets

In this paper, we deal with two datasets to train and evaluate our classifier. First dataset is Davidson et al. [2017]'s publicly available HATE dataset compiled by searching for tweets on `www.Hatebase.org`. Tweets are binarized as "hate speech", or "not". The other dataset is Golbeck et al. [2017]'s HAR harassment dataset, which identifies tweets as "harassing" or "not".[1] Retweets were removed from both datasets. Table 1 provides a numerical summary of both datasets. We also use 300 dimensional GloVe Common Crawl vector embeddings for translating words to vectors that are fed into our model (Pennington et al. [2014]).

4 Methods

4.1 Offline

Let us have $\{(x^t, y^t)\}_{t=1}^n$ denote the training set where x^t is some tweet and y^t is the corresponding class label. We modify the TWEM method (Kshirsagar et al. [2018]) to use fewer parameters in exchange of minimal drop in accuracy. First, we create 300 dimenal embeddings for each word in a given tweet x^t of word length T, so each word $\{w_i\}_{i=1}^T$ in x^t is mapped to $\{m_i\}_{i=1}^T \in \mathbb{R}^{300}$. We can preprocess the tweets before they are encoded with word embeddings but here we found that no significant increase in performance. Following this, we use average pooling operation on $\{m_i\}_{i=1}^T$, which allows us to capture the overall context of the tweet. Denote the output from this operation as a. This representation a is then fed into a 50 node 2-layer MLP followed by ReLU activation to allow for nonlinear representation learning. This is the penultimate layer which passes to a fully connected softmax layer with a class weighted cross entropy whose output is the probability distribution over the class labels.

4.2 Online

In order to adapt the above approach to online stream of tweets, we consider two queues of size L each, one holds the positive examples while the other holds the negative examples (Malialis et al. [2018]). Queue-based resampling stores the most recent example plus $2L - 1$ old ones. We will refer to the proposed algorithm as $Queue_L$. The union of the two queues is then taken at each time step to form the new training set for the classifier. The cost function is given by

$$J = \frac{1}{|q^t|} \sum_{i=1}^{|q^t|} l(y_i, h(x_i)), \tag{1}$$

[1] Available to researchers by emailing jgolbeck@umd.edu.

Table 2: Offline F1 Results

Method	HATE	HAR
Logistic Regression (Kshirsagar et al. [2018])	-	0.68
Logistic Regression (Davidson et al. [2017])	0.93	-
Naive Bayes	0.82	-
GRU Text + Metadata (Founta et al. [2018])	0.89	-
Ours	0.94	0.70

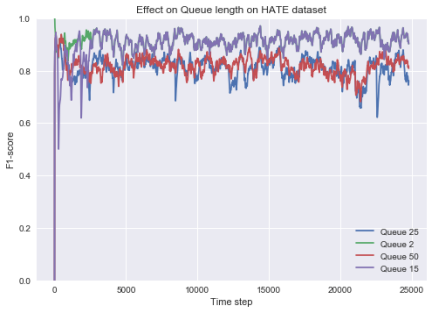

Figure 1: Prequential F1-Score against time step for different queue lengths

where q^t is the union queue with $|q^t| \leq 2L$ and $(x_i, y_i) \in q^t$; $l(u_i, h(x_i))$ is any classification cost function. At each time step, the classifier weights are updated *once* according to (1). We direct the reader to the Appendix for the pseudocode and an illustrative diagram for Queue based Learning algorithm.

The effectiveness of this approach can be attributed to the following highly intuitive reasoning. Having two queues keeps track of examples from both classes. It allows the classifier to "remember" old data from both classes, which can be seen as a form of oversampling to counteract the imbalance since the queue with less frequent examples will have same examples resampled more times than their counterparts. At the same time, we also note that since these queues are of fixed length L, it allows the classifier to "forget" relatively old data and therefore act like a sliding window over the data stream. Fine-tuning the queue size for forgetfulness handles moderate concept drifts present in the stream.

5 Experiments and Results

5.1 Offline

To train the neural network, we perform minimal preprocessing which includes tokenizing the data using Spacy and applying GloVe Common Crawl vector embeddings. Training is performed using gradient descent with ℓ_2 regularization to reduce overfitting. The regularization parameter was chosen to be 0.001 by cross validation on 80–20 training-validation split.

For comparing our model on HATE dataset, we borrow features engineered by Davidson et al. [2017] (which include part-of-speech ngrams, sentiment analysis, and Twitter specific features) and apply logistic regression. In addition, we use the statistics for Naive Bayes baseline and GRU model that uses metadata like popularity, network reciprocity and subscribed lists provided by Founta et al. [2018]. For HAR dataset, Kshirsagar et al. [2018] offer performance for a baseline model trained using logistic regression with character ngrams, word unigrams and TF*IDF.

The results are compiled in Table 2. It is evident that our model requires relatively fewer preprocessing steps and depends only on tweet text, which eliminates dependence on retrieving features like social graphs, sentiment readability, etc. while performing better than these more complex approaches.

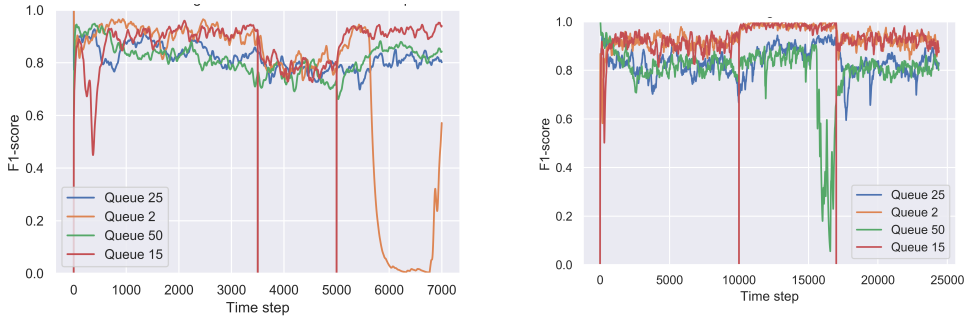

Figure 2: Performance under changing probability distribution. (2a) Probability of positive class increases from 0.05 to 0.15 at time step 3500 and back to 0.05 at 5000th time step. (2b) Probability of positive class decreases from 0.05 to 0.01 at time step 10000 and back to 0.05 at 17000.

5.2 Online

We begin by defining the prequential F1-score as our metric as suggested in Gama et al. [2012] with a fading factor of $\alpha = 0.99$. At all time steps, we compute the prequential F1-score averaged over the most recent 30 runs.

Having established the performance metrics, we inspect the performance on the problem of class imbalance. Our experiments show that without the Queue, the neural network only learns to predict majority class due to severe class imbalance in the online setting despite the use of weighted cross entropy. Henceforth, we focus on performance of $Queue_L$ for $L = \{2, 15, 25, 50\}$ as shown in Figure 1. $Queue_2$ is the quickest to learn and performs the best followed by $Queue_{15}$ which eventually matches the performance of $Queue_2$ at about 3000th time step. Below, we shall experiment a trade-off for queue size that deals with concept drift. It is also seen that $Queue_{25}$ and $Queue_{50}$ lead to excessive oversampling which forces the classifier to "remember" old data for a long time, thereby affecting the decision boundary. Consequently, they are not able to perform as good as the former algorithms.

Next, we experiment with concept drift where we induce a change in the sampling probabilities for the positive (offensive) class at different time steps. These drifts are mild in the sense that the majority class label does not change as a result of the drift. We note that to preserve the performance metric from being affected by the drift, we reset it to 0 at the drift point.

In the first experiment, we increase the probability of positive class from 0.05 to 0.15 after the 3500th tweet and then drop it back to 0.05 at 5000th time step. The results are depicted in Figure (2a). Contrary to the competitive performance of $Queue_2$ under class imbalance, here we see that it takes longer for $Queue_2$ to recover from drift due to very few stored samples while the drift flushes these samples. $Queue_{25}$ and $Queue_{50}$ still suffer from excessive oversampling but are able to recover. $Queue_{15}$ balances the oversampling and recovery the best.

Changing the direction of the drift, in the next experiment, we decrease the probability of positive to 0.01 at 10000th tweet and bring it back up to 0.05 at 17000th tweet as shown in Figure (2b). In this case, $Queue_2$ and $Queue_{15}$ behave with almost the same performance. $Queue_{25}$ does not quite beat its lower size counterparts but still recovers. $Queue_{50}$ has too many samples to affect its training when the class imbalance is in severe phase at 0.01 (cf. performance of $Queue_2$ in the previous case).

6 Conclusion

We explored an offline model that requires a minimal amount of feature engineering and outperforms its counterparts on the task of classifying a tweet as offensive or otherwise. In online setting, we show that the queue based resampling heuristic is able to handle class imbalance and abrupt concept drifts. Future work includes evaluating the effectiveness under other types of drifts (eg. gradual) and the amount of tuning required for queue length. We would also like to see explore methods to deal with noisy labels. Such a method would greatly reduce the amount of tweets to be reviewed by hands.

References

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. Automated hate speech detection and the problem of offensive language. In *Proceedings of the 11th International AAAI Conference on Web and Social Media*, ICWSM '17, pages 512–515, 2017.

Maeve Duggan and Aaron Smith. Social media update 2013. *Pew Internet and American Life project*, 2013.

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

João Gama, Raquel Sebastião, and Pedro Pereira Rodrigues. On evaluating stream learning algorithms. *Machine Learning*, 90:317–346, 2012.

Jennifer Golbeck, Zahra Ashktorab, Rashad O. Banjo, Alexandra Berlinger, Siddharth Bhagwan, Cody Buntain, Paul Cheakalos, Alicia A. Geller, Quint Gergory, Rajesh Kumar Gnanasekaran, Raja Rajan Gunasekaran, Kelly M. Hoffman, Jenny Hottle, Vichita Jienjitlert, Shivika Khare, Ryan Lau, Marianna J. Martindale, Shalmali Naik, Heather L. Nixon, Piyush Ramachandran, Kristine M. Rogers, Lisa Rogers, Meghna Sardana Sarin, Gaurav Shahane, Jayanee Thanki, Priyanka Vengataraman, Zijian Wan, and Derek Michael Wu. A large labeled corpus for online harassment research. In *Proceedings of the 2017 ACM on Web Science Conference*, WebSci '17, pages 229–233, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-4896-6. doi: 10.1145/3091478.3091509. URL `http://doi.acm.org/10.1145/3091478.3091509`.

T. Ryan Hoens, Robi Polikar, and Nitesh V. Chawla. Learning from streaming data with concept drift and imbalance: an overview. *Progress in Artificial Intelligence*, 1:89–101, 2011.

Rohan Kshirsagar, Tyus Cukuvac, Kathleen McKeown, and Susan McGregor. Predictive embeddings for hate speech detection on twitter. *arXiv preprint arXiv:1809.10644*, 2018.

Kleanthis Malialis, Christoforos Panayiotou, and Marios M. Polycarpou. Queue-based resampling for online class imbalance learning. In *ICANN*, 2018.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014. URL `http://www.aclweb.org/anthology/D14-1162`.

Dinghan Shen, Guoyin Wang, Wenlin Wang, Martin Renqiang Min, Qinliang Su, Yizhe Zhang, Ricardo Henao, and Lawrence Carin. On the use of word embeddings alone to represent natural language sequences. 2017.

Shuo Wang, Leandro L. Minku, and Xin Yao. Resampling-based ensemble methods for online class imbalance learning. *IEEE Transactions on Knowledge and Data Engineering*, 27:1356–1368, 2015.

Zeerak Waseem and Dirk Hovy. 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, June 2016. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/N16-2013`.

Peilin Zhao, Yifan Zhang, Min Wu, Steven C. H. Hoi, Mingkui Tan, and Junzhou Huang. Adaptive cost-sensitive online classification. *CoRR*, abs/1804.02246, 2018. URL `http://arxiv.org/abs/1804.02246`.

Appendix

Included in this appendix are the queue learning algorithm and an illustrative diagram on how the process works (Malialis et al. [2018]).

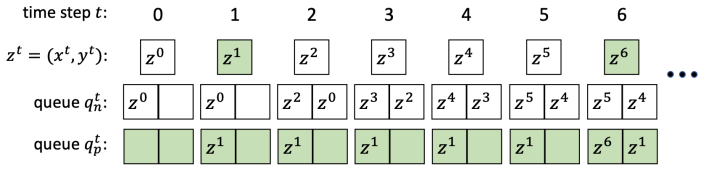

Fig. 1: Example of $Queue_2$ resampling

Algorithm 1 Queue-based Resampling

Input:
maximum length L of each queue
queues (q_p^t, q_n^t) for positive and negative examples
for each time step t **do**
  receive example $x^t \in \mathbb{R}^d$
  predict class $\hat{y}^t \in \{0, 1\}$
  receive true label $y^t \in \{0, 1\}$
  let $z^t = (x^t, y^t)$
  if $y^t == 0$ **then**
    $q_n^t = q_n^{t-1}.append(z^t)$
  else
    $q_p^t = q_p^{t-1}.append(z^t)$
  end if
  let $q^t = q_p^t \cup q_n^t$ be the training set
  calculate cost on q^t using Equation 3
  update classifier
end for