A light-weight model for target detection in offensive language

Tong Xiang, Yuegeng Li

Abstract

We propose a light-weight machine learning method for target detection in offensive language. As more and more user-generated text posted online, problems arise when a large portion of these textual information contains insults. Situations become worse where in many cases abusing languages are pointed directly toward individuals, religions, organizations, etc. In this paper, we present a simple though robust machine learning method to detect targets among offensive speeches. Our approach outperforms many neural methods, including the official bidirectional Long Short Term Memory model, while requires significantly less time as well as resources for training and produces much more explainable decisions than neural methods.

1 Introduction

With technology developing in a incredible speed, more and more people are no longer limited by time and space and get themselves involved with the massive internet. Users usually have unrestricted access to huge amount of online content without paying much effort, which brings them unimaginable benefits along with vast amount of online bullying. Due to Internet's non-restrictive nature and certain countries' legal protection of free speech also including hate speeches[1], some users take advantage of these manners to spread hatred and offensive language online, bringing disgusting social media experience toward other users. To mitigate these harmful influence, some social media platforms such as Facebook and Twitter have announced that they would tried to address these problems.

Traditionally, social media platforms rely on manual filtering as well as user reporting to identify abusing languages. However, not onlt is this non-automated method dramatically slower compared to machine-based solutions, but also introdcuing subjective human bias into it. It is significant to seek a scalable method that could prevent us from being overwhelmed by increasing user-generated content.

Automatical offensive speech detection has been researched throughout the decade. Early researchs tried to address this problem using bag-of-words method along with machine learning classifier like Support Vector Machine. In 2012, Warner and Hirschberg[2] proposed their definition of hate speech and conducted experiments on their own corpus. Their feature set includes unigrams and n-grams, which is reported to be highly predictive. Later in 2016, Waseem and Hovy[3] presented a assumption that knowing some background information about the user of post may be useful for predicting the offensiveness of that post. They conducted experiments with a variety combinations of different features including demographic features, lexical features and geographic features. Mehdad and Tetreault[4] systematically compare character level n-grams with token level n-grams for offensive language detection, and report that character level n-gram features are more predictive than token level n-gram features. Later on, Davidson et al.[5] define a fine-grained scheme of hate speech, and label these samples into three categories. They also make use of machine learning classifier and n-gram bag-of-words features weighted by TF-IDF. The most recent research using machine learning method is the one presented by MacAvaney et al.[6], where they propose a multi-view Support Vector Machine approach that achieves near state-of-the-art performance, without requiring too much computational cost as well as extremly long running time.

Deep learning method has become the most popular approach among natural language processing community in recent years. Different approaches have been proposed to solve the problem of hate speech detection. Gamback and Sikdar[7] introduce convolutional neural network into offensive language classification task in 2017. Their model with best performance is constructed based on semantic information built using word2vec. In the same year, Badjatiya et al.[8] conducted experiments with a few different neural architectures as well as a variety of word embedding methods. Their best model using Long Short Term Memory achieve state-of-the-art over the a benchmark dataset of 16K annotated tweets. Pitsilis et al.[9] tried to address the problem of discerning hateful content in online social media using a novel detection scheme where various Long Short Term Memory models are combined and thus enhance the final result.

As online hate speech usually targets at some specific entities such as individuals, organizations and so on, it is important to know what the target is in offensive languages. Zampieri et al.[10] propose a three-level hierarchical annotation schema that including target detection. Based on the English tweet dataset annotated under the guidance

of above schema, several researches are conducted throughout the SemEval-2019 challenge on Identifying and Categorizing Offensive Language in Social Media[11]. The team jhan014[12] achieve the first place in this challenge using Gated Recurrent Unit as well as Global Vectors[13] trained on 27-billion tweets. Besides, the team NULI[14] ranked 4th place in the challenge relying on fine-tuned Bidirectional Encoder Representation from Transformer[15]. It turns out that neural methods with word embedding trained on enormous corpus dominate this task. Though useful, these methods usually require extremely large corpus and extensively long time to achieve a relatively good performance, thus need numerous human efforts.

Our approach employs a machine learning solution that totally get rid of extensive computational work, and achieve relatively good performance. Our main contributions are: we develop a light-weight machine learning model for target detection which is much more interpretable and requires much less resources compared to neural methods, and results in relatively good performance.

The reminder of this paper is organized as follows: We describe the task and dataset in Section 2. We describe our model with details in Section 3. We show our model's performance along with the performances of official baselines given by [10] in Section 4. Finally we perform an analysis in Section 5.

2 Task definition and dataset

Class	Training	Testing
TIN	3876	213
UNT	524	27

Table 1: Data distribution for task b in OLID.

Zampieri et al.[10] propose a three-level hierarchical annotation schema for identifying and categorizing offensive language in social media. Beyond that, they collect and annotate a dataset called Offensive Language Identification Dataset(OLID) which was collected from Twitter API and annotated using proposed annotation schema. They defined three sub-tasks corresponding to the three-levels in their annotation: a) Offensive language identification b) Categorization of offensive language c) Offensive language target identification. The task for this paper is from sub-task b of above definition, where the goal is to predict the type of offense. In this task, only offensive posts are considered, and the only two categories in this task are as follow:

- Targeted Insult(TIN): Posts containing an insult/threat to an individual, group, or others;
- Untargeted Insult(UNT): Posts containing non-targeted profanity and swearing. Posts with general profanity are not targeted, but they contain non-acceptable language.

The distribution of the labels in OLID is shown in Table 1(For simplicity we only show the data for task b). For task b, it only contains 4400 samples for training set and 240 samples for testing set. We can easily observe that this dataset has a very imbalanced label distribution for task b, and too little testing set to conduct a fair experiment. Though, this twisted dataset reflect some phenomena in real world scenario, where the amount of targeted offensive posts are much larger than the amount of posts that don't contain any profanity word.

3 Methodology

3.1 Pre-processing

We apply a pre-processing pipeline to the dataset. The pipeline contains the following modules:

- data cleaning
- tokenizing
- removing stop-words
- lemmatizing
- Stemming
- part-of-speech tagging
- name entity recognizing

Each of our models usually only makes use of part of these modules. The data cleaning module filters out URLs, usernames, and unimportant suffixes. Initially we thought that usernames inside posts might somewhat indicate the existence of targets. But the results report that in most cases duplicate usernames are not distinguishable enough, and should be removed or reduced before further processing. We design two different procedures for data cleaning, where the first procedure removes all the usernames in a sentences and the second one reduces duplicate usernames into a single one. We conduct experiments with distinct procedures and find out that the later one contributes more to the best result. Tokenizing module breaks the sentence into small chunks called tokens, while lower-casing all the words at the same time. We have tried two tokenizers, including the general tokenizer and the Twitter-specific tokenizer, both of which are implemented inside NLTK[16]. Removing stop-words module removes the words that appear too many times in corpus and thus not vital enough. However, the results show that removing stop words will hurt the performance of our models and should not be applied to our textual data. Lemmatizing module brings the words that have different inflected forms into the same one. Stemming module reduce inflected words into their word stem, base or root form. The results imply that stemming will also hurt our models' performance. Part-of-speech tagging modules assign pre-defined part-of-speech tags to words in different posts. Here we have done experiments with two different kind of part-of-speech tagger, including NLTK tagger and spaCy tagger. These two taggers follow totally distinct tagging scheme, thus introduce variance into our system. Name entity recognizing module assigns pre-defined name entity tags to words.

3.2 Vectorization and classification

After pre-processing, we will generate sentence vectors using the features produced by pre-processing pipeline. We combine a suite of features together, including word-level unigram, part-of-speech tags and name entity tags weighted by TF-IDF.

For classifier, we apply a number of machine learning classifiers based on Scikit-learn[17]. Among all the classifiers we have tried, Logistic Regression with L2-regularised outperforms other classifiers in the first two rounds of experiment. Our best model makes use of syntactic features weighted by TF-IDF, and apply Logistic Regression with L2-regularization for classification.

4 Experiment results

4.1 Experiment settings

The experiments were run in three stages. Firstly we tried several machine learning classifiers including Logistic Regression, Support Vector Classifier and Random Forest Classifier. Experiment results reveal that Logistic Regression performs significantly better than all other classifiers. In second stage we experiment with different regularization methods and L2-regularization stands out. In the final stage we explore different selections of features and filter out the best feature combination. For all the models that we have experimented, we performed a 3-fold cross-validation and used macro F1-score as the scoring metric while training on the training set. All the selected models were optimised on the training set, and eventually evaluated on the testing set.

4.2 Feature engineering

The initial two stages of experiments have shown us that Logistic Regression with L2-regularised is undoubtedly the best classifier among the choices. In the third stage we conducted an ablation analysis on the features. The ablation analysis showed that models merely using surface features were not likely to perform well. Observation revealed that part-of-speech tags as well as name entity tags would generally benefit the performance, where the later one's contribution was not dominant. Further exploration showed that removing stop-words and stemming cause a reduction in the overall performance, thus we decided to drop these two features in further experiments. By doing these comparison, we made our final decision, where we used token-level unigram as fundamental feature, combined with part-of-speech tags as well as name entity tags features. Furthermore, we choose to use the Twitter-specific tokenizer implemented inside NLTK as well the spaCy part-of-speech tagger after carefully consideration.

4.3 Results

This task utilize a highly imbalanced dataset, with the number of positive samples(TIN) dwarfing the negative samples(UNT). We compared our model with the official baselines given by [10] in Table 2. The confusion matrix of our

	TIN			UNT			Weight Average			
Model	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**F1 Macro**
SVM	0.91	0.99	0.95	0.67	0.22	0.33	0.88	0.90	0.88	0.64
BiLSTM	0.95	0.83	0.88	0.32	0.63	0.42	0.88	0.81	0.83	0.66
CNN	0.94	0.90	0.92	0.32	0.63	0.42	0.88	0.86	0.87	0.69
LR	0.92	0.98	0.95	0.67	0.30	0.41	0.88	0.90	0.89	0.68
All TIN	0.89	1,00	0.94	-	0.00	0.00	0.79	0.89	0.83	0.47
All UNT	-	0.00	0.00	0.11	1.00	0.20	0.01	0.11	0.02	0.10

Table 2: Results for target detection in offensive language compared with official baselines.

final model is shown in Figure 1. Our model's performance is equal to the model that ranked 14th place out of 76 teams in the challenge.

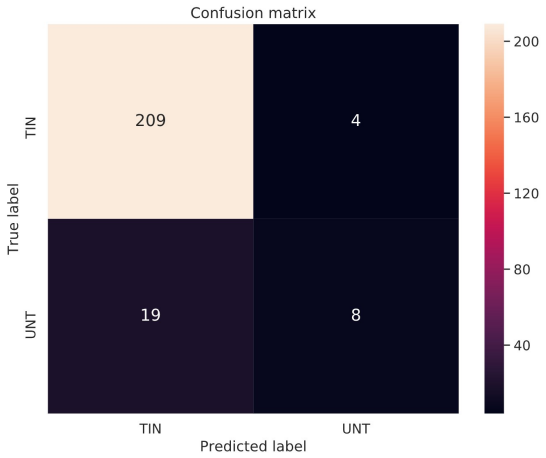

Figure 1: Confusion matrix for final model.

5 Analysis

As we can easily observe from Table 2, all of the models can do fairly good on predicting targeted insult, while performing dramatically bad on the opposite. Not being able to learn enough information and patterns from samples, models built based on this ill-defined dataset performs even worse than randomly guessing on prediction of untargeted insult. Another thing that strongly illness of dataset is the sample mislabelling. For example, *@USER @USER Still no excuse... Where TF are her parents??? They are using him & he is using her.* is considered as untargeted insult, which apparently has target *parent* in it. Nevertheless, our model shows potential in this task. By only making use of lexical and syntactic features weighted by TF-IDF, our models produce similar output compared with other neural methods without consuming too much resources.

During pre-processing, we have tried different choices for each of the modules. In data cleaning module, the result reports that reducing duplicate usernames into one outperforms directly removing all usernames. The intuition is gained by observing the original training set: both targeted and untargeted insults contain considerable amount of usernames inside, thus username might not be distinguishable enough; but completely deleting them from the sentences seems too arbitrary as username indeed indicates some information in some cases. Our result also report that removing stop-words will hurt the performance. This counter-intuitive phenomenon may comes from the assumption that stop-word, usually a preposition, plays a vital role in sentence structure when there is target inside it. They can be viewed as syntactic features, which can further indicate much information. Stemming is harmful for our model simply because stemming would possibly turns words that have semantic meaning into the same stemmed form, thus would further lead to ambiguity.

Our experiments show that Logistic regression stands out among machine learning classifiers. One of the reason could be that this simple though robust model along with reasonable regularization does not overfit the training set too much compared with other classifiers. As the task is a binary classification task, it turns out that model tend to do a better job without complicated prior assumption.

References

[1] Djuric, Nemanja, et al. "Hate speech detection with comment embeddings." *Proceedings of the 24th international conference on world wide web.* ACM, 2015.

[2] Warner, William, and Julia Hirschberg. "Detecting hate speech on the world wide web." *Proceedings of the second workshop on language in social media. Association for Computational Linguistics,* 2012.

[3] Waseem, Zeerak, and Dirk Hovy. "Hateful symbols or hateful people? predictive features for hate speech detection on twitter." *Proceedings of the NAACL student research workshop.* 2016.

[4] Mehdad, Yashar, and Joel Tetreault. "Do characters abuse more than words?." *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue.* 2016.

[5] Davidson, Thomas, et al. "Automated hate speech detection and the problem of offensive language." *Eleventh international aaai conference on web and social media.* 2017.

[6] MacAvaney, Sean, et al. "Hate speech detection: Challenges and solutions." *PloS one* 14.8 (2019).

[7] Gambäck, Björn, and Utpal Kumar Sikdar. "Using convolutional neural networks to classify hate-speech." *Proceedings of the first workshop on abusive language online.* 2017.

[8] Badjatiya, Pinkesh, et al. "Deep learning for hate speech detection in tweets." *Proceedings of the 26th International Conference on World Wide Web Companion.* International World Wide Web Conferences Steering Committee, 2017.

[9] Pitsilis, Georgios K., Heri Ramampiaro, and Helge Langseth. "Detecting offensive language in tweets using deep learning." *arXiv preprint arXiv:1801.04433* (2018).

[10] Zampieri, Marcos, et al. "Predicting the Type and Target of Offensive Posts in Social Media." *arXiv preprint arXiv:1902.09666* (2019).

[11] Zampieri, Marcos, et al. "Semeval-2019 task 6: Identifying and categorizing offensive language in social media (offenseval)." *arXiv preprint arXiv:1903.08983* (2019).

[12] Han, Jiahui, Shengtan Wu, and Xinyu Liu. "jhan014 at SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media." *Proceedings of the 13th International Workshop on Semantic Evaluation.* 2019.

[13] Pennington, Jeffrey, Richard Socher, and Christopher Manning. "Glove: Global vectors for word representation." *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP).* 2014.

[14] Liu, Ping, Wen Li, and Liang Zou. "Nuli at semeval-2019 task 6: Transfer learning for offensive language detection using bidirectional transformers." *Proceedings of the 13th International Workshop on Semantic Evaluation.* 2019.

[15] Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." *arXiv preprint arXiv:1810.04805* (2018).

[16] Loper, Edward, and Steven Bird. "NLTK: the natural language toolkit." *arXiv preprint cs/0205028* (2002).

[17] Pedregosa, Fabian, et al. "Scikit-learn: Machine learning in Python." *Journal of machine learning research* 12.Oct (2011): 2825-2830.