Offensive Language Detection using Transfer Learning

Shuo Zhang
janezhangshuo@hotmail.com

Zhexuan Xu
zhexuanx@gmail.com

Abstract

With the increasing impact of social media, offensive language generated by users becoming a rising phenomenon and has drawn significant attention from industry as well as academia. In this work, we utilize the Offensive Language Identification Dataset (Zampieri et al., 2019a) which contains over 14,000 English tweets, to develop and compare several offensive language detection models, including classical machine learning classifiers and deep neural networks. We specifically focus on transfer learning approach based on BERT (Bidirectional Encode Represnetations from Transformer), to learn efficiently and effectively from limited labeled data with a pre-trained model. We show that BERT based transfer learning model significantly outperforms classical machine learning models and other deep learning architectures like CNN, LSTM and BiLSTM+CNN ensemble networks. We also did detailed data preprocessing according to social media language behaviors.

1 Introduction

With the fast growth in social media and the increasing propagation of abusive language, offensive language detection has drawn significant attention from governments, companies, and researchers. However, it faces a few challenges. First, diversity of offensive language makes it difficult to catch features for different types, such as racism, sexism, and profanity. Second, users proactively create variations of offensive speech (e.g. ”@$$”) to thwart existing detection systems, which may require systems to update overtime, or algorithms to be able to learn subword information and make inference on a new word. Lastly, some offensive speeches may not even use offensive or profane words and further require cultural and background knowledge, which are sometimes confused with sarcasm. These all push offensive language detection algorithms to develop from spotting bad words to deeper natural language understanding.

In the history of offensive language detection, classical machine learning methods and DNN were largely applied for classification. However, comparatively small public dataset of offensive language detection makes it difficult to train complex models. Therefore, some latest works fine-tuned a task-specific embedding using DNN on top of traditional word embedding such as Glove (Pennington et al., 2014) and word2vec (Mikolov et al., 2013). Given the rise of contextual word embedding like ELMo (Peters et al., 2018), GPT (Radford et al., 2019) and BERT (Devlin et al., 2018), we hypothesize that transfer learning using Bidirectional Transformers (BERT) can improve the model performance of offensive language detection.

In this work, we implement BERT-based language detection model. We also implement and compare several baseline models, including classical machine learning algorithms (logistic regression, SVM, random forest) with sparse and dense feature vectors, LSTM, CNN, and ensemble deep learninig models. We demonstrate that our BERT-based model significantly outperforms all the baseline models on offensive language detection task. We provide the source code and detailed experiment results for above models, as well as the dataset at

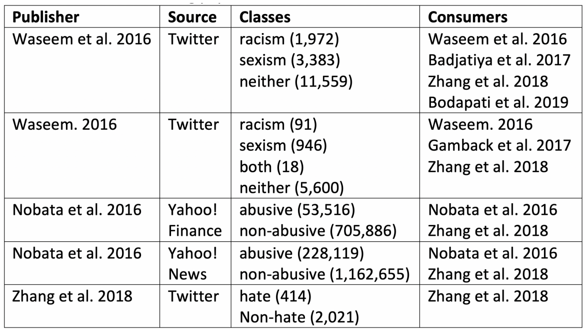
https://github.com/JaneShuoZhang/OffensiveLanguageDetection/tree/master/src

2 Related Work

2.1 Terminology and Datasets

The definition of “offensive language” is not exactly the same in all literature, and some may use similar terms like “hate speech”, “abusive language". From the dataset each work was using, we can tell their definition of "offensive language" and how they did detection through classification. For example, for articles using 16k annotated tweets dataset (Waseem and Hovy, 2016), they were specifically detecting racism and sexism, as their data labels are racism, sexism, and neither.

There isn't a benchmark dataset for offensive language detection, and the large majority of existing works were evaluated on privately collected datasets, often for different problems. Figure 1 lists some publicly available hate speech datasets, including some papers that used them in "consumer" column.



Publisher	Source	Classes	Consumers
Waseem et al. 2016	Twitter	racism (1,972) sexism (3,383) neither (11,559)	Waseem et al. 2016 Badjatiya et al. 2017 Zhang et al. 2018 Bodapati et al. 2019
Waseem. 2016	Twitter	racism (91) sexism (946) both (18) neither (5,600)	Waseem. 2016 Gamback et al. 2017 Zhang et al. 2018
Nobata et al. 2016	Yahoo! Finance	abusive (53,516) non-abusive (705,886)	Nobata et al. 2016 Zhang et al. 2018
Nobata et al. 2016	Yahoo! News	abusive (228,119) non-abusive (1,162,655)	Nobata et al. 2016 Zhang et al. 2018
Zhang et al. 2018	Twitter	hate (414) Non-hate (2,021)	Zhang et al. 2018

Figure 1: Public datasets

2.2 Methods

In general, the methods can be divided into classic methods and deep learning based methods depending on whether there is an automated feature learning process.

2.2.1 Classic Methods

Features of classic methods are manually encoded into feature vectors. There are several types of features from above literature. (1) *Simple surface features* such as bag of words, character and word n-grams. Character n-grams has been proven to be more effective than word n-grams (Nobata et al., 2016) (Schmidt and Wiegand, 2017). (2) *Syntactic features* such as POS n-grams to capture long-range dependencies dependency (Nobata et al., 2016) (Davidson et al., 2017). (3) *Distributional semantic features*, such as word embedding and comment embedding, that learn low-dimensional dense feature vectors of word/sentence meaning, like GloVe and word2vec (Nobata et al., 2016) (Badjatiya et al., 2017). (4) *Lexical resources* containing lists of negative words. Early methods were heavily based on spotting bad words. (5) *Sentiment analysis* provides polarity information. Positive sentiment and higher readability scores are more likely to belong to non-offensive class (Davidson et al., 2017). (6) *Multimodal information* from images (Yang et al., 2019). (7) *Knowledge-Based features* to bring stereotypical concepts, which could help detecting domain-specific offensive language (Schmidt and Wiegand, 2017). (8) *Metadata* such as gender and history of users.

A usual baseline model for many papers is character n-gram (up to length of 4) + Logistic Regression from Waseem's work in 2016 (Waseem and Hovy, 2016). Besides, feature selection such as Logistic Regression + L1 regularization can be a very powerful technique to improve performance of classic methods (Zhang et al., 2018). In this work, we will also implement this model on our dataset as a baseline.

2.2.2 Deep Learning Based Methods

Deep learning based methods use Neural Networks to learn abstract feature representations from input data to boost classification. Note some works use DNN to learn word or text embeddings as features and feed into another classifier like GBDT for classification. This is proven to be an effective way to train a task-specific embedding and get low-dimensional dense features (Badjatiya et al., 2017).

CNN is well known as an effective network for feature extraction, and when applied to NLP, it extracts character n-gram features or special phrases. RNN, especially Gated-RNN is powerful in modelling sequence input by learning long-range word or character dependencies. While some works applied CNN or LSTM independently (Badjatiya et al., 2017) (Gambäck and Sikdar, 2017), Zhang *et al.* started to combine CNN and GRU to utilize the power of these two (Zhang et al., 2018). CNN layers was first applied to extract n-gram features, then the feature vectors will be max-pooled and fed into GRU while still keeping sequence order. Regularization techniques such as adding a drop-out layer right after the embedding layer, and applying global max pooling layer, were shown to improve the performance.

While most DNN methods focus on the structure of classifier, Bodapati *et al.* dived into the importance of using finer units of word to learn more robust representations in neural networks (Bodapati et al., 2019). They compared the effectiveness of end-to-end character-based models, word + character embedding models, byte pair encoding (BPE), subword models, with pure word-based models. It is shown that pretraining a BERT BPE model on a large general corpus then using this for encoding input text can improve significantly for all word-based models.

3 Data

We use the dataset of competition SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media (Zampieri et al., 2019b). Offensive Language Identification Dataset (OLID) (Zampieri et al., 2019a) was collected from Twitter API by searching certain keyword set. The keywords include some unbiased targeted phrase such as 'she is', 'he is' and 'you are' which have high proportional offensive tweets. The distribution of offensive tweets is controlled around 30% by using different sampling methods.

There are three subtasks in this competition: (1) detecting if a post is offensive (OFF) or not (NOT); (2) identifying the offense type of an offensive post as targeted insult (TIN), targeted threat (TTH), or untargeted (UNT); (3) for a post labeled as TIN/TTH in second subtask, identifying the target of offense as individual (IND), group of people (GRP), organization of entity (ORG), or other (OTH). Due to the time limitation, we will try on the **first subtask** in this work.

All the three subtasks share the same dataset, with different labels. The dataset was released as three parts: start kit, training dataset and testing dataset. The data distribution is showed in Figure 2. There are in total 13,560 training tweets (including start kit), and 900 testing tweets.

Class	StartKit	Training	Testing
NOT	243	8840	620
OFF	77	4400	280
TIN	38	3876	213
UNT	39	524	27
IND	30	2407	100
GRP	7	1074	78
OTH	2	395	35

Figure 2: Data Distribution

4 Models

4.1 Data Preprocessing

We utilize TweetTokenizer from NLTK as tokenizer, and add several preprocessing steps before feeding into the tokenizer. Overall, our data preprocessing includes following steps in order:

– Replace Emoji by substituted phrase; [1]
– Convert to lowercase;
– Expand contractions, e.g. "don't" to "do not";
– Remove URLs, twitter handles, and RT. Only remain words, numbers, '!', '?' and '.';
– Remove 'url' token;
– Word segmentation, including hashtags; [2]
– Tokenization;
– Spelling corrections - We use open-source Sympell which uses the Damerau-Levenshtein distance to find the closest correct spelling for any misspelled words. We chose the edit distance to be 3; [3]
– Lemmatize using WordNetLemmatizer from NLTK.

Below is an example tweet before preprocessing. Since the emoji cannot be displayed here, we use emoji[] to represent initial emojis:

> @USER @USER Go home you're drunk!!! @USER #MAGA #Trump2020 emoji[coming fist] emoji[unite state] emoji[coming fist] URL

And the tweet after preprocessing:

> go home you be drink ! ! ! magna trump 2020 oncoming fist unite state oncoming fist

4.2 Baseline Models

We have as baselines both classical machine learning models and deep neural network models, which can also exam the improvement by deep structured models. We use GloVe (Pennington et al., 2014) as word embedding and tune on different dimensions. The 50d, 100d, 200d embeddings were trained on 2B tweets, and the 300d embedding was trained on commonly crawled 42B tokens to get a relatively large vocabulary size.

For classic models, we first choose character n-grams (up to four-grams) + logistic regression (Waseem and Hovy, 2016) because it is a frequently used baseline by many previous works. We also implemented GloVe + logistic regression/random forest/SVM. By comparing n-gram + logistic regression and GloVe + logistic regression, we can

[1]https://github.com/carpedm20/emoji
[2]https://github.com/grantjenks/python-wordsegment
[3]https://github.com/wolfgarbe/SymSpell

compare the sparse manually crafted feature with dense feature representation. By comparing different machine learning classifiers, we can see how different type of classifier performs, linear, non-linear, tree-based.

For deep neural classifiers, we implement LSTM, CNN, and BiLSTM + CNN as a representation of ensemble deep learning model. RNN is designed to ”remember” previously read values and LSTM is good at extracting sequence information. For classification, we process tweets by LSTM layer and feed the final hidden state into a linear + softmax layer to get the predicted probability. Initially designed for image recognition, CNN is well-known for feature extraction, and has subsequently been shown to be effective for NLP. In this paper, we implemented the CNN structure illustrated by Kim, Y. (Kim, 2014) as shown in Figure 3.

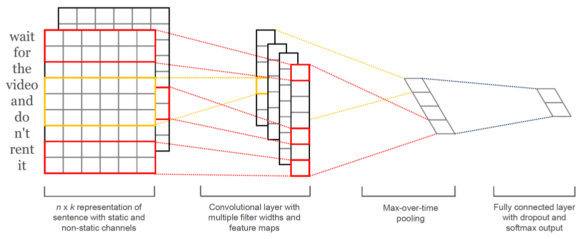

wait
for
the
video
and
do
n't
rent
it
n x *k* representation of sentence with static and non-static channels
Convolutional layer with multiple filter widths and feature maps
Max-over-time pooling
Fully connected layer with dropout and softmax output

Figure 3: Kim, Y. (2014). CNN for Sentence Classification

While trying CNN and LSTM seperately to reveal advantages of two kinds of deep networks, we implement BiLSTM + CNN to combine the strength of the two. We use the structure shown in Figure 4 (Sosa, 2017). Intuitively, the Bi-LSTM layer generates a new encoding for each token containing context information in the sentence. Then the hidden states of the LSTM layer is then fed into CNN layer to further extract local features, same as in the above CNN model.

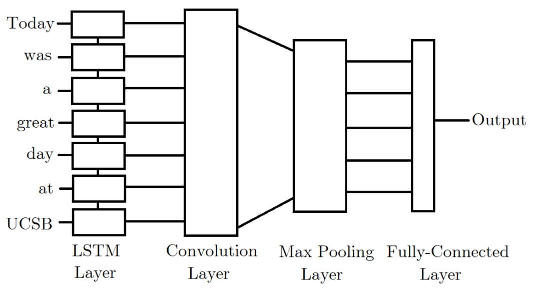


Figure 4: Sosa, P. (2017). Bi-LSTM-CNN Model.

4.3 BERT Fine-tuning Model

We use a simple BERT based model architecture in order to see the pure effect of transformer pre-trained contextual word embedding, as shown in Figure 5. In this architecture, the output for [CLS] token of 12 transformer encoders with a vector of 768, is used to feed into a linear + softmax layer to provide probabilities of final outputs.

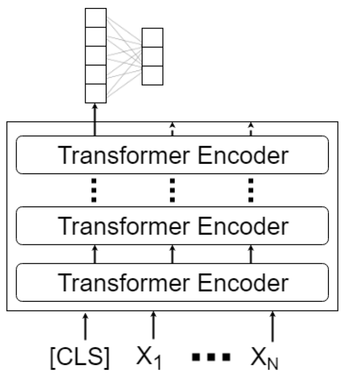


Figure 5: BERT for classification. An example for three classes. (Mozafari et al., 2019)

5 Experiment Results

Results for all models are shown in Table 1. Metrics include accuracy, macro precision, macro recall, macro F1 and F1 for offensive class. We also show the the optimal hyperparameter settings for each model after tuning in Table 2.

For classical models, logistic regression using GloVe achieves the best performance, almost as good as LSTM model. By comparing logistic regression based on n-grams and GloVe, we can see the effectiveness of dense feature representation which is rich in semantic information and faster during model training. The non-linear classifiers SVC and random forest has lower scores compared with linear classifiers, even worse than logistic regression + ngrams. From the hyperparameter tuning and cross-validation, we saw that even for the best hyperparameter setting, there is still strong overfitting for these two non-linear classifiers, SVC having 0.975 training macro-f1 and 0.676 validation macro-f1, and random forest having 0.999 training macro-f1 and 0.595 validation macro-f1.

For deep learning models, their macro-f1 scores all outperform classical models. LSTM performs worse than CNN and ensemble model, and even has a lower accuracy than logistic regression. One possible reason is that we use the final hidden state for classification which squeezes all the information

Method	Models	Accuracy	Precision	Recall	Macro F1	OFF F1
Classic Methods	LR + Char n-gram	0.7756	0.7203	0.7103	0.7148	0.5832
	LR + GloVe	0.8035	0.7625	0.7220	0.7368	0.6042
	SVC + GloVe	0.7221	0.6654	0.6808	0.6710	0.5413
	RF + GloVe	0.7628	0.7412	0.6056	0.6121	0.3704
Deep Learning	CNN + GloVe	0.8140	0.7729	0.7471	0.7578	0.6413
	LSTM + GloVe	0.7895	0.7384	0.7378	0.7381	0.6221
	BiLSTM + GloVe	0.8314	0.8091	0.7490	0.7697	0.6507
	BERT	**0.8628**	0.8347	0.8167	**0.8249**	**0.7435**

Table 1: Comparison of Various Method (Embedding Size = 50 or 300 for GloVe)

Models	Optimal Hyperparameter Setting
LR + Char n-gram	penalty=l1, solver=liblinear
LR + GloVe	embed dim=300, penalty=l1, solver=saga
SVC + GloVe	embed dim=300, regularization=0.8, kernel=poly
RF + GloVe	embed dim=300, number of estimators=100
CNN + GloVe	embed dim=50, out channels=30, max iter=10, dropout=0.1
LSTM + GloVe	embed dim=300, batch size=64, hidden dim=64, eta=0.001
BiLSTM + GloVe	embed dim=50, batch size=1024, hidden dim=50, out channels=30, max iter=10
BERT	max iter=3, batch size = 32, optimizer=Adam, eta=2e-05, eps=1e-8

Table 2: Optimal Hyperparameter

of a tweet into only one vector, and our sequences were controlled to have a maximal length of 128, which is still expensive and hard for LSTM to remember long-distance dependency. In comparison, although CNN may have a shallower understanding of the whole sentence, it is good at extract local patterns, which in offensive language situation, it is able to extract keywords and negative phrases, and most offensive tweets tend to contain negative words or phrases targeting at certain group of people. However, for offensive tweets that do not contain any abusive word, CNN is less capable of detecting them. For example, for the tweet below, CNN fails to label it as offensive class:

> #SerenaWilliams is so full of herself...she is just as painful to watch as to listen to...

By combining the two deep neural networks, the BiLSTM+CNN model significantly improves CNN's macro-f1 score by 1.6%. This model keeps the benefit of CNN model above, besides, instead of using the final hidden state of LSTM model, it uses every hidden state representing each word, together with the contextual information before and after. The BiLSTM layer acts as a re-embedding layer, which infuses the original GloVe word embedding with more semantic information given a grounded language environment. As expected, the model has better performance than pure CNN model.

Without extra down-stream layers like CNN and RNN, the regular BERT based fine-tuning model significantly outperforms other models, and the f1 score for offensive class boosts almost 0.1 compared with other models, which indicates that BERT model is specifically good at detecting offensive class. The confusion matrix of BERT prediction is in Figure 6. The Transformer encoder structure is able to catch the bi-directional information as BiLSTM, and on top of that, it directly uses attentions which makes it much easier to extract long-distance dependency. Multi-heads also enable the model to capture different aspects of syntactic and semantic information. Lastly, since the relatively small dataset makes it difficult to train complex models, we can benefit a lot from fine-tuning a task-specific embedding using pre-trained embeddings like BERT. Below is an example tweet that only BERT succeeds to detect it as offensive language:

> @USER Do you get the feeling he is kissing @USER behind so he can humiliate him later?

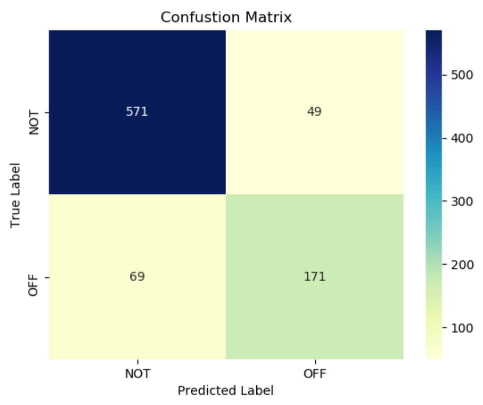


Figure 6: Confusion Matrix of BERT Model.

6 Conclusion

From the experiments, deep learning models has better performance than classical machine learning classifiers in offensive language detection task. And transfer learning model based on BERT gives significant improvement on all other models. By comparing GloVe and manually crafted sparse feature, we can also see the benefits of dense feature representation that pre-learned syntactic and semantic information.

We haven't got time to try upgraded BERT-based architectures. Instead of using a fully connected network without hidden layer, more robust classifier can be achieved by inserting nonlinear layers like CNN or BiLSTM layer before final softmax activation (Liu et al., 2019). It would be interesting to try these more complex models. Also, it would be better to try all the models on more than one dataset to see the robustness of different models, in order to avoid overfitting.

It would be very helpful to collect an even larger dataset for offensive language detection, including metadata like user information, even multimodal information from images. A large benchmark dataset enables fare comparisons between algorithms.

Most works have so far focused on offensive language detection in English. Due to different culture background and probably different need from government, it would be a good application to detect offensive language for other languages.

References

Pinkesh Badjatiya, Shashank Gupta, Manish Gupta, and Vasudeva Varma. 2017. Deep learning for hate speech detection in tweets. In *Proceedings of the 26th International Conference on World Wide Web Companion*, pages 759–760.

Sravan Babu Bodapati, Spandana Gella, Kasturi Bhattacharjee, and Yaser Al-Onaizan. 2019. Neural word decomposition models for abusive language detection. *arXiv preprint arXiv:1910.01043*.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. In *Eleventh international aaai conference on web and social media*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Björn Gambäck and Utpal Kumar Sikdar. 2017. Using convolutional neural networks to classify hatespeech. In *Proceedings of the first workshop on abusive language online*, pages 85–90.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Ping Liu, Wen Li, and Liang Zou. 2019. Nuli at semeval-2019 task 6: transfer learning for offensive language detection using bidirectional transformers. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 87–91.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Marzieh Mozafari, Reza Farahbakhsh, and Noel Crespi. 2019. A bert-based transfer learning approach for hate speech detection in online social media. In *International Conference on Complex Networks and Their Applications*, pages 928–940. Springer.

Chikashi Nobata, Joel Tetreault, Achint Thomas, Yashar Mehdad, and Yi Chang. 2016. Abusive language detection in online user content. In *Proceedings of the 25th international conference on world wide web*, pages 145–153.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9.

Anna Schmidt and Michael Wiegand. 2017. A survey on hate speech detection using natural language processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media*, pages 1–10.

Pedro M Sosa. 2017. Twitter sentiment analysis using combined lstm-cnn models. *Eprint Arxiv*.

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.

Fan Yang, Xiaochang Peng, Gargi Ghosh, Reshef Shilon, Hao Ma, Eider Moore, and Goran Predovic. 2019. Exploring deep multimodal fusion of text and photo for hate speech classification. In *Proceedings of the Third Workshop on Abusive Language Online*, pages 11–18.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019a. Predicting the type and target of offensive posts in social media. *arXiv preprint arXiv:1902.09666*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019b. Semeval-2019 task 6: Identifying and categorizing offensive language in social media (offenseval). *arXiv preprint arXiv:1903.08983*.

Ziqi Zhang, David Robinson, and Jonathan Tepper. 2018. Detecting hate speech on twitter using a convolution-gru based deep neural network. In *European semantic web conference*, pages 745–760. Springer.