Toxicity Classification in Online Comments Using Machine Learning

Zhaoning Lu

1. Introduction

As general social media and the internet rapidly growing in recent years, toxic comments have become one of the most urgent problems to be solved. Toxic words are usually connected with rude, violent, impolite, and disrespectful, which brings no positive effect on society. One way to control such phenomenon is by developing machine-learning or natural-language-processing algorithms to judge the toxicity of a sentence automatically. The study of analyzing human language has gained massive focus these years.

Under such backgrounds, this project aims to detect and classify toxic comments using machine learning. Three types of data have been used, which are TFIDF, embedded vector, and raw comment. We also used 24 identity labels from five categories to aid the topic analysis. The method and experiment results will be detailed. This paper will also dig into how different categories impact prediction accuracy. 'Christian', 'Muslim', 'Female', 'Homosexual gay or lesbian', and 'Male' categories will be the focus. The dataset used is from [1]

2. Literature Review

As the results provided by [2], LSTM (long-short-term memory) gives a huge improvement compared with the Naïve Bayes model. The true positive rate using LSTM model is about 20 percent higher. RNN (recurrent neural network) has also been tested in their research. However, it shows that the nature of RNN model will treat the early words in a sentence as early nodes, which suffers a vanishing gradient problem. Resulting in the first several words in a long sentence cannot contribute much to the final prediction. And the words at the end of the sentence have more impact. This is intuitively wrong because a toxic word can appear at any position of a sentence with almost equal probability. Therefore, [2] decided to implement LSTM model, which is more capable of dealing with long-term information.

In terms of feature engineering, the method adopted by [3] is to remove 'HTML' element in the dataset. One interesting finding from their research is the effect of 'stop words'. As reported, the model gives better performance if the 'stop words' is remained in the sentences. The reason may because 'stop words' are frequently used in toxic comments to refer to people.

3. Method

3.1. Dataset Analysis

To get a better understanding of the dataset. The nltk library has been used to show the statistical results. An additional dictionary of hate words original from [4] is also used for analysis, which contains the most common 1493 hate words in English.

As shown in figure [3.1.1], the overall top50 hate words in the training dataset appear frequently. The most common hate words are 'gay', 'sexual', 'racist', etc. However, the mis-spelling and lemmatization problem is not considered.

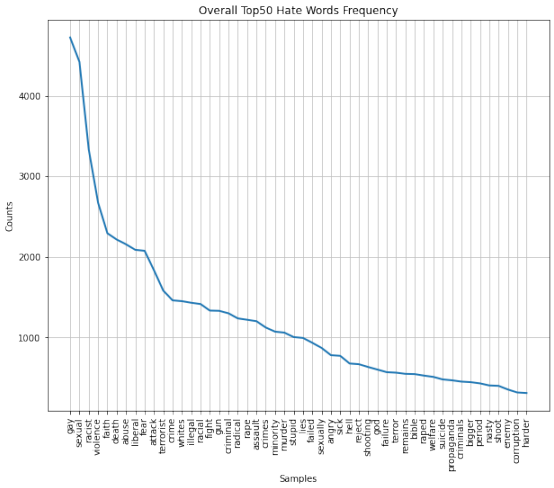

Figure [3.1.1]

The training dataset contains 140000 samples in total, which are labeled as toxic or non-toxic. Among them, there are 22486 toxic samples (16.1%) and 117514 non-toxic samples (83.9%). The dataset is severely imbalanced. It is intuitively true, as non-toxic comments should be dominant in the real world. To reflect the fair situation, 2000 comments are randomly sampled from toxic and non-toxic groups separately. Then the hate words frequency check is applied to both groups. The results are shown below.

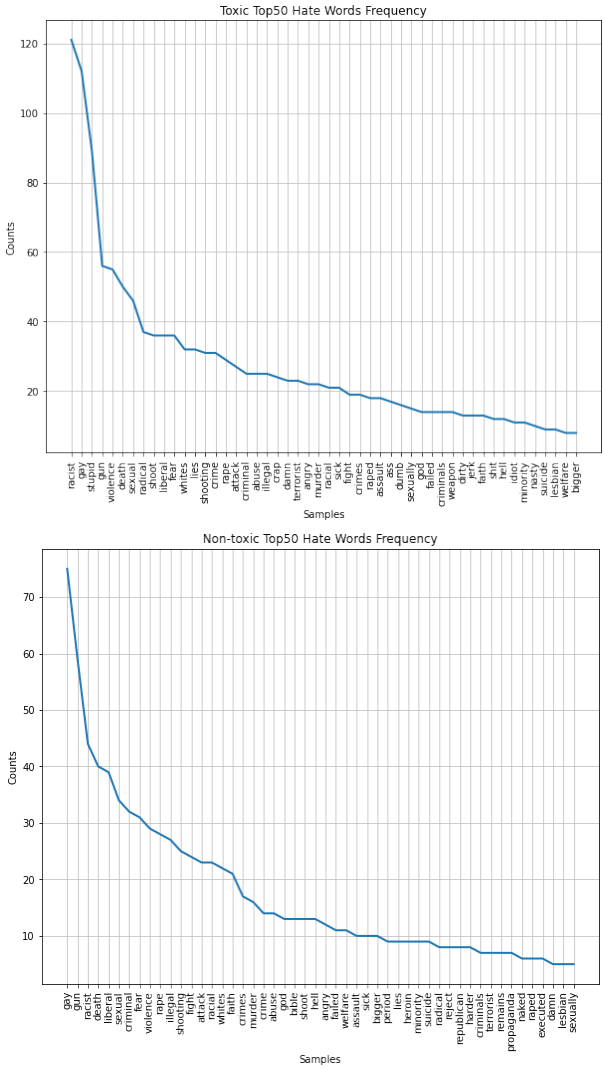

Figure [3.1.2]

Based on the graph, we can observe that hateful words appear in both toxic and non-toxic comments. For instance, “This is a great story. Man. I wonder if the person who yelled "shut the fuck up!" at him ever heard it.” is labeled as non-toxic. Therefore, simply detecting hate words and using a threshold to classify toxic comments should not work in most cases.

We also examined the distribution of the comment length on 2000 randomly sampled data for each class. The result shows that toxic comments usually have shorter lengths compared with non-toxic comments. In other words, short comments are more likely to be toxic.

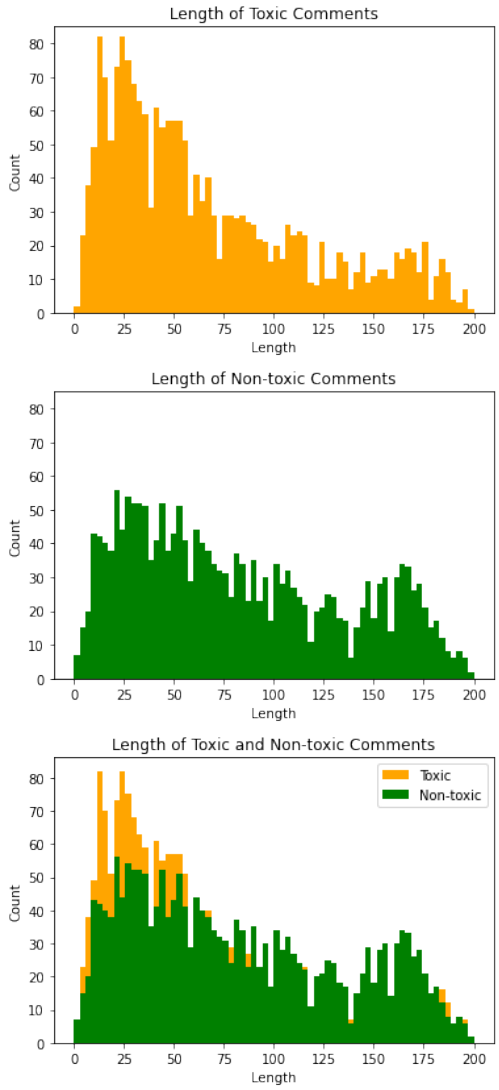

Figure [3.1.3]

3.2. Baseline

The baseline used for this project is by counting the number of hate words’ appearance in each of the

sentences. The source of the hate words dictionary is from [4]. A threshold has been fine-tuned by the training and development set in order to achieve maximum accuracy. As seen in Figure [3.1.3], under the same number of samples (both 2000), it is common to see more hate words in toxic comments. Therefore, theoretically, this baseline should give better results than random choice.

3.3. Bernoulli Naïve Bayes

The first machine learning model adopted in this project is the Bernoulli Naïve Bayes. The implementation is based on the sklearn library. The Bernoulli NB model takes binary features as input and outputs the predicted class labels. To make the data type convention easier, the TFIDF dataset is used here. The format of TFIDF data is a 1000-dimensional vector, where each cell represents the TFIDF score for the word in the given dictionary. And the range of the score is between 0 and 1. The convention method used is simply to set all values greater than 0.1 to be 1, and else to be 0.

3.4. Multilayer perceptron

Another method implemented is multi-layer perceptron (MLP). The MLP model takes numerical data as input and is more capable of handling hidden internal connections between data. The structure of the network is built with the TensorFlow Keras library. Embedded sentences are used as input, where each sample is a 384-dimensional vector. One important thing to note is the dataset imbalance problem mentioned in section 3.1. The MLP model will be more likely to predict a sample as non-toxic. Because the model will gradually find out that simply predicting samples as non-toxic will usually give a smaller loss. To solve such an issue, the training dataset is firstly under-sampled for non-toxic classes. Initially, there were 117514 non-toxic samples. This number has been reduced to 50000 after under-sampling. The remaining 22486 toxic samples are oversampled to match 50000. The oversampling method is by randomly copying original toxic comments. The loss function used in the training phase is 'binary_crossentropy', which is suitable for the task. The network structure used is shown below.

Model: "sequential_2"

Layer (type)	Output Shape	Param #
dense_8 (Dense)	(None, 50)	19250
dense_9 (Dense)	(None, 32)	1632
dropout_2 (Dropout)	(None, 32)	0
dense_10 (Dense)	(None, 12)	396
dense_11 (Dense)	(None, 1)	13

Total params: 21,291
Trainable params: 21,291
Non-trainable params: 0

3.5. LSTM

An LSTM model is implemented using a similar pipeline as the MLP model. The LSTM model has more advantages in dealing with time sequence data, which in the context, can be seen as the sentence. The main reason for implementing LSTM is to directly compare with the performance of MLP. The structure is shown as followed.

Model: "sequential"

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 384, 128)	66560
flatten (Flatten)	(None, 49152)	0
dense (Dense)	(None, 1)	49153

Total params: 115,713
Trainable params: 115,713
Non-trainable params: 0

3.6. Bias Exploration

The method used to explore the bias is by separately selecting subsets of samples that only contain one of the five categories. Samples having more than one category are neglected to reduce noise and other influences.

4. Results

4.1. Baseline Model

The baseline model mentioned in previous section has 78.1% accuracy on the development dataset. However, the recall is only 18.7% and the precision

is 25.5%. This is because the dataset used for evaluation is imbalanced. In the test set, where the two classes are equally distributed, the baseline achieved 54% accuracy.

Table [4.1.1] Baseline Performance

Metrices	Baseline Model
Accuracy (Test Set)	54%
Accuracy (Dev Set)	78.1%
F1 (Dev Set)	21.6%
Precision (Dev Set)	25.5%
Recall (Dev Set)	18.7%

4.2. Bernoulli Naïve Bayes

The Bernoulli Naïve Bayes model generally outperformed the baseline. On the development set, this model gives 75.9% accuracy, 51.1% recall, and 39.5% precision. The accuracy on the test set is 66.4%. Compared with the baseline model, the Bernoulli NB improved the recall and precision by 32.2% and 14% respectively

Table [4.2.1] NB Performance

Metrices	Bernoulli Naïve Bayes	Improvement on Baseline
Accuracy (Test Set)	66.4%	+12.4%
Accuracy (Dev Set)	75.9%	-2.2%
F1 (Dev Set)	44.5%	+22.9%
Precision (Dev Set)	39.5%	+14.0%
Recall (Dev Set)	51.1%	+32.4%

Table [4.2.2] Category Bias

Metrices	Christian	Female	Homosexual gay or lesbian	Male	Muslim
True Positive	14.3%	25.2%	46.9%	42.9%	63.2%
True Negative	98.0%	88.9%	79.5%	81.4%	70.0%
Avg Accuracy	56.1%	57.0%	63.2%	62.2%	66.6%

4.3. Multilayer Perceptron

The MLP models showed huge potential in dealing with embedded data. In detail, the overall accuracy of the development set can be 73.3%. On the test set, this number is 73.8%, which is quite close to the development result. For other metrics, this model gives 76.3% recall and a 38.3% precision rate. The parameter used for training is 4 epochs, binary_crossentropy loss, adam optimizer, and relu activation function. This model showed its great potential for this task, it has a very high true positive rate, which is the predominant focus. Another interesting observation is the MLP model can very easy to be overfitted. Based on experiments, we found that the model's performance will drop a lot for epochs greater than 8, even with dropout layers. The detailed discussion will be in section 5.

Table [4.3.1] MLP Performance

Metrices	Multi-Layer Perceptron	Improvement on Baseline
Accuracy (Test Set)	73.8%	+19.8%
Accuracy (Dev Set)	73.3%	+19.3%
F1 (Dev Set)	44.5%	+22.9%
Precision (Dev Set)	38.3%	+12.8%
Recall (Dev Set)	76.3%	+57.6%

Table [4.3.2] Category Bias

Metrices	Christian	Female	Homosexual gay or lesbian	Male	Muslim
True Positive	63.6%	74.1%	75.4%	70.0%	79.2%
True Negative	90.1%	76.9%	51.7%	79.1%	58.2%
Avg Accuracy	76.8%	75.5%	63.5%	74.5%	68.7%

4.4. LSTM

The performance of LSTM is close to MLP, after 5 epochs of training, the LSTM model gives 73.6% accuracy on the test set, 72.5% accuracy on the development set, 75.2% recall rate, and 38.3%. In comparison to the MLP model, the LSTM would take a much longer training time without GPU acceleration.

Table [4.4.1] LSTM Performance

Metrices	LSTM	Improvement on Baseline
Accuracy (Test Set)	73.6%	+19.6%
Accuracy (Dev Set)	72.5%	+18.5%
F1 (Dev Set)	50.4%	+28.8%
Precision (Dev Set)	38.3%	+12.8%
Recall (Dev Set)	75.2%	+56.5%

Table [4.4.2] Category Bias

Metrices	Christian	Female	Homosexual gay or lesbian	Male	Muslim
True Positive	53.6%	70.8%	75.8%	75.7%	78.5%
True Negative	90.1%	78.2%	58.5%	77.9%	61.2%
Avg Accuracy	71.8%	74.5%	66.9%	76.8%	69.8%

5. Discussion

Table [5.1] Overall Comparison

Metrices	Baseline Model	Bernoulli Naïve Bayes	Multi-Layer Perceptron	LSTM
Accuracy (Test Set)	54%	66.4%	73.8%	73.6%
Accuracy (Dev Set)	78.1%	75.9%	73.3%	72.5%
F1 (Dev Set)	21.6%	44.5%	44.5%	50.4%
Precision (Dev Set)	25.5%	39.5%	38.3%	38.3%
Recall (Dev Set)	18.7%	51.1%	76.3%	75.2%

The previous section has shown that both the MLP and the LSTM method give a significant improvement on the baseline and the Naïve Bayes.

The reason for Naïve Bayes giving bad performance could be the feature vector is too sparse. As we are using the 1000-dimensional TFIDF data, in which most of the cells are zero. Another reason could be the TFIDF may reflect some hate words have high relevance to a sentence, but as illustrated in section 3.1, some hate words do appear in non-toxic comments. This would bring a lot of errors of 'hate words in non-toxic comment' type. And since we didn't do the lemmatization, it could also be a problem. The 'Naïve' assumption itself contradicts the task, as the words in a sentence should not be independent of each other.

For the MLP and LSTM methods, while their overall performance is quite close, their performance on long comments varies. Observed from the development set, the MLP's accuracy is about 70% and the LSTM's performance is above 72%. The main reason for this could be the LSTM model can better handle long-term dependencies. Another difference noted is the recall rate. The MLP model has a higher recall rate compared with the LSTM method. During training, it has been noted that the MLP and LSTM models can very easy to overfit on the 384-dimensional embedded vector. To prevent this, drop-out layers can be added to randomly 'kill' some neural in every epoch. Another possible way is reducing training epochs. Based on numerical data, adding drop-out layers, and carefully choosing training epochs can improve the model's performance by 3-5 percent. Apart from this, balancing the training dataset can also improve the model's accuracy by almost 10 percent.

In terms of categorical bias, for the 384-dimensional embedded vector dataset, the 'Muslim' and 'Homosexual gay or lesbian' category is witnessed with significantly low average accuracy. The statistical results show that the training set only contains 3797 'Homosexual gay or lesbian' samples and 11148 'Muslim' samples, while there are 22477 and 22361 'Christian' and 'Female' samples respectively. One possible way to decrease the gap is to do the data balancing on each of the categories separately. The pipeline could be the same as mentioned in section 3.4. Another performance gap is witnessed in the true positive rate on 'Christian', which is about 20% lower than other categories. To solve this, the 'Christian' category can be trained separately. In the prediction phase, the category with high bias can be predicted with their specific model. It has been verified that training separate models for the high-bias category can improve the overall accuracy by about 1%. For the 1000-dimensional TFIDF dataset, the 'Christian' and 'Female' category has about 7% average accuracy than others. After analyzing the training dataset, we found those two categories have the longest average comment length. The bias for the Naïve Bayes model with the TFIDF dataset could be long comments can have more internal connections, but the Naïve Bayes model cannot be aware.

6. Conclusions

In conclusion, this paper has shown both the MLP and LSTM models have a significant advantage in classifying toxic comments. The overall accuracy of these two methods is quite close (above 74%), but the MLP gives a higher recall rate. In practice scenarios like social media, filtering toxic comments should focus more on a high recall rate. Therefore, the MLP model is the most preferable method in the context.

The bias in different categories has also been discussed. There are three different biases have been reported. The potential reason and solution to them are also noted.

References

[1] Jigsaw/Conversation AI. *Jigsaw unintended bias in toxicity classification.* https://www.kaggle.com/competitions/jigsaw-unintended-bias-in-toxicity-classification Accessed: July, 2022

[2] Zaheri, Sara; Leath, Jeff; and Stroud, David (2020) *"Toxic Comment Classification,"* SMU Data Science Review: Vol. 3: No. 1, Article 13. Available at: https://scholar.smu.edu/datasciencereview/vol3/iss1/13

[3] Pallam Ravi, Greeshma S Hari Narayana Batta, and Shaik Yaseen. *Toxic comment classification.* International Journal of Trend in Scientific Research and Development (IJTSRD), 2019.

[4] Hakimov, S. and Ewerth, R. (2021) *Combining Textual Features for the Detection of Hateful and Offensive Language,* arXiv.org. Available at: https://arxiv.org/abs/2112.04803 (Accessed: 3 October 2022).