# Toxic Comment Classification

Zhao, Cai Liu, Yitong Lu, Kuojian Wang, Chao Ge, Mengye

## Abstract

In this project, we utilize Natural Language Processing techniques to solve the toxic comment classification problem, which is a Kaggle competition using data from Wikipedia's talk page edits. Specifically, several different methods are used to extract features from the comment text and then complete classification. We evaluate the performance of these methods and also make comparisons.

## 1 Introduction

With the extremely convenient access to the internet, toxic comments (i.e. comments that are rude, disrespectful or otherwise likely to make someone leave a discussion) on various platforms are a pretty evident phenomenon. These toxic comments may result in really bad effects on the victims, even death. Thus, it is necessary and meaningful to build models to detect these toxic comments and prohibit them being posted online. This is actually a text classification problem [1], a fundamental task in NLP (Natural Language Processing) applications.

In this project, we use five different text classification methods to tackle the toxic comment classification task.

The project report is organized as follows: We first describe the problem of text classification and hence the toxic comment classification task in section 2. Then we introduce the dataset to be used for this project in section 3. The methods are then followed in detail in section 4. Finally, the evaluation results will be described and comparisons will be made in section 5.

## 2 Problem Statement

Text classification is a classic topic in NLP and an essential component in many applications, such as web searching, topic categorization and sentiment analysis. The text classification problem can be simply defined as follows: given a set of documents D and a set of classes C, define a function F that assigns a value for each class of C to each document in D.

To solve the text classification problem, a key issue is feature extraction and representation, which is commonly based on the bag-of-words (BoW) model [3], where unigrams, bigrams or n-grams are typically extracted as features from text. Furthermore, several feature selection methods, such as pLSA (Probabilistic Latent Semantic Analysis) [4], LDA (Latent Dirichlet Allocation) [5], are applied to select more discriminative features. Another effective method is to use deep neural network, which is capable of extracting the latent features from text automatically. Convolutional Neural Network (CNN) [6] and Recurrent Neural Network (RNN) [7], the most two commonly used models, can capture the semantics of a document, thus leading to better feature representation compared with the traditional methods.

For the toxic comment classification task, which is an application of text classification, the first step is to extract features that can represent the toxicity from the comment text. Then taking the extracted features as input, use a classifier such as logistic regression or Support Vector Machine (SVM) to complete classification for different types of toxicity.

## 3 Dataset

The dataset for the toxic comment classification task is provided by a Kaggle competition[1]. It includes 223,549 annotated user comments collected from Wikipedia talk pages. These comments were annotated by human with six types of toxicity, which are toxic, severe toxic, obscene, threat, insult, and identity hate respectively. Comments can

[1] https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge

be associated with multiple classes at once.

### 3.1 Data Exploration

Our group chose to visualize the dataset, so that we can learn about the data more intuitively. First, we want to see the number of comments per class, so that we can know the distribution of these comments. Most comments do not have any labels, while some of them actually have multiple tags. There are even 31 comments that have all six different labels. Second, for the six classes, we want to know their correlation. We got that, toxic, obscenity and insult, they are highly related. Third, the length of each comment. As results, we can see that most of them are under 1000 words. Last, we want to see the most frequent words in both labeled and unlabeled comments. The most frequent words in unlabeled comments are one, talk, page, article, and so on. For the most frequent words in labeled comments. They are nigger, jew, fat, die and some other really bad words.

### 3.2 Data Prepossessing

In this toxic comments classification project, data prepossessing methods were used learning from the knowledge of Natural Language prepossessing. It is a sub-field of computer science, information engineering, and artificial intelligence concerned with the interactions between computers and human languages, in particular how to program computers to process and analyze large amounts of natural language data. In a word, computers need translate human language to some labels that they could used for classifying new input sequences. For this particular task, we have replaced all common abbreviate words to the complete spelling words. For example, some of the texts in our data set have used abbreviate likes 'we'll', it was transferred to 'we will' in the new text files. Moreover, we have deleted most of the common stop words in English, because these kinds of words have no useful meaning to do the classification works. In addition, there is one more issue in our datasets. Both of the training and testing datasets have some texts that are too length or too short to analyzed. Meaningful comments rarely exceed 200 words. In order to keep the input datasets have the similar distributions. We filled up the sequences and unified the length of every input comment.

## 4 Approaches

### 4.1 TF-IDF

Term Frequency-Inverse Document Frequency(TF-IDF) score represents the relative importance of a term in a document and the whole corpus. TF-IDF score is composed by these two terms, and the formulas are:

$$TF(t) = \frac{number\ of\ times\ term\ t\ appears\ in\ a\ comment}{total\ number\ of\ terms\ in\ the\ comments}$$

$$IDF(t) = log_e \frac{total\ number\ of\ comments}{number\ of\ comments\ with\ term\ t\ in\ it}$$

$$TFIDF(t) = TF(t) * IDF(t)$$

Therefore, a term will have a high TF-IDF score if it appears frequently in a comment but not very often throughout the corpus. TF-IDF vectors can be generated at different levels of input tokens, such as word level, n-gram level, and character level. In this project, n-gram is chosen. Using TF-IDF, m comments string can be translated into a m*n matrix, where m is the size of our training set and n is the number of features. The matrix value is the TF-IDF score and the matrix is the input of the model.

Logistic regression (LR) is a widely used model to do classification. There are 6 target classes in the task, so we build a model for each class, and finally we have 6 models in total.

Support Vector Machine extracts a best possible hyperplane that segregates the two classes. Using the same input matrix as we still build a model for each target class, but the model is replaced by SVM instead of LR.

### 4.2 Latent Dirichilet Allocation

In this approach, we use LDA to extract toxic features from comments, and then invoking logistic regression to solve the toxic comment classification problem.

LDA is a topic model which is able to discover the latent topics from documents. Specifically, for a document, LDA produces its probabilities over the latent topics as its explicit representation, and this representation can then be used for text modeling or document classification.

The input of LDA is a document-term matrix, where each term is usually represented by term frequency, and the output is a document-topic matrix, that is, each document is represented as the probabilities over the latent topics.

As for the toxic comment classification problem, which is a multi-label classification task, LDA is much suitable, since we can assume each comment is a probability distribution over the latent toxic topics corresponding to the labels. And also, the number of latent toxic topics can be more than the number of labels, so we can discover much more latent toxic information for the comments.

After extracting the latent toxic topic features from comments by LDA, we simply use logistic regression to complete classification.

## 4.3 Convolutional Neural Network

CNN has proven to be very effective in toxic comment classification [8]. The components of a comment (the words) have to be encoded before fed to the CNN [9]. For this purposed, we may use a vocabulary. The vocabulary is constructed as an index containing the words that appear in the set of document texts, mapping each word to an integer between 1 and the vocabulary size. An example of this procedure is illustrated in Figure 1. The variability in documents length (number of words in a document) need to be addressed as CNNs require a constant input dimensionality. For this purpose the padding technique is adopted, flling with zeros the document matrix in order to reach the maximum length amongst all documents in dimensionality.

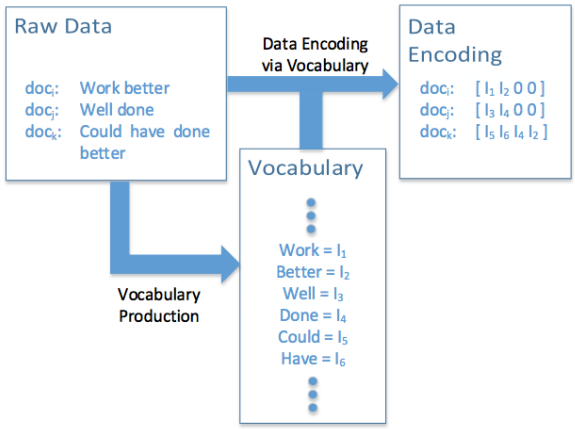


Figure 1: Example of encoding a text using a vocabulary

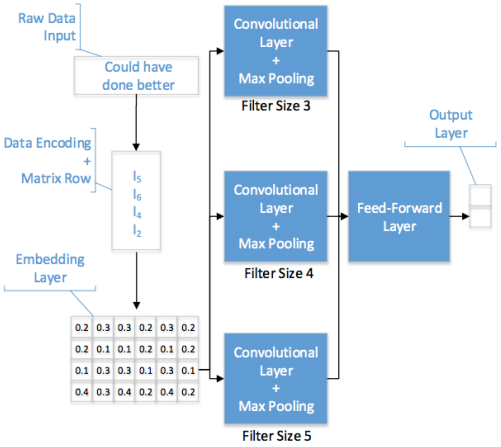


Figure 2: The CNN for text classification process

In the next step the encoded documents are transformed into matrices for which each row corresponds to one word. The generated matrices pass through the embedding layer where each word (row) is transformed into a low-dimension representation by a dense vector [10]. The procedure then continues following the standard CNN methodology. As to word embedding method, Glove [11] is chose , using pre-trained word vectors glove.840B.300d.txt. The values of these vectors do not change during the training process, unless there are words not already included in the vocabulary of the embedding method in which case they are initialized randomly. An indicative description of the process is illustrated in Figure 2. We only use one convolutional layer: the filter number is 250; the size of kernel is 3; the activation is relu; the initializer of kernel is glorot uniform. when training the model, we the loss function we use is binary crossentropy; the optimizer is adam; the metrics is accuracy; the batchsize is 32.

Table 1 shows the summary of the final CNN model we use. The ROC-AUC scores are 0.9794 in private board and 0.9804 in public board that obtained by submission in Kaggle. This means the model can solve toxic comment classification effectively.

| Layer (type) | Output Shape | Param # | Connected to |
|---|---|---|---|
| input_1 (InputLayer) | (None, 200) | 0 | |
| embedding_1 (Embedding) | (None, 200, 300) | 58431300 | input_1[0][0] |
| spatial_dropout1d_1 (SpatialDro | (None, 200, 300) | 0 | embedding_1[0][0] |
| conv1d_1 (Conv1D) | (None, 198, 250) | 225250 | spatial_dropout1d_1[0][0] |
| global_max_pooling1d_1 (GlobalM | (None, 250) | 0 | conv1d_1[0][0] |
| global_average_pooling1d_1 (Glo | (None, 250) | 0 | conv1d_1[0][0] |
| concatenate_1 (Concatenate) | (None, 500) | 0 | global_max_pooling1d_1[0][0] global_average_pooling1d_1[0][0] |
| dense_1 (Dense) | (None, 6) | 3006 | concatenate_1[0][0] |

Total params: 58,659,556
Trainable params: 58,659,556
Non-trainable params: 0

Table 1: the summary of the final CNN model

## 4.4 Long Short-Term Memory

In Natural Language Processing, traditional methods, such as TF-IDF, requires manual text feature extraction, thus they are not suitable for every specific application. While neural network is able to automatically extract text features through its deep network layers. But deep neural network with fully connected layers is not much effective since it fails to identify the semantic relationships between words. Recurrent neural network, on the other hand, is much suitable for sequence data, especially for text. But it also has difficulty in long-term dependencies between words in a sentence due to the vanishing gradient problem. While LSTM, compared with vanilla RNN, is long-term dependencies available owing to its gate mechanism, which also makes LSTM easier to train. So this is why we use LSTM to solve this text classification problem.

Table 2 shows the summary of the final Bi-LSTM (Bidirectional LSTM) we use. And the ROC-AUC scoresare 0.9797 in private and 0.9793 in public obtained by late submission in Kaggle. This model gets the highest private score in our 6 models. we only use one bidirectional LSTM(Long Short-Term Memory) layer: the dimension-

ality of the output space is 128; fraction of the units to drop for the linear transformation of the inputs is 0.2; the fraction of the units to drop for the linear transformation of the recurrent state is 0.15. when training the model, the loss function we use is binary crossentropy; the optimizer is adam; the metrics is accuracy; the batchsize is 32.

| Layer (type) | Output Shape | Param # |
|---|---|---|
| input_1 (InputLayer) | (None, 200) | 0 |
| embedding_1 (Embedding) | (None, 200, 300) | 58431300 |
| spatial_dropout1d_1 (Spatial | (None, 200, 300) | 0 |
| bidirectional_1 (Bidirection | (None, 200, 256) | 439296 |
| global_max_pooling1d_1 (Glob | (None, 256) | 0 |
| dense_1 (Dense) | (None, 6) | 1542 |

Total params: 58,872,138
Trainable params: 58,872,138
Non-trainable params: 0

Table 2: the summary of the final Bi-LSTM model

The architecture of our Bidirectional LSTM model is shown in Figure 3. Firstly, it's an embedding layer at the bottom, where each word in a sentence is encoded as a low-dimensional dense vector as its semantic representation. Bidirectional LSTM layer then learns the features of this sentence, especially those that contribute to the type of toxicity. Following Bidirectional LSTM network, a global max pooling operation is applied to the outputs of Bidirectional LSTM at each time step. This pooling layer is capable of obtaining the most important features for the toxicity classification. Two fully connected layers are then followed as classifier for the extracted text features.

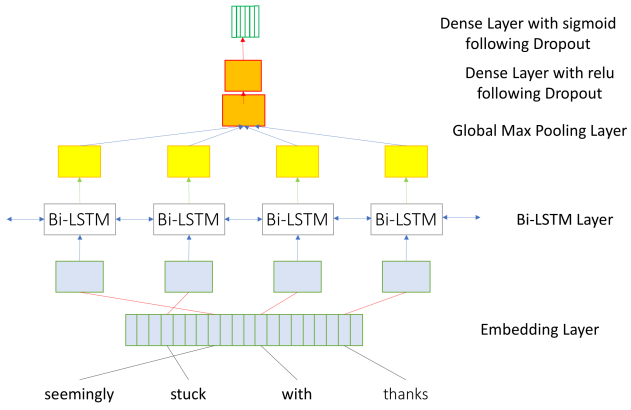


Figure 3: the architecture of the final Bi-LSTM model

The highlights of our bidirectional LSTM model are in two parts. One is the embedding layer. Instead of initializing the word vectors with a uniform distribution, we use pre-trained word embedding with our data prepossessing method, which is described previously. The pre-trained word embedding mechanism enables the bidirectional LSTM network learns from a semantically meaningful state rather than a random state, thus improving the effectiveness of text feature extraction. Another part is the global max pooling operation, which is a commonly used technique in convolutions neural network. Intuitively, instead of extracting all features from a sentence, we would like to pay more attention to those that have the most contribution to the type of toxicity, and this is implemented by pooling operation, just like in image recognition.

## 4.5 LSTM and CNN

Table 3 shows the summary of the final Bidirectional LSTM and CNN we use. The parameters of Bidirectional LSTM layer are: the filter number is 250; the size of kernel is 3; the activation is relu; the initializer of kernel is glorot uniform . And the parameters of CNN layer are: the filter number is 250; the size of kernel is 3; the activation is relu; the initializer of kernel is glorot uniform.when training the model, we the loss function we use is binary crossentropy; the optimizer is adam; the metrics is accuracy; the batchsize is 32. And the ROC-AUC scores are 0.9795 in private and 0.9808 in public obtained by late submission in Kaggle. This method can also solve the problem effectively.

| Layer (type) | Output Shape | Param # | Connected to |
|---|---|---|---|
| input_1 (InputLayer) | (None, 200) | 0 | |
| embedding_2 (Embedding) | (None, 200, 300) | 58431300 | input_1[0][0] |
| spatial_dropout1d_2 (SpatialDro | (None, 200, 300) | 0 | embedding_2[0][0] |
| bidirectional_1 (Bidirectional) | (None, 200, 256) | 439296 | spatial_dropout1d_2[0][0] |
| conv1d_2 (Conv1D) | (None, 198, 250) | 192250 | bidirectional_1[0][0] |
| global_max_pooling1d_2 (GlobalM | (None, 250) | 0 | conv1d_2[0][0] |
| global_average_pooling1d_2 (Glo | (None, 250) | 0 | conv1d_2[0][0] |
| concatenate_2 (Concatenate) | (None, 500) | 0 | global_max_pooling1d_2[0][0]<br>global_average_pooling1d_2[0][0] |
| dense_2 (Dense) | (None, 6) | 3006 | concatenate_2[0][0] |

Total params: 59,065,852
Trainable params: 59,065,852
Non-trainable params: 0

Table 3: the summary of the final Bi-LSTM and CNN model

## 5 Evaluation and Comparison

For all the five methods, we uploade the final results to Kaggle website, and get both public and private scores. We put all of them into one table (Table 4), so that we can compare them more easily. As we can see, the evaluation result of LDA is not much good as we expect. The accuracy is even lower than TF-IDF. The reason is that it is difficult to determine the number of latent topics and different number of topics can result in totally different performance. The first three models all belong to neural network. The scores are very high and pretty much similar. While Bidirectional LSTM got the highest score. For Logistic Regression, the score is only a little bit lower than the best one. But is has a distinct advantage. The running time is the shortest among all models.

| Model | Private score | Performance |
|---|---|---|
| Bidirectional LSTM + CNN | 0.9795 | Not bad |
| Bidirectional LSTM | 0.9797 | The best |
| CNN | 0.9794 | Not bad |
| TF-IDF with LR | 0.9739 | The fastest |
| TF-IDF with SVM | 0.9445 | Not bad |
| LDA with LR | 0.9135 | The worst |

Table 4:Comparison of Models

## 6 Conclusion

For conclusion, Bidirectional LSTM performs better than other models. But it costs much more training time and it has higher requirement of computing devices. While Logistic Regression works much faster than other models. And the results are also good. In this project, to , we use TF-IDF, LDA, CNN and LSTM to extract features from comments.

## 7 Acknowledgments

Data exploration is done by Ge, Mengye, and data preprocessing is done by Wang, Chao. Liu, Yitong implements TF-IDF with logistic regression and SVM, and Lu, Kuojian implements LDA with logistic regression. The approaches of CNN and Bidirectional LSTM are implemented by Zhao, Cai.

## References

[1] Charu C. Aggarwal, ChengXiang Zhai:A Survey of Text Classification Algorithms. Mining Text Data 2012: 163-222.

[2] Murty, M. Text Document Classification basedon Least Square Support Vector Machines with Singular Value Decomposition. International Journal of Computer Applications, 2008, 27: 21-26.

[3] Harris, Zellig. Distributional Structure. Word, 1954, 10: 146–62.

[4] Lijuan Cai, Thomas Hofmann: Text categorization by boosting automatically extracted concepts. SIGIR 2003: 182-189.

[5] Swapnil Hingmire, Sandeep Chougule, Girish K. Palshikar, Sutanu Chakraborti: Document classification by topic labeling. SIGIR 2013: 877-880.

[6] Yoon Kim: Convolutional Neural Networks for Sentence Classification. EMNLP 2014: 1746-1751.

[7] Siwei Lai, Liheng Xu, Kang Liu, Jun Zhao: Recurrent Convolutional Neural Networks for Text Classification. AAAI 2015: 2267-2273.

[8] Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classi cation. CoRR abs/1408.5882 (2014). arXiv:1408.5882 http://arxiv.org/abs/1408.5882

[9] Spiros V. Georgakopoulos, Sotiris K. Tasoulis, Aristidis G. Vrahatis, Vassilis P. Plagianakos Convolutional Neural Networks for Toxic Comment Classification https://arxiv.org/abs/1802.09957

[10] Ronan Collobert, Jason Weston, and Pavel Kuksa. 2011. Natural Language Processing (Almost) from Scratch. J. Mach. Learn. Res. 12 (Nov. 2011), 2493–2537. http://dl.acm.org/citation.cfm?id=1953048.2078186

[11] Je rey Pennington, Richard Socher, and Christopher D. Man- ning. 2014. Glove: Global vectors for word representation. In In EMNLP.