Toxic Comments Classification

Yue Xiong
University of California, San Diego

Hongjian Cui
University of California, San Diego

ABSTRACT

This paper analyses a data set of comments from Wikipedia's talk page edits and tries to detect different types of toxicity, such as threats, obscenity, insults, and identity-based hate. It attempts to do the task by building a Logistic Regression model and comparing different sets of TF-IDF features to find best one to classify each toxicity type.

1 INTRODUCTION

Online abusing occurs everywhere. It makes discussing things we care about online difficult. While we want to chat with others, there is always somebody insulting people online. The threat of harassment online means that many people stop expressing themselves and give up on seeking different opinions. Obscene, hateful, and nasty comments or texts not only lead to verbal violence online, including sexual harassment, personal attacks on other organizations or characters by a well-known organization, denying or slandering them in a disrespectful and vicious manner, bullying others or even Racism in hostile languages, and sometimes life-threatening, can be found on online social platforms such as blogs, hate sites, and comment sections. Although people are working to improve the security of the online environment based on the ability to crowdsource voting plans or condemn comments, in most cases these technologies are inefficient and cannot predict the potential for toxicity. Platforms struggle to effectively facilitate conversations, leading many communities to limit or completely shut down user comments. There are already lots of researchers working on tools to help improve online conversation. One area of focus is the study of negative online behaviors, like toxic comments (comments that are disrespectful or obscene). While in the past, the toxic comments are often detected by human, nowadays we can use machine learning tools to help us.

In this study, we will try to implement a machine learning algorithm and try to classify the toxic comments from normal comments. What's more, if a comment is identified as "toxic", the model we built will also determine which "toxic" category it belongs to.

The paper is organized as follows: Section 2 tells the information of dataset and explores the relationship of feature with in the dataset. Section 3 how we preprocess our data in order to fit it into our model. Section 4 discusses the models we use in this task. Section 5 discusses the related literature of similar work (Classifying toxic comments). Section 6 summarize the work and describe the results and conclusions.

2 DATASET

2.1 Identify a dataset to study

We use a data set from Kaggle that contains 223,549 comments from Wikipedia talk page edits to perform this study. We choose this data set mainly because it is well labeled. Each comment (data point) has been exactly classified as non-toxic or toxic (if it is toxic, it tells which kind of toxic it belongs to).

We separate the dataset and use 158571 number of comments as our training set. A validation set (contains 159,571 comments), is used to evaluate each of our model performs and select the best one. We will use the test set (contains 57,580 comments) to evaluate the "best" model's performance. We will evaluate the performance by doing cross-validation. A final prediction accuracy is calculated from performance on the test set.

2.2 Exploratory analysis

2.2.1 *Dataset's feature explanation.*

Each data point contains of the following features.

Feature name	Explanation
id	The id of the comments
comment_text	The text of the particular comment
severe_toxic	One-hot encoding indicating whether a comment contains severe toxic information
obscene	One-hot encoding indicating whether a comment contains obscene information
threat	One-hot encoding indicating whether a comment contains threat information
insult	One-hot encoding indicating whether a comment contains insult information
identity_hate	One-hot encoding indicating whether a comment contains identity hate information

2.2.2 *Feature exploratory.*

We first make a plot and take a look at the general information of the dataset. Figure 1 shows the distribution of the toxic comments. We can see that the distribution of the toxic comments is not uniform. The number of "severe_toxic", "threat" and "identity_hate" comments are extremely less than the others. Also, noted that the sum of all toxic comments is 15,294, which is far less than total number of comments (159,571) we can know that there are lots of non-toxic comments in the training set (the one-hot encoding corresponds to toxic information of which are all 0).

Furthermore, we look inside the comment and try to dig some useful information. Since one comment can be classified as different kinds of category at the same time, there might be some relationship between different types of toxic comments. What we do is we compute a correlation matrix and visualize it to find out the correlation between each of the category (Figure 2). The result shows that "toxic" comments are clearly strongly correlated with both "obscene" (pearson correlation of the two is 0.68) and

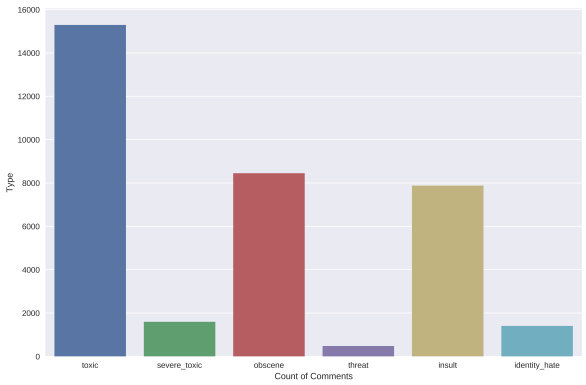

Figure 1: distribution of toxic comment

"insult" comments (pearson correlation of the two is 0.65). The "insult" comment is also strongly correlated with "obscene" comment" (pearson correlation of the two is 0.74). To see the overlap between these toxic categories more clearly, we generate a Venn Diagram for "toxic", "insult" and obscene" comments. It proves what we get from the correlation matrix.

Figure 2: correlation matrix of each type of comment

What we then do is try to look into each category of toxic comments and try to find out the words which are most frequently use by different types of toxic comments. Below are the word clouds generated from each category of toxic comments (Figure 3, Figure 4, Figure 5, Figure 6, Figure 6, Figure 7, Figure 8). From these graphs, we can actually see that toxic comments of category "toxic", "severe_toxic", and "obscene" actually share very similar words, while category of "insult" and identity_hate" have very similar most frequently used words.

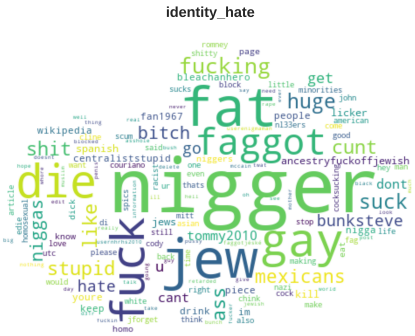

Figure 3: wordcloud of comments from identity_hate

Figure 4: wordcloud of comments from insult

Figure 5: wordcloud of comments from obscene

Figure 6: wordcloud of comments from toxic

Figure 7: wordcloud of comments from severe_toxic

Figure 8: wordcloud of comments from threat

3 PREDICTIVE TASK

Our predictive task is to determine whether a comment is toxic or not. If it is toxic, we also need to determine which of the 6 toxic categories it belongs to. In other words, what we will do is try to predict the one-hot features in the dataset. Note that since one comments can belong to several category at the same time, the way we build our model is different than normal. Instead of building one model that predict the result in one time, we build 6 separate model to predict whether a certain comment belongs to 6 of the certain toxic categories and combines the results together. The reason we do this is that in this way we can maximize the accuracy in predicting single toxic category and thus increase the total accuracy in the end.

The baseline we will use for comparison is a naïve model from what we have used from Assignment 1 task 2 which is doing "Bag of Words" approach and use the number of each words' frequency as feature matrix and apply logistic regression. The reason that why the model is appropriate for this task is that from what we have seen from the word cloud, different categories of toxic comments tend to have different most frequently used words. So if we count the how many times each words appear in the comments, it is likely that we will be able to distinguish the comments. Also, if we dig this task and Assignment 1 task 2 deeper, we can find that both tasks try to identify the category of a certain text (comment), so what works in the past should also have pretty good result in this task.

In order to improve our performance, we need to preprocess the data and generate some new features so that it can fit into our machine learning models.

From the result of our exploratory analysis and the task itself we know that the key of the task is try to identify the words feature in the text. What we need to do first is merge different inflections of words ("stemming"). After the that, we remove the "stopwords" because we don't want meaningless words appear in our feature vector. What we then do to our dataset is basically "Bag of Words" and TF-IDF approach and try to generate word feature vectors that represent each comment.

4 MODEL

We use Logistic Regression as our machine learning model. Generally, we choose this model we learned from class because it got pretty good result in our assignment 1.

4.1 Baseline

As what we do in Assignment 1 Task 2, after we generate the feature vector. We try the unigram approach. What we do is count each word's frequency and compute the feature vector. Then put it into the logistic regression model. One of the key issues we faced in estimating the model is tuning the parameters. In this task, we use gird search to tune the parameters of logistic regression (the value of C, which corresponds to the penalty we add to the model to prevent overfitting) and number of words we want to use as our feature vector.

Below are the results on validation set.

category	Accuracy
toxic	0.95411
severe_toxic	0.94236
obscene	0.96347
threat	0.94235
insult	0.95753
identity_hate	0.95436

4.2 TF-IDF approach

Because from what we have seen from the word cloud, each type of toxic comments has many similar most frequently used words. This might be an issue in the baseline model because we may not be able to classify two types of comments that have almost the same 10 most frequently used words. This is why we turn to use TF-IDF approach. Instead of simply counting the number of times each word appears in comments, TF-IDF approach tries to look deeper. It increases proportionally to the number of times a word appears in the document and is offset by the number of documents in the corpus that contain the word, which helps to adjust for the fact that some words appear more frequently in general. Using the sklearn function TfidfVectorizer we can extract "Bag of Words" from the text and apply TF-IDF weights to each of the words.

After preprocessing the data, we apply unigram approach. It is similar as the baseline, but now instead of using the number of times each word appears, we use the computed tfidf value. Again, we apply grid search to find the best parameter. The accuracy is as follows

Below are the result on the validation set.

category	Accuracy
toxic	0.97256
severe_toxic	0.98273
obscene	0.98633
threat	0.98753
insult	0.97801
identity_hate	0.97512

4.3 Bi-grams

Using unigram may face some problems such as making word "good" and "not good" having same weights associated in a sentiment model. This issue can be fixed by applying bigrams, which is counting words of two at one time. We apply bigrams method in our model and same as what we did before, we use grid search to estimate the best value of parameter. The result is as follows

Below are the result on the validation set.

category	Accuracy
toxic	0.97411
severe_toxic	0.97453
obscene	0.98947
threat	0.98983
insult	0.97314
identity_hate	0.97041

As we can severe_toxic and identity_hate type classification gives best score with unigram features, the rest of the types do best with bigrams. So now we can apply corresponding vectorizers to test data and make predictions.

4.4 Problems

One of the issues we met in implementing our approach is that we always find memory error when we are making the feature vector and training our model. The reason of which is that our naïve way of building feature matrix always use loop and didn't use sparse matrix, which is very inefficient and memory expensive. We solve this by trying using the existed sklearn method, "CountVectorizer", "TfidfVectorizer". It not only solve the memory error problem, but also reduce our time of training one time from several hours to couple of minutes!

5 RELATED LITERATURE

The dataset we use is from Kaggle (https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge). It was used as a competition whose purpose is to build a multi-headed model that's capable of detecting different types of toxicity like threats, obscenity, insults, and identity-based hate better than Perspective's current models (similar to our task).

There are ongoing experiments to test and calculate the existence of various toxicity on online platforms, including effective models established by industry and research groups on micro and macro blog sites that can detect and predict online toxicity reviews.

Due to the rapid growth of online interactions among users, this is of great significance in the research field. The rise of the deep neural system (DNN) has greatly benefited natural language processing (NLP) due to their high performance and fewer design highlights. DNN has two basic structures: Convolutional Neural Network (CNN) (LeCun et al., 1998) and Recurrent Neural System (RNN) (Elman, 1990)[3]. The input of most NLP tasks is sentences or documents represented as matrices. Each row of the matrix corresponds to a token, which is usually a word, but it can also be a character. That is, each row is a vector representing a word. Usually, these vectors are word embeddings (low-dimensional representations), such as word2vec or GloVe, but they can also be one-stop vectors for indexing words into a vocabulary[2]. Using the word embedding technique, the results of the primary neural network algorithm are also compared with complex convolutional neural networks and recursive neural networks, and the result is: LSTM (Long Short-Term Memory) results. The analysis obtained shows that the performance of LSTM is better than CNN. Given the same number of epochs, both accuracy and time performance are the same, so it is preferable to CNNs with word-level embeddings. By using finer pre-processed data and progress in developing the proposed LSTM system, word embedding can be further improved, resulting in more accurate and promising results[1].

6 RESULT AND CONCLUSIONS

Below are the result on the test set.

category	Accuracy
toxic	0.97334
severe_toxic	0.98789
obscene	0.98486
threat	0.98032
insult	0.97266
identity_hate	0.97752

We start our work by building a feature vector that only count the number of times each word appears. We improve this by using TF-IDF approach which uses the tf-idf value of the word as feature matrix. The latter model performs much better than the previous one. The parameters of the model represent the weight of each words' tf-idf value contributes to the result. The reason why TF-IDF approach is much better than the previous one is due to the fact that there are many words which have very similar frequency in all comments, but very different frequency in different types of toxic comment.

7 FUTURE WORK

As what we mention in the related literature, maybe we should try to use CNN, RNN approach and try to use word2vec or GloVe method to compute the feature vector.

REFERENCES

[1] Maeve Duggan. *Online harassment.* Pew Research Center, 2014.
[2] Hossein Hosseini, Sreeram Kannan, Baosen Zhang, and Radha Poovendran. Deceiving google's perspective api built for detecting toxic comments. *arXiv preprint arXiv:1702.08138*, 2017.
[3] Ye Zhang and Byron Wallace. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *arXiv preprint arXiv:1510.03820*, 2015.