Predicting Toxicity over Diverse Online Conversation

Arti Singh
UMBC - Computer Science
artis2@umbc.edu

Rohit Kumar
UMBC - Computer Science
rohitk1@umbc.edu

Abstract

In today's world, we have a high multitude of online social media platforms like Twitter, Facebook, CNN etc. where people actively participate and comment over the published articles, news or videos. These comments may be toxic which is a biggest threat to open thinking and thus needs to be identified. The threat of abuse and harassment online means that many people stop expressing themselves and give up on seeking different opinions, so there is a need to protect voices in conversations. This paper aims to implement a model to detect toxicity in an online conversation. The model solves some of the significant challenges related to the field. We implemented the model in three phases: preprocessing of data, creation of feature vectors like TFIDF, Word2Vec and Doc2Vec, algorithms and evaluation metrics. Further, we optimized our output by working and experimenting with various features and models like SVM, Logistic Regression, Naive Bayes and NN . We created a baseline model using SVM for our binary classification task to use it as a standard to compare our future models implementation.

1 Introduction

As the world is evolving at a fast pace to digitization with more use of online presence and forums, people actively participate in conversation and provide comments online. However, sometimes as the conversation proceeds, it may become abusive, hate-based or insulting. This may leave people stop expressing themselves and to stop seeking others opinions out of fear of being abused, thus changing their mind. Therefore, it becomes a matter of utmost concern to handle toxicity in the conversation.

Toxicity, in general term is defined as anything rude, disrespectful or otherwise likely to make someone leave a discussion. Inspired by this thought, we thus planned to provide a solution to this problem by determining the toxicity level of a comment and classifying the comment as toxic or non-toxic. Solutions to this problem involves dealing with some of the common challenges of Natural Language Processing like out-of-vocabulary words, longer dependencies, slang or misspelled words. We used Wikipedia comments data-set (Jigsaw, 2019) from Kaggle Toxic comments classification challenge to implement and evaluate our proposed solutions. Each comment in the data-set has a toxicity label and several additional toxicity sub-type attributes. As per our analysis from proposed solutions, Convolutional Neural Network with bigram computation based on word2vec embedding created on our corpa performed better for our goal to predict toxicity efficiently and accurately.

In our work, we have experimented with word embeddings like TFIDF, word2vec, contextual and n-gram features with various models like SVM, Naive Bayes, NN etc. to maximize the correctness of predicting the target toxicity of a comments and evaluate the outcome of these models using F1-score. We included the idea of using the word distributed representation to be fed as input to neural network models like MLP, Bi-LSTM, CNN as described by (Djuric et al., 2015) to determine the target toxicity present in the sentence.

We first applied TfIdf on the 235K comments and used the TfIdf vectorizer to create feature vectors and passed this feature vector as input to SVM, Naive Bayes and Logisitc Regression models. However, these models with the cre-

ated feature vector did not gave the expected F1 score. So, to improve the accuracy further, we used word2vec's continuous bag of word and skip-gram. This was used to create word embeddings which we later passed as an input to Neural Network models like basic MLP, Convolutional Neural Network and Bi-LSTM to maximize the F1 score of model such as to increase the score of predicting the toxic comments giving better F1 score results.

2 Proposed Solution

2.1 Data Pre-processing

We would be doing data pre-processing on the selected data-set which would involve looking at size of input records in each category, finding any trend or pattern among the data in order to get a clear insight of the data-set.

Based on the data visualization, we would further clean data by removing/updating stop words, punctuation marks, adding lemma and morph detail of each word. Once the data pre-processing is complete, we would take out the bag-of-word to apply further algorithm technique so as to classify the comments as toxic or non-toxic.

2.2 Proposed Models

We plan to start by creating a baseline implementation for our binary classification task which serves as a standard comparison model for further implementation. Here, we plan to use Support Vector Machine for this task. Further, we will implement n-gram language models, Multinomial Naive Bayes and Gaussian Naive Bayes. We would be using various pre-trained embeddings to create diverse models such as TfIdf, Word2Vec, etc.

However, to minimize unintended bias so as to handle scenarios where a word may be toxic as word but not in the context of comment, we plan to implement N-gram language model, Maxent model, RNN like LSTM, or bi-LSTM. Implementing LSTM, bi-LSTM (Uppal, 2019), NN may become a potential challenge for us in this project as in Betty (van Aken et al., 2018).

2.3 Evaluation Methods

For each model, we would perform evaluation testing on development data, and the best performing model from these based on F1 score, would then be tested on the test data to predict the target toxicity of comments. For evaluation, test set examples with target $>= 0.5$ will be considered to be in the positive class (toxic comments).

To evaluate the models, we would make comparison across models using accuracy, precision, recall and F1 score. These can be obtained by building a confusion matrix. The confusion matrix provides the positive and negative predictions through the count values for each category. This helps to gain insight about the error measure made by model for each of the class. Since, we aim for the model to classify toxic comment as toxic and non-toxic as non-toxic, we would aim to maximize both recall and precision indicating a better F1 score and a good model.

3 Related Works

Toxicity detection can be broadly classified under sentiment analysis as negative, positive or neutral toxic comments. Sentiment analysis consisting of spams and toxicity has been studied and researched a lot during last decade where scientists and researchers have applied all the best tools and technology in the field of Data Science, Machine Learning and NLP. Toxicity present in a sentence has become a hot topic of research and discussion now-a-days, mainly due to the vast presence and growth of online platform where people come together and express their opinions. Several works have been proposed over the years.

(Yin et al., 2009) work based on sentiment analysis is the first one which provides a base for toxic comment detection on web. Lots of the previous work in this field has been spread across the overlapping aspect of toxic comment classification at its severity level which includes (Dadvar and de Jong, 2012) work on cyber-bullying detection, (Nobata et al., 2016) work on abusive language detection and (Chen et al., 2012) work on offensive language detection on social media.

(Yin et al., 2009) work involves n-gram supervised classification based on content, sentiment and contextual features of the document with its base as TF-IDF approach. (Yu et al., 2017) provides enhanced word embedding idea which captures more semantic information and can be applied to pre-trained word vectors like Word2Vec and Glove. (Chen et al., 2012) takes a step further and is the first one to use the lexical and parser features for the detection of offensive language. They also provides a supervised classification using the

Support Vector Machine approach (SVM) including n-grams features, word blacklist and manually developed grammars and dependency parse features.

Various other recent works include models based on deep learning and neural network based classification using sentence representation. An interesting work in this field includes the proposed idea by (Djuric et al., 2015) which makes use of the distributed representation of sentence and paragraph in neural network and deep learning models as proposed in the paper by (Le and Mikolov, 2014b).

In our work, initially we experimented with TFIDF word embeddings considering unigram, contextual and bigram features as input to SVC, Naive Bayes and Logistic Regression using methods and works from the paper cited above. A.D. Pozzolo (Pozzolo et al., 2015) work show that using UnderSampling doesn't impact the probabilities but increases the overall accuracy of a classifier. So, we have used RandomUnderSampling to handle the unbalanced data for our model's binary classification to improve the accuracy and other measures of the models. Based on experiments and observation these models didn't perform as per the expectations. So as described by (Djuric et al., 2015), we experimented with various Neural Network models at word-level classification. Further, as per the work and results provided by Zhang (Zhang and Wallace, 2017) Convolutional Neural Network performed better than other models. So, we implemented CNN using word2vec word embedding over provided corpa and we observed that for our experiments also, CNN outperformed other models.

4 Data-set and Analysis

For the implementation of our proposed solution, we are using Wikipedia comments data-set (Jigsaw, 2019) from Kaggle Toxic comments classification challenge. This data includes 3 csv files: train.csv, test.csv and sample_submission.csv file. Each comment in Train data-set has a toxicity label (target). This attribute (and all others) are fractional values which represent the fraction of human raters who believed the attribute applied to the given comment. The data also has several additional toxicity sub-type attributes. Data contains different comments that have the exact same text. Different comments that have the same text have been labeled with different targets or subgroups.

As a first step towards our implementation, we have pre-processed and visualized the available data set based on different categories. We did an quantitative analysis of comments over the toxicity level. Figure 1 provides a visualization of comments spread over the various level of toxicity.

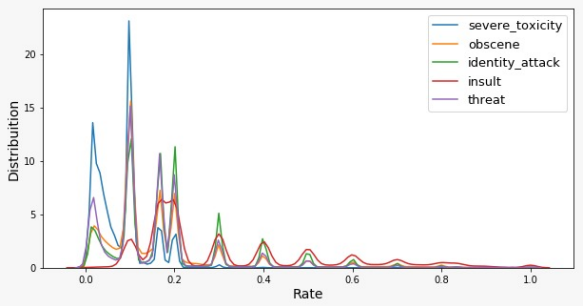

Figure 1: Density Plot of Comments over various Category of Toxicity

Additionally, we also did a ratio analysis of toxic and non-toxic comments present in the data-set. Figure 2 provides the count of non-toxic and toxic comments found in the available labeled data-set considering comments with 'target' value greater than or equal to 0.5 to be toxic.

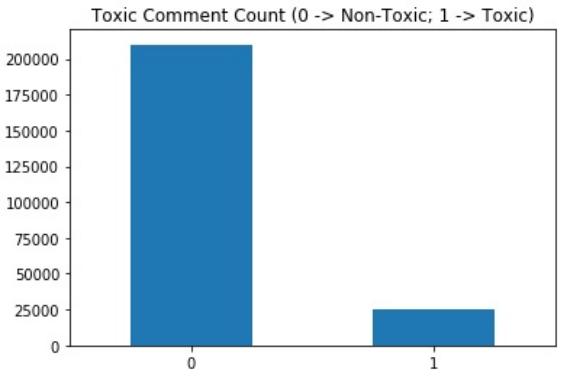

Figure 2: Very small ratio of available data is found to be toxic comment as per the label provided in the data-set.

Further, we have pre-processed the data. After some initial analysis and cleaning, we have tokenized each comments and stemmed the words from the comments. This provides the vocabulary and its actual length from the available comments in the data-set. We have then represented each word using the TF-IDF Vectorizer to compute the score for each comment. These values along with the available label is fed to SVM classifier for binary classification task. We would be considering

this as our baseline model to compare the further implementations.

5 Methods and Experiments

We experimented with various models as discussed in the earlier part of paper by using word embeddings to create feature vectors of words and passing this feature vector to different classifiers as input to the model. Our goal was to experiment with various word and document representation methods combined with the different classifiers available over this corpa and thus maximizing the chances of detecting a toxic comment. To make comparison across the models, we evaluated these models on the development data using precision, recall and F1 score and finally opted the best performing model on dev for test run. As mentioned in (Zhang and Luo, 2018), we plan to use the macro-average F1 score to evaluate our models as it is more indicative than the micro-average F1 score for imbalanced data sets. We have spliited the train data-set in the ratio of 80:10:10 as train, dev and test set for training and testing purpose.

Considering **TFIDF** as a well known method to covert textual data to numeric form, we have initially experimented with the TFIDF uni-gram values. TFIDF takes 2 information 'Term Frequency' and 'Inverse Document Frequency' to get the numeric form of the text data. The Inverse Document frequency provides the importance of each term in regards to its importance in the given sentence or document.

5.1 Support Vector Machine (Baseline)

Considering SVM as the best model for binary classification, we built our baseline with the inbuilt sklearn SVC classifier. First, we converted the input textual comment data into feature vectors using the inbuilt TfIdf vectorizer to pass as input to the SVC model. Our baseline model didn't performed great with this data, mainly because of the imbalanced toxic and non-toxic comments in the data-set. We got a F1 score of 47% and an accuracy of 89% with this input data for model. So, we decided to use under-sampling techniques to tackle the issue of imbalanced data. We used RandomUnderSampler here, which is used to perform random under-sampling. It under-samples the majority class(es) by randomly picking samples with or without replacement. In our case, it reduced the majority class so as to balance the data-set for toxic and non-toxic comments.

5.2 SVC with UnderSampling

We used the same SVC model with the under-sampled data-set and found that it performed better than SVC model alone with 59% F1 score keeping the other factors same as in above model.

5.3 Multinomial Naive Bayes using TFIDF

Using the same TFIDF vectorizer's feature vector input, as passed to SVC and using UnderSampling, we trained Multinomial Naive Bayes model and found that it gave a better F1 score of 60%, a score higher than any of the earlier models result.

5.4 Logistic Regression using TFIDF

Again, using the same TFIDF vectorizer's feature vector input, as passed to SVC and using UnderSampling, we trained and built a new model using Logistic Regression and observed that we achieved a F1 score of 70%, which is still higher than any of the earlier models we created. Although, we achieved a higher score but a F1-score of 70% is still not a better result, so we thought to use a different approach for our feature vector creation and train the model.

5.5 Logistic Regression using Distributed Representation

Referring the results from the work of Le (Le and Mikolov, 2014a), we experimented with inbuilt Doc2Vec representation for each comments as a document to be converted into vector of 50 dimensions. Here, we analyzed the Distributed Bag of Word and Distributed Memory model by passing its output to Logistic Regression as input feature vectors. We opted to use Logistic Regression because this classifier out-performed all the models created earlier by us. These trained models resulted in a F1 score of 55% and 47%, respectively. We experimented further by combining both the above feature vectors together to build a 100-dimension feature vector in hope of increasing the score. This increased the accuracy result to 90% however, we still only had F1 score of 56%, which is still low.

5.6 Neural Networks using Word2Vec Representation

To further build a high performing model for toxic comment detection, we experimented with various

Experimented Model Name	Results				
	Accuracy	Precision	Recall	F1	ROC AUC
TfIdf + SVM without UnderSampling	89	45	50	47	50
TfIdf + SVM with UnderSampling	89	67	57	59	56
TfIdf + Multinomial NaiveBayes with UnderSampling	72	61	77	60	77
TfIdf + Logistic Regression with UnderSampling	84	67	81	70	80
Doc2Vec -> Distributed Bag of Word + LR	92	73	54	55	54
Doc2Vec -> Distributed Memory + LR	89	45	50	47	50
DBOW + DM + LR	90	72	54	56	55
Word2Vec + Neural Network (MLP)	86	86	86	86	86
Word2Vec + BiLSTM	50	25	50	33	43
Word2Vec + CNN (Bigram)	87	87	87	87	87

Figure 3: Results showing Comparison between Various Predictive Models based on Accuracy, Macro-average Precision, Recall and F1; and ROC AUC Scores

neural network models using word embedding created over our corpa. To handle the imbalanced data here, we randomly dropped non-toxic comment rows from the data-set such as to make 50:50 ratio of toxic and non-toxic comments, reducing the total number of comments to 288K. The examples provided by Keras (keras) were used as reference.

The **word embedding** used in these models are created using inbuilt Gensim Word2Vec model over the given comments corpa, considering only the training and development corpa as a whole. We applied Continuous Bag of Word and Skip-grams, both with window of size 2 and trained over 40 epochs. Each of them created a vocabulary size of 64257 and 435044632 features vectors of 100 dimension using continuous bag of word and skip-grams. This word embedding is used as input to all the classifier models built below.

Multi-Layer Perceptron

As a base to our Neural Network models, we create a simple multi-layer perceptron model, using our word embedding at embedding layer, which is flattened and then passed down to 2 dense layer with activation relu and 64 units to process the word embeddings, each followed by a dropout of 5%. Finally, we have the dense layer with sigmoid activation as we have used binary-crossentropy loss to measure the loss of the model. This model resulted a better accuracy along with better precision, recall and F1 score of 86%.

BiLSTM

The built word embedding matrix over our corpa is passed at embedding layer to this model, and before passing the word embeddings to Bidirectional LSTM layer of unit 70, we made a dropout of 10% before, and also after the Bidirectional LSTM layer. Finally, output from the Bidirectional LSTM layer with dropout is passed down to dense layer with sigmoid activation to get the binary cross-entropy of the model. We expected this model to perform better than MLP, however we had a lower score than MLP.

Convolutional Neural Network

So, in order to further improve our F1 score using our word embedding, following the results by Zhang (Zhang and Wallace, 2017), we built a CNN model to do CNN computation over bigram using kernel size of 2. So, we first passed our word embedding through the embedding layer following a dropout of 2% it is passed to Conv1D layer with relu activation. This layer performs the computation over bigram word vectors and outputs a feature vector of size 310*100. We than reduce the 310*100-D matrix to 100-D 1-dimensinal vector using GlobalMaxPooling layer, which takes the maximum value from each filter as output. This output is then directly passed to dense layer with sigmoid activation to calculate binary cross-entropy loss. Here, we observed that along with the increase in F1 score, we also got higher accuracy of 87% and precision, recall values of 85% and 90%.

6 Results and Conclusion

By making a comparison over these predictive models for the macro-average precision, recall and F1 score, we achieved a higher F1 score for CNN model over bigram computation. Our word-level

CNN model out-performed all the other models including the Logistic Regression using TFIDF, Gensim Doc2Vec and other Neural network models. A comparative result is shown in the Figure 3, where we can observe that the Convolutional Neural Network with bigram computation using word-level word embedding has achieved a higher F1 score of 87%. We further tested our best model CNN over the test data and also achieved a F1 score of 87.4%. So, using word-level embedding represented by the word2vec feature vectors, along with bigram computation has increased the overall performance of the model.

7 Challenges

Based on our data visualization as in Figure 2, we observe high imbalance between non-toxic and toxic comments. This imbalanced data-set proved little bit challenges to us in our earlier step of model creation. As, we can see above that during our earlier phase of model creation (baseline model), we got a F1 score of 47% which we can attribute to imbalanced data-set. This is because, when we used under-sampling technique as discussed above, we got an improved higher F1 score. Another challenges faced was to handle Out-of-vocabulary words i.e., words that are not present in training data words like slangs and misspellings. We used word-embeddings for this. Also, there may be the case when the toxicity of a comment depends on the earlier part of comment made which may a challenging situation in the case of longer comments(longer dependencies). We tried to handle this using Neural Networks.

8 Future Works

In our work, we have seen the use of feature vector creation using word embeddings like TFIDF, Word2vec and Doc2Vec and its impact on models performance. We can further extend this work using another word embedding called Glove (Pennington et al., 2014) over another available larger corpa to check the model's performance. GloVe is a count-based model that learns vectors or words from their co-occurrence information, i.e. how frequently they appear together in large text corpora while word2vec is a predictive one. Also, as a part of further study, we may consider the creation of feature vector using character level embeddings and how it perform in comparison to the word embeddings. We can further extend our work of CNN where we tried the model's training and evaluation only on bi-gram. We may extend it to tri-gram and check the models performance for the given corpora.

References

Betty van Aken, Julian Risch, Ralf Krestel, and Alexander Löser. 2018. Challenges for toxic comment classification: An in-depth error analysis. In *Proceedings of the 2nd Workshop on Abusive Language Online (ALW2)*, pages 33–42, Brussels, Belgium. Association for Computational Linguistics.

Ying Chen, Yilu Zhou, Sencun Zhu, and Heng Xu. 2012. Detecting offensive language in social media to protect adolescent online safety. In *Proceedings - 2012 ASE/IEEE International Conference on Privacy, Security, Risk and Trust and 2012 ASE/IEEE International Conference on Social Computing, SocialCom/PASSAT 2012*, Proceedings - 2012 ASE/IEEE International Conference on Privacy, Security, Risk and Trust and 2012 ASE/IEEE International Conference on Social Computing, SocialCom/PASSAT 2012, pages 71–80.

Maral Dadvar and Franciska de Jong. 2012. Cyberbullying detection: A step toward a safer internet yard. In *Proceedings of the 21st International Conference on World Wide Web*, WWW '12 Companion, pages 121–126, New York, NY, USA. ACM.

Nemanja Djuric, Jing Zhou, Robin Morris, Mihajlo Grbovic, Vladan Radosavljevic, and Narayan Bhamidipati. 2015. Hate speech detection with comment embeddings. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15 Companion, pages 29–30, New York, NY, USA. ACM.

Jigsaw. 2019. Jigsaw civil comment toxicity. *Kaggle*.

keras. Keras examples.

Quoc Le and Tomas Mikolov. 2014a. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, pages II–1188–II–1196. JMLR.org.

Quoc V. Le and Tomas Mikolov. 2014b. Distributed representations of sentences and documents. *CoRR*, abs/1405.4053.

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*, WWW '16, pages 145–153, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee.

Jeffrey Pennington, Richard Socher, and Christopher 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, Doha, Qatar. Association for Computational Linguistics.

A. D. Pozzolo, O. Caelen, R. A. Johnson, and G. Bontempi. 2015. Calibrating probability with undersampling for unbalanced classification. In *2015 IEEE Symposium Series on Computational Intelligence*, pages 159–166.

Akshay Uppal. 2019. Sentence classification using bi-lstm. *Medium Site*.

Dawei Yin, Brian D. Davison, Zhenzhen Xue, Liangjie Hong, April Kontostathis, and Lynne Edwards. 2009. Detection of harassment on web 2.0.

Liang-Chih Yu, Jin Wang, K. Robert Lai, and Xuejie Zhang. 2017. Refining word embeddings for sentiment analysis. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 534–539, Copenhagen, Denmark. Association for Computational Linguistics.

Ye Zhang and Byron Wallace. 2017. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 253–263, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Ziqi Zhang and Lei Luo. 2018. Hate speech detection: A solved problem? the challenging case of long tail on twitter. *CoRR*, abs/1803.03662.