# Toxic Text Classification

Abhishek Rapelli
arapelli@iu.edu

Pramod Duvvuri
vduvvuri@iu.edu

***Abstract*— In this paper, we write in detail about our efforts to build a predictive model that can classify if a given line of text can be considered as toxic or not. We used various machine learning and deep learning algorithms to handle this text classification problem. We train our algorithms with data that has been manually labeled by a community of users. We also propose a novel method that could be used to tackle this problem in a different way. We have tried both binary classification and multi-class classification. The dataset we have is highly imbalanced with the toxic to non-toxic being in the ratio of 1:10. To tackle this problem we have used resampling techniques to handle this problem of unbalanced classes. To prevent overfitting we use K-fold cross-validation technique with different folds to minimize the error and also build a predictive model that generalizes well and performs as expected when newer data is encountered in the future.**

## I. INTRODUCTION

A survey conducted by the Pew Research Center in 2017 showed that four-in-ten or 40% of Americans have experienced or been subject to some sort of harassment or abuse online. The main objective of our project is to build a predictive that could detect and flag content not suitable to be on the internet. We approach this task initially by using the traditional approaches of text classification on our labeled dataset. We can discuss the possibility of using a different approach to tackle this problem.

The traditional approach to text classification has always been centered around using the Continuous Bag of Words (CBOW) model or the Text Frequency-Inverse Document Frequency (TFIDF) model. We decided to explore these approaches to build a working predictive model. This text classification problem is often mostly popularly known as *Hate Speech Identification*. All the previous research mostly involves using traditional supervised machine learning algorithms for this classification task. Any progress in this area has been severely limited by the amount of labeled data we have publicly available to us. If we had more data we could use the latest state-of-the-art deep learning architectures such as Recurrent Neural Networks (RNN) and Long Short-Term Memory (LSTM). Nevertheless, we still explore the idea of using deep learning architectures for this classification task. We had about 160,000 rows in our dataset and 7 labels. The data section describes our dataset in more detail.

## II. DATA

The dataset we primarily used was from Kaggle in a challenge aptly named as *Toxic Text Classification Challenge* [1]. The dataset contains about 160,000 rows and six features with binary values. This is a manually labeled dataset, the text is comments from Wikipedia edits and the labels were manually done by a community of users. Each line of text belongs to one of the class and can also have multiple labels.

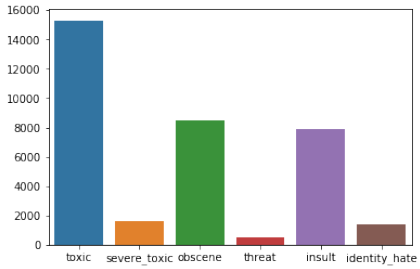


Fig. 1. Class Distribution

The above figure describes the initial class distribution in our dataset. The six labels are TOXIC, SEVERE_TOXIC, OBSCENE, THREAT, INSULT, IDENTITY_HATE. The next section describes how we have pre-processed the data before using it to train our algorithms. No information has been made public on how the community of users labeled the text.

### A. PRE-PROCESSING

The data pre-processing plays a crucial role in any machine learning project. As mentioned above, each line of text in the dataset could belong to a single class or multiple classes. A comment which is severely toxic is also toxic. So as a part of pre-processing, we combined these two features into a single feature called toxic. Then we combined the five total features now into a single feature in order for us to fit a predictive model that could

do multi-class classification and predict given the input line of text if it belonged to any of the above-mentioned classes or if the text was non-toxic. The input text contains stop-words and punctuation symbols but these were not removed. Stop words mainly consist of the articles *a, an, the*. These were not removed because we believe these symbols might help the model understand the context better since they help us humans understand the text.

## III. PROPOSED METHOD

The below figure illustrates the general flow used in our project. After the text has been pre-processed we then first attempt to perform binary classification for each class and find the best machine learning algorithm for this binary classification task. We then plot two confusion matrices a non-normalized and a normalized to view the classification results of the best predictive model.

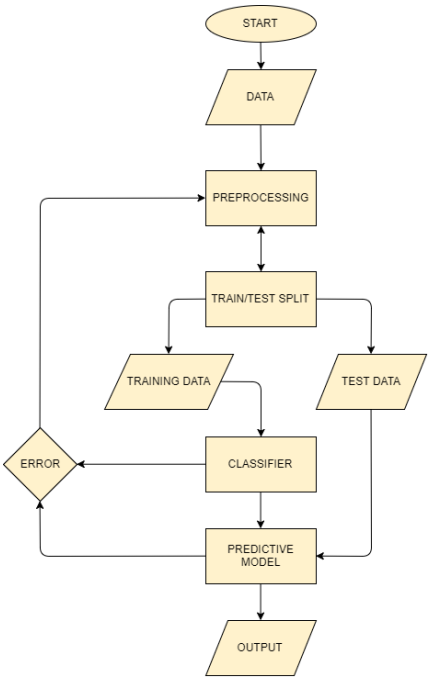


Fig. 2. Flow Diagram

### A. Machine Learning

In machine learning algorithms we have used all the state-of-the-art supervised algorithms. The algorithms we have tried are: Naive Bayes, Logistic Regression, Random Forest, Support-Vector-Machines (SVM), SVM with Stochastic Gradient Descent, XGBoost (Gradient Boosting). We have used all the open-source implementation of these algorithms available in Python [5] using the scikit-learn [3] machine learning framework. For the task of binary classification, the best algorithm we found was *Logistic Regression*. The accuracy results obtained for each algorithm have been tabulated and shown in the results section. Also for each class, we have fitted a model to perform binary classification and the algorithm that performs the best on average was *Logistic Regression*. The results of this class level binary classification have also been tabulated in the results section. The heavy imbalance in the data leads to very high accuracy rates so we wanted to use resampling to observe how these accuracy rates vary.

To resample the data we observed the number of rows in each sample and found that the class imbalance ratio in terms of toxic to non-toxic classes is about 1:10 so we wanted to reduce this imbalance to about 1:2 or 1:3. So we wrote up a function that resamples accordingly. The labels are text classes so they are an index from 0-5 before they are given as input to a machine learning algorithm. We then perform multi-class classification using this data and again we use all the supervised classification algorithms to find the best one. The accuracy of these algorithms is tabulated in the results section.

### B. Deep Learning

In Deep learning section, we used recurrent neural networks (RNNs) because they can handle varying lengths of input and output and use previous states memory to process the current states. Such sequential processing is highly useful for natural language processing problems, where the context needs to be carried forward to process a current word or sentence. This makes them best suitable for both speech and text analysis. [6] However since simple RNNs suffer from long-term dependency problem due to lengthy text size, we went for LSTMs, which overcome this problem by controlling the information flow, based on cell states. [7]

In the data preprocessing stage, we used Python's NLTK package [8] for stemming, lemmatization and removal of stop words from the text apart from another basic text cleaning like removing apostrophes, punctuation, and symbols. The text was then vectorized.

The LSTM architecture we developed consists of three layers. The first layer is the embedding layer, that takes an input of word vectors and embeds these word sequences into a vector space of a given size. [9] The

second layer is the Global Average Pooling layer, which downsamples the vector representation of text from the first layer. This helps in dimensionality reduction resulting in decreased complexity and computation. [10] The third layer is a dense output layer that outputs the probability of each of the six classes. The activation function used for this layer was sigmoid. [20] The model was trained using Python's Keras package [4] on a train-test split ratio of 4 to 1 on 20 epochs. The validation error got converged for this setting at an average accuracy of 99% and a log-loss of about 0.296 for cross-validation. However, the high accuracy rate may be attributed to the imbalance in the data and lack of considerably large size of data.

## IV. RESULTS

The below tables show the different models and classification accuracies we obtained for different data. For k-fold cross validation we mostly used five and ten folds and since there was not much difference between them we reported our accuracies for 5-folds only.

TABLE I

BINARY CLASSIFICATION RESULTS

| Algorithm | Technique | Folds | Accuracy |
|---|---|---|---|
| Naive Bayes | TFIDF | 5 | 93.0 % |
| Logistic Regression | TFIDF | 5 | **95.7 %** |
| Random Forest | TFIDF | 5 | 93.9 % |
| SVM | TFIDF | 5 | 90.3 % |
| XGBoost | TFIDF | 5 | 94.1 % |

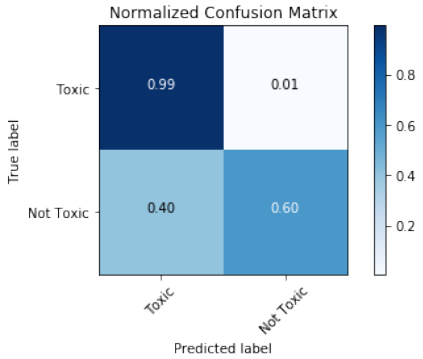


Fig. 3. Confusion Matrix for Binary Classification

Although the data in real-world scenarios is most likely to imbalanced this suggests that the accuracy of toxic comments being misclassified as not-toxic is highly unlikely but the confusion matrix does indicate how non-toxic comments are more likely to be misclassified as toxic and this does reflect well on our predictive model. We need to be able to tackle this problem mainly.

TABLE II

CLASS-WISE BINARY CLASSIFICATION RESULTS

| Algorithm | Technique | Class | Accuracy |
|---|---|---|---|
| Logistic Regression | TFIDF | toxic | 95.6 % |
| Logistic Regression | TFIDF | severe_toxic | 98.9 % |
| Logistic Regression | TFIDF | obscene | 97.8 % |
| Logistic Regression | TFIDF | threat | 99.7 % |
| Logistic Regression | TFIDF | insult | 97.2 % |
| Logistic Regression | TFIDF | identity_hate | 99.2 % |

TABLE III

MULTI-CLASS LABELS

| S.No | Label | Index |
|---|---|---|
| 1 | NON-TOXIC | 0 |
| 2 | HATE | 1 |
| 3 | INSULT | 2 |
| 4 | OBSCENE | 3 |
| 5 | THREAT | 4 |
| 6 | TOXIC | 5 |

TABLE IV

MULTI-CLASS CLASSIFICATION RESULTS

| Algorithm | Technique | Folds | Accuracy |
|---|---|---|---|
| Naive Bayes | TFIDF | 5 | 86.1 % |
| Logistic Regression | TFIDF | 5 | 87.1 % |
| Random Forest | TFIDF | 5 | 78.8 % |
| SVM with SGD | TFIDF | 5 | **88.1 %** |
| XGBoost | TFIDF | 5 | 81.4 % |

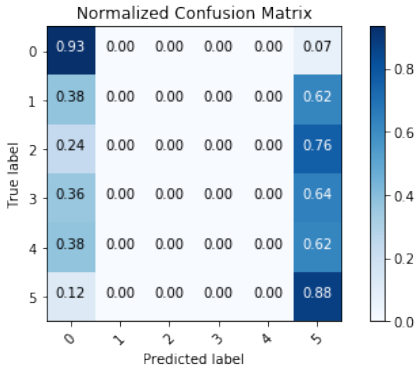


Fig. 4. Confusion Matrix for Multi-Class Classification

After resampling we observe the accuracies have decreased almost 10 percent. But by observing the confusion we still that the problem of misclassification still persists even after resampling the data.

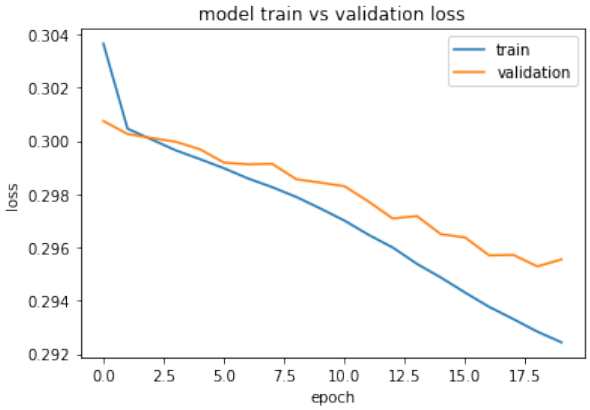


Fig. 5. Three layers LSTM model log-loss against Epoch

The log-loss is calculated based on the categorical cross entropy. We can observe from the plot that the log-loss drastically reduces by Epoch number for both training as well as cross-validation data. The convergence is observed at this setting with cross-validation error settling at 0.296 approximately.

## V. CONCLUSIONS

The results are indeed encouraging and do indicate that building a predictive model for solving the problem of detecting such hate speech is possible. It is important to note that the accuracy of models do increase with the removal of stop-words but for our project, we chose to include stop-words as a part of the data pre-processing. This accuracy increase is about 2-3 percent in each model. As we collect more and more data from social networks we also show that deep learning models can help us in achieving our task of text classification with high accuracy levels. The final code along with the data has been uploaded to GitHub **here**.

There are still many challenges which prevent us from solving this problem of text classification. Some of these problems are still being actively researched upon by the academia who work on Natural Language Processing. Some of the challenges are:

- The amount of data we have for this task. We mentioned that all the data used for solving the problem of *Hate Speech Identification* has been manually labeled by a community of users and only a limited amount is made publicly available and this makes collective progress difficult in achieving the goal
- The class imbalance is another issue and we need datasets that are more balanced
- How do we make a model understand the fine line of difference between offensive and hate speech
- We can go for advanced natural language processing like dependency parsing, which can carry context at a far lesser error rate. Stanford's CoreNLP could be used, but this would be effective only when we have large scale data. [21]

## VI. FUTURE WORK

We propose that this detecting such behavior could be extended to audio data. The challenge is to find audio data which can help us build such a model. Social networks companies such as *Google, Facebook* are currently working on building smart AI systems that can detect and flag such content for moderation.

We propose the usage of *Knowledge Graphs* [19] generated from comments that have been labeled as hate speech only to tackle this problem. We looked for any papers that used knowledge graphs for this hate speech identification task and could not find any. Hate speech identification was seen as a keyword detection task but knowledge graphs would help us reduce the number of keywords we can search for before we could label text as hate or no-hate. This idea of using knowledge graphs for this problem because we were discovering about knowledge graphs as a part of another coursework this semester. We have used our data and imported into the ElasticSearch, Logstash, and Kibana (ELK) [16] stack to visualize the common words that occur frequently in each class. This was done mainly to identify and discover patterns among the words. Kibana helps us generate graphs which are undirected and have weighted edges. This helps us discover words that occur more commonly in each of the above-mentioned classes when we are looking to identify if the given line of text is hate or no-hate. The graphs are not being included in the report because mostly they contain abusive language but we shall try to provide images along with our code.

## VII. ACKNOWLEDGMENTS

We would like to thank Professor Dr. Minje Kim for his feedback on our proposal and also during the poster presentation session. We also would like to thank the Associate Instructors of this class who gave us their feedback during the poster presentation. We would like to thank Kaggle for hosting this challenge and for providing us with the dataset. As we mentioned all the data we have had regarding this hate speech classification task has been manually labeled so we would like to thank the community of users who took time to manually label this dataset which helps in making progress towards making the internet a better

place for everyone to use. We also would like to the open source communities for contributing and maintaining such optimized machine learning and deep learning frameworks such as Scikit-learn and Keras.

## REFERENCES

[1] https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/data
[2] http://www.pewinternet.org/2017/07/11/online-harassment-2017/
[3] https://scikit-learn.org/stable/
[4] https://keras.io/
[5] https://www.python.org
[6] https://bit.ly/2PDDACk
[7] https://bit.ly/2C43TOI
[8] https://www.nltk.org/
[9] https://bit.ly/2qlHUM7
[10] https://qr.ae/TUtgfm
[11] Detecting Hate Speech and Offensive Language on Twitter using Machine Learning: An N-gram and TFIDF based Approach by Gaydhani, Doma, Kendre and Bhagwat
[12] Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. In Proceedings of ICWSM
[13] Shervin Malmasi and Marcos Zampieri. 2017. Detecting Hate Speech in Social Media. In Proceedings of the International Conference Recent Advances in Natural Language Processing (RANLP), pages 467472
[14] Del Vigna, Fabio & Cimino, Andrea & Dell'Orletta, Felice & Petrocchi, Marinella & Tesconi, Maurizio (2017) Hate me, hate me not: Hate speech detection on Facebook
[15] A Web of Hate: Tackling Hateful Speech in Online Social Spaces by Saleem, Haji Mohammad; Dillon, Kelly P; Benesch, Susan; Ruths, Derek
[16] https://www.elastic.co/
[17] https://www.elastic.co/products/kibana
[18] https://www.ontotext.com/knowledgehub/fundamentals/what-is-a-knowledge-graph/
[19] https://en.wikipedia.org/wiki/Knowledge_Graph
[20] Pengfei Liu, Xipeng Qiu and Xuanjing Huang. Recurrent Neural Network for Text Classification with Multi-Task Learning
[21] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. The Stanford CoreNLP Natural Language Processing Toolkit