Toxic Comment Classification

Isha Pandya - ipandya
Meenakshi Madugula - mmadugu
Suraj Siddharudh - ssiddha
Prashanthi Kanniappan Murthy - pkannia

1 Background

1.1 Problem Statement

Keeping online conversations constructive and inclusive is a crucial task for platform providers. With a rise in the number of online platforms for people to interact and share their opinions, dangers involved in it have also increased. Online texts with high toxicity can cause personal attacks, online harassment and bullying behaviors. In addition, new regulations in certain European countries have been established enforcing to delete illegal content in less than 72 hours [1]. Because the comments sometimes may be abusive, insulting or even hate-based, it becomes the responsibility of the hosting organizations to ensure that these conversations are not of negative annotation.

The goal of this project is to analyze a dataset consisting of comments from Wikipedia's talk page edits and label them for toxicity based on various categories like toxic, severe toxic, obscene, threat, insult, and identity hate.

We aim at correctly identifying toxic comments so that they can be blocked and the user posting them could be discouraged from doing so. Thus, our focus is on minimising false positives (since we cannot penalise users posting harmless comments).

1.2 Literature Survey

We referred quite a few papers dealing with different aspects of the problem statement we are addressing. Before even thinking about suitable models for data training, we came across this paper about challenges for toxic comment classification[2]. It highlights the problem with false positives which is a crucial issue for our project as well. They mainly arise due to Usage of swear words in a positive annotated comment, references to toxic or hateful language and idiosyncratic, rare words in actual non-hateful comments.

Next for selection of appropriate models we studied various papers dealing with comparison of language models with others. We resonated with their goal to measure the gains in performance with more intensive training[8]. The paper [8] demonstrates how a language model (Recurrent Neural Network Language Model) in their case works better than any other models like SVM or Naive Bayes. We thus adopt the same approach and implement BERT language model [4] to check if it outperforms other models. In order to check for best performance, we combine the simple models into a complex ensemble model following the approach of Pete Burnap and Matthew L. Williams[3] which they claim to improve macro-averaged F1-measure especially on sparse classes and data with high variance.

2 Proposed Method

We aim to compare the performance of a Language model with standard classifiers like Naive Bayes, Logistic Regression and Support Vector Machine. The following sections explain each model in detail.

For Naive Bayes, Logistic Regression and SVM, we use OneVsRest strategy. OneVsRest strategy can be used for multi-label learning, where a classifier is used to predict multiple labels for instance. The Multi-label algorithm accepts a binary mask over multiple labels. The result for each prediction will be an array of 0s and 1s marking which class labels apply to each row input sample.

2.1 Naive Bayes

Naive Bayes classifiers are a family of simple probabilistic classifiers based on applying Bayes' theorem with strong independence assumptions between the features. They are among the simplest Bayesian network models.

Naive Bayes supports multi-class, but we have a multi label scenario. So we wrap Naive Bayes in OneVsRestClassifier to obtain multi-label results.

The Bayes classifier is a starting point of modeling the text dataset, however, the assumption of the model is based on that all the variables are independent, which is impractical in real life language context cause the words in a sentence would be likely to have an effect on each other.

Implementing Occam's razor principle and thus choosing the simplest model we created a Naive Bayes classifier to get a baseline performance for our data set. This would work as a starting point for comparative analysis.

2.2 Logistic Regression

Regression analysis is a form of predictive modelling technique which investigates the relationship between a dependent (target) and independent variable (s) (predictor). This technique is used for finding the causal effect relationship between the variables.

Logistic Regression is a Machine Learning classification algorithm that is used to predict the probability of a categorical dependent variable. In logistic regression, the dependent variable is a binary variable that contains data coded as 1 (yes, success, etc.) or 0 (no, failure, etc.).

In natural language processing, logistic regression is the baseline supervised machine learning algorithm for classification, and also it has a very close relationship with neural networks, hence we decided to see how it performs in identifying the toxicity.

2.3 Support Vector Machine

We plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiate the two classes very well. Support Vectors are simply the co-ordinates of individual observation.

Support Vector Machine is a frontier which best segregates the two classes(hyper-plane/ line) and is considered to be very robust when it comes to solving classification problems.

SVM has shown success to information retrieval problems, particularly text classification. It is simple yet a powerful classifier that might work well for our data which appears to be linearly inseparable. Hence we decided to check its performance.

2.4 Ensemble

Ensemble learning uses multiple learning algorithms to improve machine learning results by combining several models. This approach allows the production of better predictive performance compared to a single model.

After the above 3 models are done with default configuration, different weights to each model was assigned and the optimum weight was found using hyper parameter tuning using grid search.

2.5 BERT

Proper language representation is key for solving the task at hand. Context-free models such as word2vec or GloVe generate a single word embedding representation for each word in the vocabulary.

Table 1: Training Data Set

Labels	Label = True
toxic	15294
severe_toxic	1595
obscene	8449
threat	478
insult	7877
identity_hate	1405

Table 2: Testing Data Set

Labels	Label = True
toxic	6090
severe_toxic	367
obscene	3691
threat	211
insult	3427
identity_hate	712

Contextual models instead generate a representation of each word that is based on the other words in the sentence. Language models typically require large amounts of data to achieve a decent performance, but there are currently no large-scale datasets for abuse detection. To overcome this challenge, we exploit the power of BERT, a contextual model capturing relationships among words in a bidirectional way.

BERT is a language model released by Google[4]. It is pre-trained on Book Corpus[9] and Wikipedia[7] database on two tasks - next word prediction and next sentence entailment. We can fine tune the model with our training data since it is readily available. The fine-tuning code needs to be modified to support the current multi-label dataset. The review's text is given as input to the model. They are passed through 12 layers of encoders. Finally the last layer logits of the special CLS token are passed through a classifier (linear in our case). The classifier's outputs are passed through an argmax function used to predict the final result per label.

3 Plan & Experiment

3.1 Dataset

The data set used in this project is from a Kaggle Competition sponsored by Google and Jigsaw [5]. The data set contains 2 files, one each for training and testing. Each of data point contains comments with their binary labels for the following toxicity: toxic, severe_toxic, obscene, threat, insult, identity_hate.

The training data set has 160k data points and the testing data set has 64k data points, of which NaN and empty comments were dropped. After data pre-processing, the data set distribution is depicted in Table [1], [2].

3.2 Pre-Processing

The data contains verbose comments which needed to be scrubbed so that they could be used for training models. We have taken following steps in order to clean our data before classification:

- Non ASCII Removal: To standardize the encoding of the comments, non Ascii characters were removed using the regex module of python.
- Tokenization: It is the process of converting the data into tokens before forming the vectors. An article consisting of paragraphs and sentences, is converted to an array of words. This was done using NLTK's Tokenize library.
- Lowercase: We converted all the characters to lower case so that the same words and stop words can be recognized.
- Stop Words: Commonly occuring words are irrelevant and do not affect the contextual meaning of the comment and can be removed without affecting performance. Figure 1 depicts that the frequency distribution of the data set is pre-dominated with stop words. Hence, they were removed as seen in Figure 2. Using NLTK's stopwords corpus we removed these from our tokens. e.g. "the","is","at" etc.
- Lemmatisation: It is the process of grouping together the inflected forms of a word so they can be analysed as a single item, identified by the word's lemma, or dictionary form. We use NLTK's WordNetLemmatizer library to lemmatize our words (e.g. "better" -> "good")

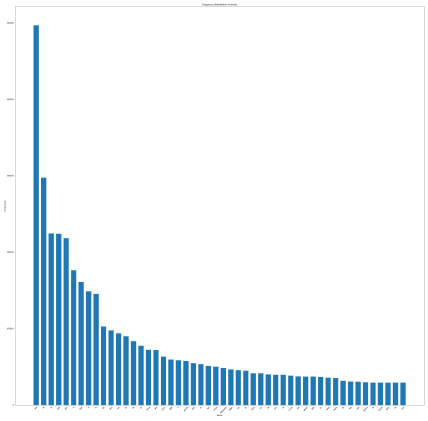

Figure 1: Frequency dist before preprocessing

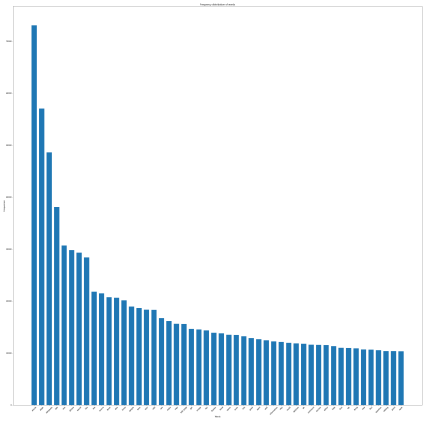

Figure 2: Frequency dist after preprocessing

- Stemming: It is the process of reducing the words to their base form and removing morphological affixes, leaving only the word stem. We use NLTK's PorterStemmer library to stem our words. (e.g. "running" -> "run")

3.3 Hypothesis

Based on our prior motivation, we hypothesize the following:

- Potential negative comments can be filtered by identifying the types of toxicity of the comment.
- A pre-trained language model identifies toxicity of comments better than the standard classifiers.

3.4 Experimental Design

The main goal while designing our experiment is to filter out toxic comments with high Precision and reduce falsely filter non-toxic comments. For this project, we define that any false positives might have negative impact on the user's interaction with the system. To achieve this, we focused on Precision in the model results. A high precision represents that there is high correct predictions among all the predictions. We also focus on AUC-ROC score as the dataset is sparse and highly imbalanced.

We used NLTK, an open-source libraries for building Python programs to work with human language data. It provides a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning. We have used this library for preprocessing our data.

As the data set is made of strings, and to apply learning models, we need to convert data into numerical feature. To achieve this, we used TFIDF Vectorization. TFIDF [6] tells the importance of the word in the corpus or dataset. TF boosts the relevance of the words that occur multiple times in a document as it should be more meaningful than other words that appear fewer times. At the same time, if a word occurs many times in a document but also along many other documents, maybe it is because this word is just a frequent word; not because it was relevant or meaningful (IDF).

As the data set being used is a multi-label, many of the models are not designed to handle them. For this reason, we trained our models using OneVsRest classifier as described in the previous section for each of the label and calculated the metrics. As BERT supports training multi-labels at a time, we used multi label classification.

Table 3: Naive Bayes results

Labels	Precision	Recall	Accuracy	F1 Score	AUC-ROC
toxic	0.2	0.91	0.92	0.33	0.60
severe_toxic	0	0	0.99	NaN	0.50
obscene	0.11	0.97	0.94	0.20	0.55
threat	0	0	0.99	NaN	0.50
insult	0.05	0.94	0.94	0.09	0.52
identity_hate	0.31	0.63	0.99	0.42	0.66

Once the Naive Bayes, Logistic Regression, SVC models are run with the default configuration on the data set, we applied hyper parameter tuning using grid search (with an increment of 0.1) to find the optimum weights for each of the model to increase the precision. In this process, we tried to find the optimum weights for Naive Bayes, Logistic Regression, SVC and threshold. The optimum values were then used to ensemble these 3 models.

Further, to support our hypothesis we used a language model, BERT [4] to train our data set. We used the BERT base uncased model for our experiments. It has 12 layers with a hidden size of 768. Hyper parameter tuning was applied to select the optimal sequence length and batch size. The product of batch size and maximum sequence length has to be 4096. BERT was run with varying sequence lengths of 32, 64, 128 and 256. It was observed that best results were obtained using a sequence length of 128. The model was fine tuned for 3 epochs which is generally suggested.

For each of the models trained we evaluated Confusion Matrix. From confusion matrix, we then evaluated Accuracy, Precision, Recall, F1 Score, AUC (Area Under The Curve) - ROC (Receiver Operating Characteristics). As discussed earlier, our major focus was on precision and AUC-ROC score.

4 Results

The metric performances of Naive Bayes, Logistic Regression, SVM, Ensemble and BERT are shown in Table - 3, 4, 5, 6, 7 respectively.

As seen in the tables, performance of toxic label is better than the other labels. We attribute this behaviour to the presence of more positive samples in toxic label. This shows that the performance can be improved if the dataset had more positive samples for other labels as well.

All observations are made focusing on the results of toxic label (due to the presence of more positive samples) for Precision and AUC-ROC score (as described above). Naive-Bayes has a precision of 0.2 and clearly the independence assumption does not work. It is worse than randomly predicting a class label. Logistic Regression drastically improved the performance over Naive Bayes by 50%. We think this is because Naive Bayes optimizes a generative objective function, while logistic regression optimizes a discriminative objective function. Hence, a discriminative classifier is more suitable for our problem. SVM's precision jumps by 9% over Logisitc regression. This is because SVM tries to maximize the margin between the closest support vectors while LR the posterior class probability. Thus, SVM finds a solution which is as fare as possible for the two categories and it works for our data set.

As observed from tables 5 and 6, there is not much performance improvement for the ensemble model over SVM. This can be explained as SVM has highest weight after hyper parameter tuning. The weights for ensemble model are 0.2, 0.6 and 0.8 for NB, LR and SVM respectively.

As seen in Table 7, BERT's outperforms the strongest individual method (SVM) by approximately 14 percent precision and 5 percent AUC-ROC score for toxic label. BERT's precision is doubled for labels threat and identity_hate, this finding shows that the language model is preferred even for sparse labels. An explanation for this is that the pre-trained model is able to differentiate what words contribute to the toxicity of the comment. This finding is accompanied by Figure 4 which compares the performance metrics of all the models. From the figure, it can be clearly observed that BERT out performs the standard classifiers in all the metrics.

Table 4: Logistic Regression results

Labels	Precision	Recall	Accuracy	F1 Score	AUC-ROC
toxic	0.69	0.66	0.93	0.68	0.82
severe_toxic	0.32	0.39	0.99	0.35	0.65
obscene	0.60	0.80	0.96	0.68	0.80
threat	0.19	0.52	0.99	0.68	0.60
insult	0.51	0.73	0.96	0.60	0.75
identity_hate	0.33	0.63	0.99	0.43	0.66

Table 5: Support Vector Machine results

Labels	Precision	Recall	Accuracy	F1 Score	AUC-ROC
toxic	0.78	0.59	0.93	0.67	0.86
severe_toxic	0.34	0.37	0.99	0.35	0.67
obscene	0.67	0.69	0.96	0.68	0.82
threat	0.31	0.52	0.99	0.40	0.66
insult	0.57	0.67	0.96	0.61	0.78
identity_hate	0.32	0.63	0.99	0.42	0.66

5 Conclusion

In this project, we experimented multiple approaches for toxic comment classification. The problem is difficult as the dataset is highly imbalanced. Moreover, there is no simple standard pattern in the comment texts to identify toxicity. Since, we plan to filter comments based on toxicity, it is challenging to differentiate false positive. For example a comment such as "I'm a stupid person. Sorry friend." is not offensive, but due to the presence of a swear word it may be classified as toxic.

All the standard classifier models (Naive Bayes, SVM, Logistic Regression) make different errors and can be combined into an ensemble with improved F1-score. The ensemble performs better on classes with less examples such as insult and severe_toxic. We find that using a pre-trained language model improves the precision and AUC-ROC score significantly. BERT especially outperforms the ensemble model even for labels with sparse examples such as threat and identity_hate. We also find that a large source of errors is the lack of consistent quality of labels. Additionally the performance of labels other than toxic can be improved if there are more positive samples in the dataset. We conclude that by using BERT hosting organisations can filter potential dangerous comments with a precision of 92 percent. Finally, we suggest further research in representing the comments in vector form - by using different word embeddings or better feature extractions methods. We hypothesize this would improve distinction between false positive contexts in standard classifiers.

References

[1] `https://www.bbc.com/news/technology42510868`.

[2] R. K. Betty van Aken, Julian Risch and A. Loser. Challenges for toxic comment classification: An in-depth error analysis. *arXiv:1809.07572v1*, 2018.

Table 6: Ensemble results

Labels	Precision	Recall	Accuracy	F1 Score	AUC-ROC
toxic	0.78	0.62	0.92	0.69	0.86
severe_toxic	0.38	0.36	0.99	0.37	0.68
obscene	0.69	0.69	0.96	0.68	0.83
threat	0.33	0.53	0.99	0.41	0.67
insult	0.60	0.66	0.96	0.63	0.79
identity_hate	0.33	0.63	0.99	0.43	0.66

Table 7: BERT results

Labels	Precision	Recall	Accuracy	F1 Score	AUC-ROC
toxic	0.92	0.60	0.92	0.72	0.91
severe_toxic	0.54	0.36	0.99	0.44	0.77
obscene	0.82	0.60	0.96	0.70	0.89
threat	0.67	0.44	0.99	0.53	0.83
insult	0.76	0.62	0.96	0.68	0.87
identity_hate	0.61	0.60	0.99	0.60	0.80

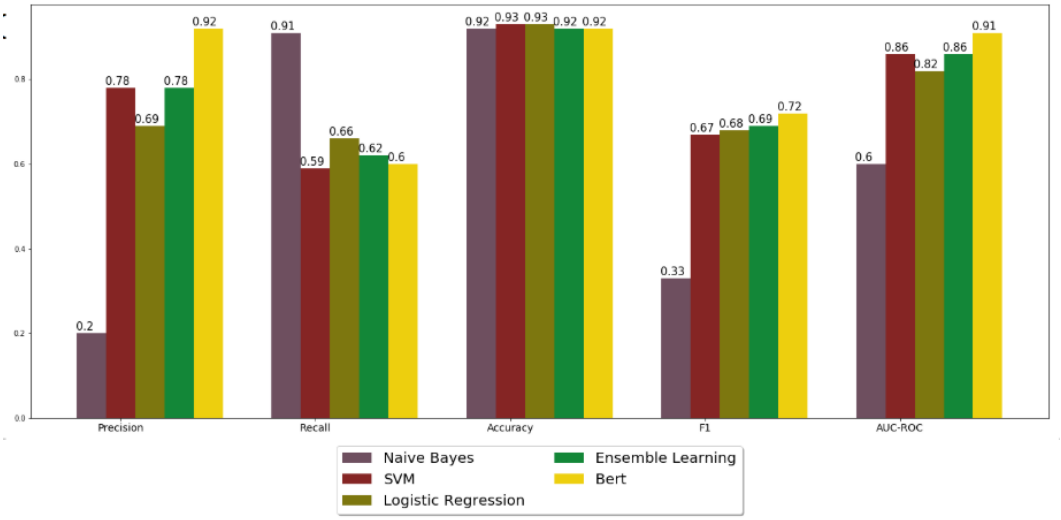

Figure 3: Comparison of toxic label performance

[3] P. Burnap and M. L. Williams. Cyber hate speech on twitter : An application of machine classification and statistical modeling for policy and decision making. *https://orca.cf.ac.uk*, 2015.

[4] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[5] G. Jigsaw. *Jigsaw Toxic Comment Classification Challenge*, 2017. `https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge`.

[6] P. Pantola. Natural language processing: Text data vectorization. 2018.

[7] Wikipedia contributors. Plagiarism — Wikipedia, the free encyclopedia, 2004. [Online; accessed 22-July-2004].

[8] J. T. Yashar Mehdad. Do characters abuse more than words? *Proceedings of the SIGDIAL 2016 Conference*, 2016.

[9] Y. Zhu, R. Kiros, R. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *arXiv preprint arXiv:1506.06724*, 2015.

Github Link: https://github.com/Suraj-Siddharudh/Toxic-Comment-Classifier