Automatic Filtering of Malicious Comments on the Twitter and Wikipedia Data

Ashish Jain
M.Tech CSE
IIIT DELHI
Okhla Phase III, New Delhi
ashish18052@iiitd.ac.in

Piyush Dhyani
M.Tech CSE
IIIT Delhi
Okhla Phase III, New Delhi
piyush18131@iiitd.ac.in

Sarosh Hasan
M.Tech CSE
IIIT DELHI
Okhla Phase III, New Delhi
sarosh18084@iiitd.ac.in

Abstract

Detection and filtering of abusive language in user-generated online comments have become an issue of importance in recent years. Websites that allow users to leave feedback causes plaguing websites with false comments which harms the online business and overall user experience. Toxic comment detection and filtering has now become the active research field with many proposed approaches. We compare different approaches on comment dataset of Twitter and Wikipedia. We will work on an ensemble based model and word embedding techniques.

1 Introduction

Our focus is the study of negative online behaviors, like hateful comments (i.e., comments that are disrespectful, obscene, rude or otherwise likely to make someone leave a discussion, stop sharing their opinion or give negative feedback to the site that may elevate the problem in online discussions and social networking sites).

So far a lot of models have been developed online. The issue that we found with models available is that they still make some error in classifying the comments on the basis of negativity, and yet we cannot filter comments on the level of how toxic the comment is actually. The threat of online malicious and obscene comments lead many people to stop giving their opinions and taking opinion from others because of such malicious comments.

2 Dataset

We take two datasets into account to investigate these errors: comments on Wikipedia talk pages presented by Google Jigsaw during Kaggles Toxic Comment Classification Challenge [1] dataset distribution see Figure 2 and a Twitter dataset by (Davidson et al., 2017) dataset distribution see Figure 1. These dataset contains different challenges. They are labeled based on different definitions of toxicity, they include diverse language from user comments and Tweets, and they present a multi-class and a multi-label classification task respectively.

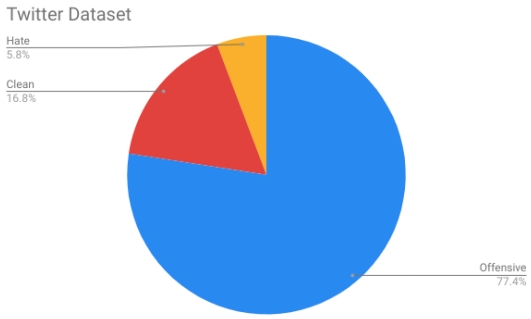

Figure 1: Twitter Data Distribution

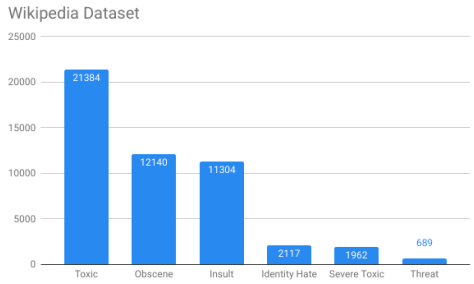

Figure 2: Wikipedia Data Distribution (Multi-Label)

3 Baseline Models

3.1 Using twitter dataset

In the unbalanced twitter dataset the Tf-idf based, Sentimental based, Flesh Reading Ease (FRES),Flesh-Kincaid (F-K) scores are used as features. FRES and FK scores indicates the score

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

for reading ease text. In FRES score high score indicates material is easier to read and less is more difficult.

$$206.835 + 1.015 * \left(\frac{TotalWords}{TotalSentences}\right) - 84.6 * \left(\frac{TotalSyllables}{TotalWords}\right)$$

FK score is US grade level to check the readability level of various books. If score is greater than ten it indicates no of years of education required to understand the text.

$$-15.59 + 0.39 * \left(\frac{TotalWords}{TotalSentences}\right) + 11.8 * \left(\frac{TotalSyllables}{TotalWords}\right)$$

The number of occurrences of each term along with idf weight (Tf-idf) used as n-gram features with other features. For classification different classifiers were used but the Logistic Regression and Support Vector Machine performs better. Out of which Logistic Regression with L2 Regularization performs best in terms of precision and recall of each class. The results with above features and Logistic Regression is shown in Figure 3 and the confusion matrix is shown in Figure 4.

3.2 Using Multi-label Wikipedia dataset

Multilabel classification problems are the problems in which an instance can assign to various categories, where we have a set of target labels. There are different methods to solve a multi-label classification problem. We use Problem Transformation method in our project. The proposed approach can be carried out in three different ways as:

1. Binary Relevance
2. Classifier Chains
3. Label Powerset

3.2.1 Binary Relevance

This technique treats each label as a separate single class classification problem and classifies the comment as per the regular classifier would.

3.2.2 Classifier Chains

In initial stage the first classifier is trained on the first input data and then in each next stage the classifier is trained on the input data and all the previous classifiers results in the chain. The previous class label is then appended as the feature in the data and then again classified as per the training model for the next label.

Previous Approach	Precision	Recall	F1 Score	Support
Hate	0.51	0.36	0.42	290
Offensive	0.92	0.96	0.94	3832
Neither	0.87	0.78	0.82	835
Macro Average	0.76	0.7	0.73	4957
Micro Average	0.89	0.89	0.89	4957

Figure 3: Twitter Data Evaluation Result Baseline

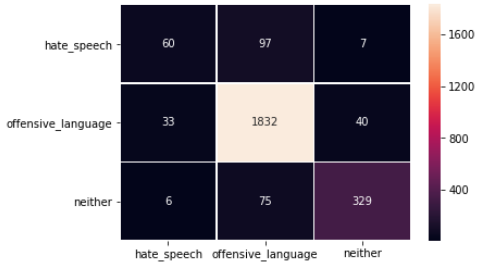

Figure 4: Confusion Matrix Twitter Data

3.2.3 Label Powerset

In this approach, we are changing the problem into a multi-class problem with one multi-class classifier is trained on all unique label that are created on the powerset of the combinations found in the training data labels. The Problem with this approach is that it takes a lot of time and is computationally costly .

The result of accuracies using different Multi-Label Classifier on tf-idf based feature are shown in Figure 5.

4 Embedding Techniques

The Embedding techniques are used to convert the text document to word vectors or forming a vector representation.

4.1 Word Embedding Techniques

Word embedding is a mapping technique that allows words with similar meaning to have similar representation. Here we may try different models

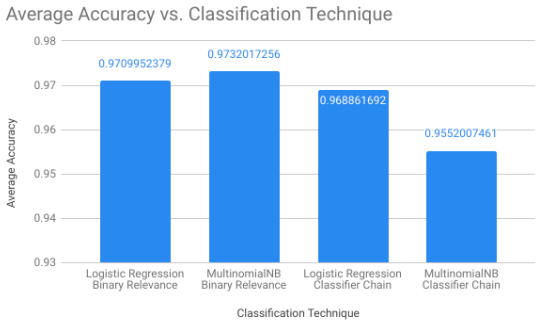

Figure 5: Accuracy of Baseline Model on Wikipedia Data

$$RankingLoss(h,\mathcal{U}) = \frac{1}{L}\sum_{i=1}^{L}\frac{1}{|Y_i||\overline{Y_i}|}|\mathrm{R}_i|$$
$$R_i = \{(y_1,y_2)|h(x_i,y_1) \leq h(x,y_2),(y_1,y_2) \in Y_i \times \overline{Y_i}\}.$$
$\overline{Y_i}$ denotes the complementary set of Y_i in Y.

Figure 6: Ranking Loss Function

$$SubsetLoss(h,\mathcal{U}) = \frac{1}{n}\sum_{i=1}^{n} I(h(x_i) \neq y_i)$$
$I(\cdot)$ denotes the indicator function, i.e. $I(\pi) = 1$ iff π holds, otherwise $I(\pi) = 0$.

Figure 7: Subset Loss Function

$$Micro - F1(h,\mathcal{U}) = \frac{2\times\sum_{i=1}^{n}\|h(x_i)\bigcap y_i\|_1}{\sum_{i=1}^{n}\|h(x_i)\|_1+\sum_{i=1}^{n}\|y_i\|_1}$$

Figure 8: Micro F1 Function

$$OE(h,\mathcal{U}) = \frac{1}{L}\sum_{i=1}^{L}\|argmax_{y\in Y}h(x_i,y) \notin Y_i\|$$
$\|\pi\|$ equals 1 if π holds and 0 otherwise.

Figure 9: One Error Function

with Word2Vec, FastText and GloVe word embedding.

4.1.1 FastText

FastText is an extension to Word2Vec.It generates the n-grams of the words. The word embedding vector will be the sum of all these n-grams. Rare words can now be properly represented as some of their n-grams may appear in other words with a higher chance (van Aken et al., 2018).
Fast text is an open source library which provides many pre-trained embedding models like English word vectors and Multi-lingual word vectors. These model can be used by reducing size on different devices. It is also a word or text embedding technique, but it is different from glove and word2vec. The word2vec and glove both consider a word is the smallest unit whose vector representation is to be found, but fast text breaks every word into n-grams character. This approach has many benefits such as it helps to find vector representation for the rare words. It gives the vector representation for the words that are not present in the dictionary, unlike the glove and word2vec.

4.2 Glove

The glove is used for obtaining the vector representations for the given words. Glove model is the combination of the features of global matrix factorization and local context window methods. The glove model is made from the word-word co-occurrence statistics from a corpus and this results in forming an embedding of a word to a vector representation. The process of using matrix factorization methods to perform rank reduction on a big term-frequency matrix is called Global matrix factorization and the family of word embedding model learns semantics by passing over the corpus is local context window. It is a text2vec implementation which has many pre-trained embedding models of different corpus available online. The twitter dataset pre-trained embedding is also available which is used in the project.
Word2vec model of word embedding is based on the two models i.e. bag of words model and the skip-gram model. The two models are used to predict the word in the center position and around it respectively using the neural network approach whereas the Glove model forces the vectors to encode the frequency distribution of which words occur near them.

4.3 Word2Vec

word2vec is a word embedding technique which is used to generate the vector representation of the words that are present in the dictionary. There are many pre-trained embedding models present such as gensim model of word2vec.

5 Long short-term memory (LSTM)

Long short-term memory (LSTM) is a feed forward recurrent neural network, (RNN) based model which has the ability to remember selective patterns for long durations of time. In conventional neural network (CNN) the information does not pass from one level to another, so that is the drawback of the model. On the other side, RNN gains knowledge of the prior level and output in every step of the given number of iteration. But the learning or information gain is only from the previous state and not from the series of level which is called as a long-term dependency. The RNN approach is volatile approach as it is sensitive to error gradient. So the drawback of RNN lead to the model called Long Short Term Memory which can solve this problem as it uses gates to control the process of memorization. The Lstm has input gates to take information as input, output gate to output information on every iteration and Forget gate, to forget or drop information that is not required.

$$HammingLoss(h,\mathcal{U}) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{L}\|h(x_i) \bigoplus y_i\|_1$$

$\bigoplus$ stands for the symmetric difference of two sets (XOR operation), and $\|.\|_1$ denotes the l_1-norm.

Figure 10: Hamming Loss Function

$$Macro - F1(h,\mathcal{U}) = \frac{1}{L}\sum_{k=1}^{L}\frac{2\times\sum_{i=1}^{n}h^k(x_i)y_i^k}{\sum_{i=1}^{n}h^k(x_i)+\sum_{i=1}^{n}y_i^k}$$

y_i^k is the k-th entry of y_i and $h^k(x_i)$ is the k-th entry of $h(x_i)$

Figure 11: Macro F1 Function

6 Evaluation Metrices

Evaluation metrices helps to evaluate the performance of our model. In multiclass classification metrices like precision, sensitivity, specificity, F1 score, AUC etc. works good to evaluate the model but in case of multilabel classification these metrices may not work well (Shi et al., 2014).
Following are the metrices which would work well on multilabel Classification.

6.1 Hamming Loss

The rate of average error over all the binary labels given by the classifier is Hamming Loss. The range of this evaluation metric is between 0 and 1. Less the value of Hamming loss more the performance of Classifier Figure 10.

6.2 Micro F1

It consider both precision and Recall micro-average on all binary label with equal weightage. The Classifier which provide more Micro F1 value will have good performance. The value of Micro precision is between 0 to 1 Figure 8.

6.3 Macro F1

It consider both precision and Recall macro-average on all binary label with equal weightage. The classifier which provide more macro F1 value will have good performance. The value of Micro precision is between 0 to 1 Figure 11.

6.4 Subset 0/1 Loss

It is loss metric having value between 0 to 1. It gives the prediction of percentage of classifier's label set when it is exactly correct. Less value of subset 0/1 Loss would be good for the classifier Figure 7.

$$Accuracy(h,\mathcal{U}) = \frac{1}{n}\sum_{i=1}^{n}\frac{|h(x_i)\bigcap y_i|}{|h(x_i)\bigcup y_i|}$$

Figure 12: Accuracy Function

$$coverage(h,\mathcal{U}) = \frac{1}{L}\sum_{i=1}^{L} max_{y\in Y_i} rank^h(x_i,y) - 1$$

Figure 13: Coverage Function

Glove + LSTM	Precision	Recall	F1 Score	Support
Hate	0	0	0	290
Offensive	0.9	0.94	0.92	3832
Neither	0.67	0.76	0.71	835
Macro Average	0.52	0.57	0.54	4957
Micro Average	0.86	0.86	0.86	4957

Figure 14: Twitter Data Results using Glove + LSTM

6.5 Accuracy

It evaluates the fraction of correct labels across all the example set.The more the value of accuracy, the more classifier is of good performance. It's value lies between 0 to 1 Figure 12.

6.6 Ranking Loss

The average number of disordered label pair for an example in the sample. Less the value of Ranking Loss, more good the performance of classifier Figure 6.

6.7 One Error

In true label set,how many top ranked label are not there of an example. Less the one Error more good the classifier is. It's value is between 0 to 1 Figure 9.

6.8 Coverage

It Evaluate on average, how many steps are needed to move down the label list in order to cover all the true labels of an example Figure 13.

7 Result

For Twitter dataset, the data is split in 80:20 train-test set ,the tf-idf + FK and FRE based features trained with Logistic Regression with L2 Regularization gives best results. The results are shown in Table 3.For other models like Fast Text and Glove with LSTM classifier the results are shown in Figure 15 and 14 respectively.

For Wikipedia Dataset Fast Text model and different ensemble base techniques like classifier

FastText	Precision	Recall	F1 Score	Support
Hate	0.5	0.28	0.35	290
Offensive	0.93	0.95	0.94	3832
Neither	0.83	0.87	0.85	835
Macro Average	0.75	0.7	0.71	4957
Micro Average	0.9	0.9	0.9	4957

Figure 15: Twitter Data Results using FastText

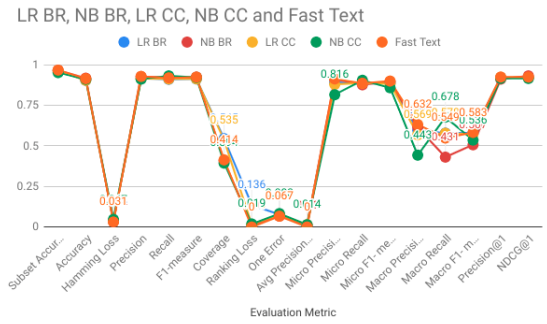

Figure 16: Wikipedia Data Evaluation Results Graph

Evaluation Metric	LR BR	NB BR	LR CC	NB CC	Fast Text
Subset Accuracy	0.964	0.967	0.962	0.952	**0.968**
Accuracy	0.905	**0.918**	0.905	0.911	0.915
Hamming Loss	0.035	0.032	0.037	0.047	**0.031**
Precision	0.916	0.928	0.915	0.912	**0.9285**
Recall	0.911	0.922	0.913	**0.932**	0.92
F1-measure	0.913	0.925	0.914	0.922	**0.924**
Coverage	**0.545**	0.408	0.535	0.394	0.414
Ranking Loss	0.136	0.008	0.015	0.019	**0**
One Error	0.077	**0.067**	0.078	0.083	**0.067**
Avg Precision Score	0.013	0.01	0.014	0.014	**0**
Micro Precision	0.893	0.911	0.88	0.816	**0.913**
Micro Recall	0.884	0.877	0.887	**0.905**	0.887
Micro F1- measure	0.888	0.897	0.884	0.859	**0.9**
Macro Precision	0.597	0.615	0.569	0.443	**0.632**
Macro Recall	0.579	0.431	0.578	**0.678**	0.549
Macro F1- measure	0.581	0.507	0.574	0.536	**0.583**
Precision@1	0.922	0.912	0.921	0.916	**0.925**
NDCG@1	0.922	**0.932**	0.921	0.916	0.925

Figure 17: Wikipedia Data Evaluation Results Table

chain, Binary Relevance were applied. The comparison of results on different evaluation metrics of multilabel classification are shown in Table 17 and in Graph .16

8 Conclusion

In Wikipedia dataset Fast Text model performs better than different ensemble base techniques like classifier chain, Binary Relevance. Though the accuracy of all the models are nearly same but for other multilabel metrics like hamming loss, F1 measure, Micro precision, one error FastText performs better than other ensemble based techniques. For Twitter Dataset Tf-Idf based + FK+FRE based features trained with Logistic Regression with L2 Normalization performs better than fast Text and Glove + LSTM. Though the accuracy of all the applied models are same but best model outperform other in terms of recall.

References

Betty van Aken, Julian Risch, Ralf Krestel, and Alexander Löser. 2018. Challenges for toxic comment classification: An in-depth error analysis. *arXiv preprint arXiv:1809.07572*.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. In *Eleventh International AAAI Conference on Web and Social Media*.

Chuan Shi, Xiangnan Kong, Di Fu, Philip S Yu, and Bin Wu. 2014. Multi-label classification based on multi-objective optimization. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 5(2):35.