TOXIC COMMENTS CLASSIFICATION

PROJECT REPORT

Aarav Varshney and Argha Chakrabarty

Ashoka University

December 2021

Contents

1 Introduction **1**

2 Literature Survey **1**

2.1 Toxic Comment Classification by Zaher et al. 1

2.2 A ML Approach to 1

3 Project Description and goals **2**

4 Dataset Specification **3**

5 Implementation details **3**

6 Observations **4**

6.1 Binary Classifier SVM 4

6.2 Multilabel Classifier SVM 5

6.3 Logistic Regression 5

6.4 Naive Bayes 5

6.5 Decision Trees 5

7 Conclusions and future directions **5**

1 Introduction

The project presents an application of NLP to classify texts posted on online platforms as comments as either toxic or non-toxic. While social media is a great medium for sharing opinions, it is more and more being used to bully and harass people. This project aims to classify such comments and hopefully make social media a bit more welcoming. We study the impact of tf-idf and SVMs, Decision Trees, Naive Bayes and CNNs. We evaluated our approaches on comments from the Kaggle Toxic Comments Classification Challenge. Additionally we host the classifier so that users can interact with it.

2 Literature Survey

This are several papers already published on the topic of Toxic Comments Classification. This was helpful for us to understand the topic and the way experts have approached it. We have used the following papers for our analysis:

2.1 Toxic Comment Classification by Zaher et al.

The authors used Naive Bayes (as benchmark), RNN and LSTM to classify toxic comments. They trained the classifier on the same kaggle dataset.

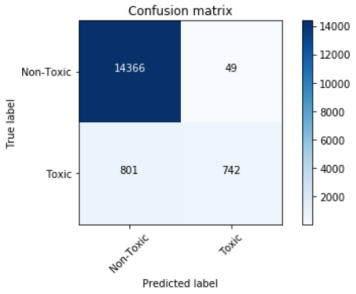


Fig. 5. Evaluation of Naïve Bayes algorithm using confusion metrics.

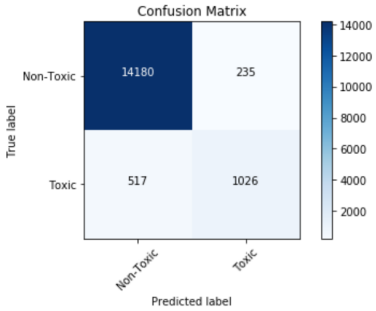
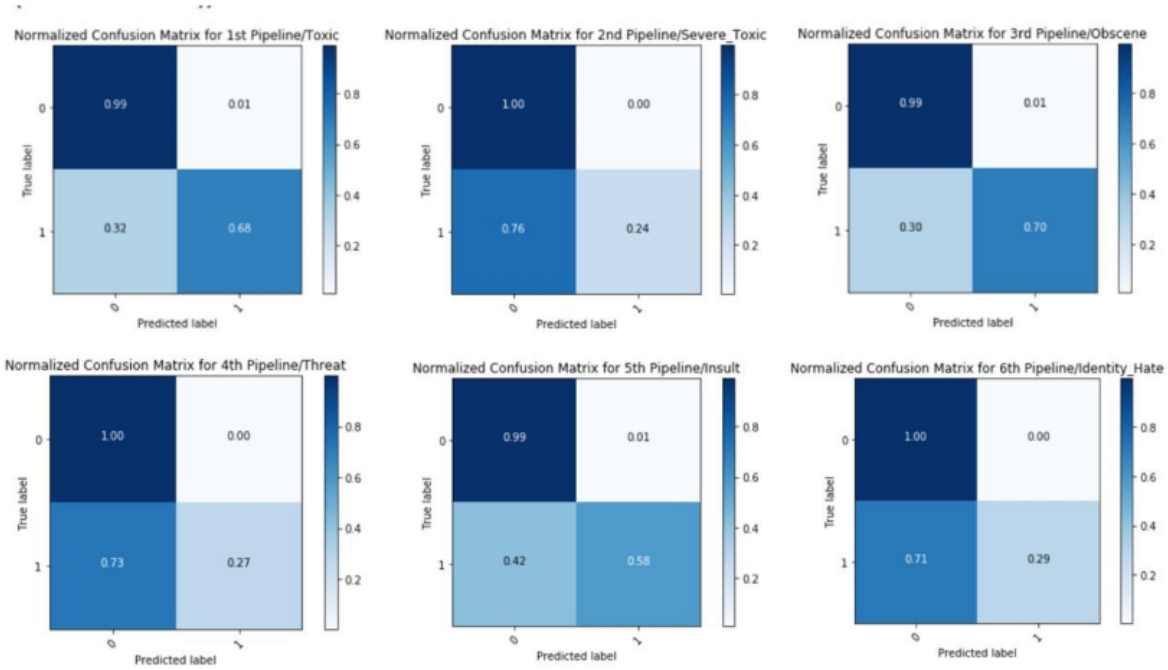


Fig. 6. Evaluation of LSTM algorithm using confusion metrics.

The results showed that LSTM has an almost 20% increase in True Positive Rate (TRP) which means that among all the identified toxic comments LSTM has 20% more sensitivity in correct assignment of the comments. Furthemore there was an increase in Recall and F1 score when using LSTM.

2.2 A ML Approach to

The authors solved the problem of multi-label classification rather than binary and further classified toxic comments as hate, toxic, severelytoxic and so on. Two approaches were used for 3 labels each out of 6 labels. The authors used Bag of words using Word count vectorizer and then the tf-idf transformer which is finally used to train. Support Vector Machine (SVM) classifier with 100 as maximum iterations and C as 1.0. Whereas for the other three labels, decision tree classifier was used instead.



The paper claims that this apporach performs better than LSTM and Naive Bayes and the results validates this claim. We want to validate this claim on binary classification.

3 Project Description and goals

The project is to classify toxic comments on social media platforms. The goals are as follows:

Use Different Classification Techniques

1. **Naive Bayes**: Use the results as a benchmark to compare other results with. This was chosen due to its reputation as a text classifier. This is a Generative Learning Algorithm that uses discrete values rather than continuous. Here we model p(x|y) that is the probability a particular word occurs given the comment is toxic or not. We make a strong assumption that xi's are conditionally independent given y. Finally after fitting the model, we calculate p(y = 1|x) which is just p(x|y = 1)p(y = 1)/ p(x)

2. **Logistic Regression**: We wanted to try regression and for discrete values, we used logistic regression.

3. **Decision Trees**: Here at each level we place the best feature at the node and then the split happens depending on the features that produce the minimum classification error. And we chose the output based on the class of majority of data in the decision node.

4. **Support Vector Machines**: Support vector machine models are unique in that they find the boundaries between classes by looking at the distances between the separating line and the nearest point for each class, and are resistant to outliers.

Deploy The Model

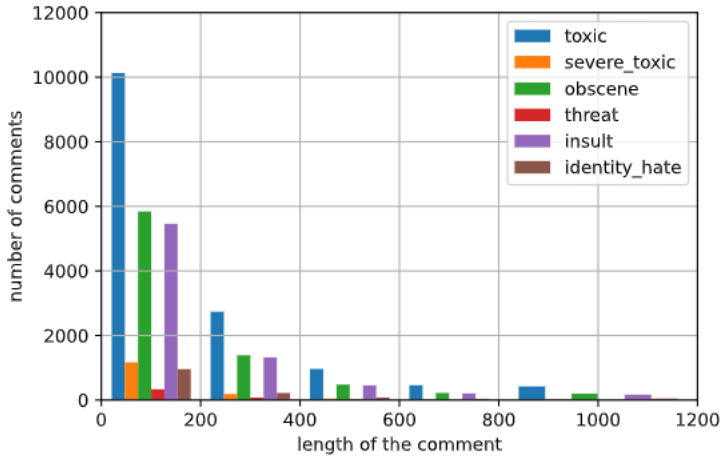
Finally, we want to create REST API for the classifer that can be deployed on any platform. We used Flask for this.

	id	comment_text	toxic	severe_toxic	obscene	threat	insult	identity_hate
0	0000997932d777bf	Explanation\nWhy the edits made under my usern...	0	0	0	0	0	0
1	000103f0d9cfb60f	D'aww! He matches this background colour I'm s...	0	0	0	0	0	0
2	000113f07ec002fd	Hey man, I'm really not trying to edit war. It...	0	0	0	0	0	0
3	0001b41b1c6bb37e	"\nMore\nI can't make any real suggestions on ...	0	0	0	0	0	0
4	0001d958c54c6e35	You, sir, are my hero. Any chance you remember...	0	0	0	0	0	0

Figure 1: A snapshot of the dataset

4 Dataset Specification

The dataset was acqured from Wikipedia comments which was then labelled by Jigsaw and then finally provided by Kaggle. The dataset had 1,59,571 comments, where each comment was labelled as toxic, severe toxic, obscene, threat, insult, identity hate or none. The data was labelled by the teams at Jigsaw (Alphabet) where each comment was labelled by at least 7 humans to reduce bias. The length of each comment varied from 10 words to 1200 words. We ran an analysis to chose comments with word count less than 600 as we saw that after this length, the number of toxic comments were too few. So, a total of 1,32,327 comments were chosen.



This is a multilabel classifier porblem which we converted to a binary classifier problem by relabelleing any comments identifying as any of the seven categories mentioned earlier. But the results for both have been published.
The dataset also went through stemming and lemmatizeg to remove any or all stopwords. Furthermore, we cleaned punctuations, numbers also ip-addresses as well attempted to remove usernames so that particular usernames do not make the model biased.

5 Implementation details

Data Preprocessing

After the data is read, we select only comments with less than 600 words in it, and label each comment as toxic or not depending on the pre-existing labels. nltk's wordnet module is used for stemming and lemmetizeing each comment after removing all the punctuations and numbers along with any other redundant utf-8 characters.

Vectorization

After all the comments are cleaned, we use TF-IDF to vectorize our comments so that we can use them for training our model. TfidfVectorizer is used as is from the scikit-learn module. TF-IDF stands for Term frequency-inverse document frequency, is a numerical statistic that is intended to reflect how important a word is to a document in a collection. A document in this case is the comment whether toxic or not. It does this by comparing the frequency of usage inside an individual comment as opposed to the entire data set. The importance increases proportionally to the number of times a word appears in the individual coment which is called Term Frequency. But if multiple comments contain the same word repeatedly (example stopwords) then their importance decreases.

TF(t) = Number of times term t appears in a document / Total number of terms in the document
IDF(t) = log_e(Total number of documents / Number of documents with term t in it).
Value = TF * IDF

This vector becomes the feature vector for each comment.

Training and Testing

The test to train split ratio is 3:1. Then we trained our model on SVM, Logistic Regression, Naive Bayes and Decision Trees. We trained the model on a laptop with i5-8520U and 16 gigs of RAM. We also trained a multi-label model using Binary Relevence from scikit-multilearn module which is built on top of scikit-learn eco-system and provides multi label classification.

Deployment

After training, we decided to deploy our model for demo purposes. Since we used python for our project, we decided to go with a simple FastAPI server. We created a pipeline with two stages, the first one vectorizing any input text and at the stage the actual fitting or prediction occurs. To save the model we pickle the pipeline object using python's in-built pickle library and write the raw binary data onto a file. Afterwards, the file is un-pickled at the server-side and a single string array is given to it for prediction. The API takes a single string as it's payload and returns the predicted outcome as it's response. The user has to create a POST request with the comment as the payload and we return whether the comment is toxic or not in the response body. Lastly we deployed the whole thing in a heroku server. You can take a look at the API at this link :- https://iml-tox-det.herokuapp.com/docs (please keep in mind that we are using a free server, it might take some time to start). We also went on and created a frontend website that uses the api: https://adoring-bell-52f360.netlify.app

6 Observations

6.1 Binary Classifier SVM

Confusion Matrix:

```
array([[29258,   189],
       [ 1263,  2372]], dtype=int64)
```

Other metrics

recall	precision	f1 score
0.926	0.653	0.766
0.926	0.653	0.766

6.2 Multilabel Classifier SVM

6.3 Logistic Regression

Confusion Matrix:

```
array([[29304,   143],
       [ 1425,  2210]], dtype=int64)
```

Other metrics

recall	precision	f1 score
0.939	0.608	0.738
0.939	0.608	0.738

6.4 Naive Bayes

Confusion Matrix:

```
array([[29432,    15],
       [ 2646,   989]], dtype=int64)
```

Other metrics

recall	precision	f1 score
0.985	0.272	0.426
0.985	0.272	0.426

6.5 Decision Trees

Confusion Matrix:

```
array([[28591,   856],
       [ 1055,  2580]], dtype=int64)
```

Other metrics

recall	precision	f1 score
0.751	0.71	0.73
0.751	0.71	0.73

7 Conclusions and future directions

References