Toxic Statement Classification

Akshay Kumar Chaturvedi

16 May 2018

Contents

Chapter 1

Introduction

1.1 Problem Statement

With the advent of social media lots of people have found their voice, on one hand this has led to democratization and diversification of opinions on whole range of subjects (from politics to education to food), on the other hand it has also led to people abusing anyone who disagrees with them, at times even threatening others with physical harm.

But not all the statements are threatening; some are insulting and toxic but not threatening. Aim of this project is to build a multi-headed model (a statement can fall into multiple categories simultaneously) that is capable of detecting different types of toxicity in statements like insulting, obscene, threatening etc. This will hopefully help in making online discussions more productive and respectful.

1.2 Data

Data has been taken from Kaggle; dataset contains comments taken from Wikipedia talk page edits. Dataset marked as 'train' has been taken from Kaggle to complete this project, this data was divided into two parts, one to train the models and another to test the models. A sample of data is below:

Table 1.1: Toxic Statements Sample Data

id	comment_text	Toxic	severe_toxic	Obscene	threat	insult	identity_hate
0000997932d777bf	Explanation Why the edits made under my username Hardcore Metallica Fan were reverted? They weren't vandalisms, just closure on some GAs after I voted at New York Dolls FAC. And please don't remove the template from the talk page since I'm retired now.89.205.38.27	0	0	0	0	0	0
000103f0d9cfb60f	D'aww! He matches this background colour I'm seemingly stuck with. Thanks. (talk) 21:51, January 11, 2016 (UTC)	0	0	0	0	0	0

Dataset has 8 columns, first one is id of statements, second one is the statement itself, from third to eighth are the six categories of statements, they are ‘toxic’, ‘severe_toxic’, ‘obscene’, ‘threat’, ‘insult’ and ‘identity_hate’. Each of these categories can have values 0 or 1, 0 means the statement does not fall in that category and 1 means it does.

Dataset has 159571 rows or observations. Out of these 159571 statements, 143346 do not fall into any category. If we consider different combinations of labels as different classes we get 41 different classes in the data. Table below presents the distribution of these different classes. First row of the table below justifies the above statement that ‘Out of 159571 statements, 143346 do not fall into any category’.

Table 1.2: Different categories by combination of labels

	toxic	severe_toxic	obscene	threat	insult	identity_hate	Count
0	0	0	0	0	0	0	143346
1	0	0	0	0	0	1	54
2	0	0	0	0	1	0	301
3	0	0	0	0	1	1	28
4	0	0	0	1	0	0	22
5	0	0	0	1	1	0	3
6	0	0	1	0	0	0	317
7	0	0	1	0	0	1	3
8	0	0	1	0	1	0	181
9	0	0	1	0	1	1	18
10	0	0	1	1	0	0	2
11	0	0	1	1	1	0	2
12	1	0	0	0	0	0	5666
13	1	0	0	0	0	1	136
14	1	0	0	0	1	0	1215
15	1	0	0	0	1	1	134
16	1	0	0	1	0	0	113
17	1	0	0	1	0	1	7
18	1	0	0	1	1	0	16
19	1	0	0	1	1	1	3
20	1	0	1	0	0	0	1758
21	1	0	1	0	0	1	35
22	1	0	1	0	1	0	3800
23	1	0	1	0	1	1	618
24	1	0	1	1	0	0	11
25	1	0	1	1	1	0	131
26	1	0	1	1	1	1	56
27	1	1	0	0	0	0	41
28	1	1	0	0	0	1	3

29	1	1	0	0	1	0	14
30	1	1	0	0	1	1	7
31	1	1	0	1	0	0	11
32	1	1	0	1	0	1	1
33	1	1	0	1	1	0	1
34	1	1	1	0	0	0	158
35	1	1	1	0	0	1	6
36	1	1	1	0	1	0	989
37	1	1	1	0	1	1	265
38	1	1	1	1	0	0	4
39	1	1	1	1	1	0	64
40	1	1	1	1	1	1	31

Figure below presents the distribution of statements per label or category. Category 'toxic' dominates as 15294 statements have this label and 'threat' which is the most extreme form of label has lowest number of 478.

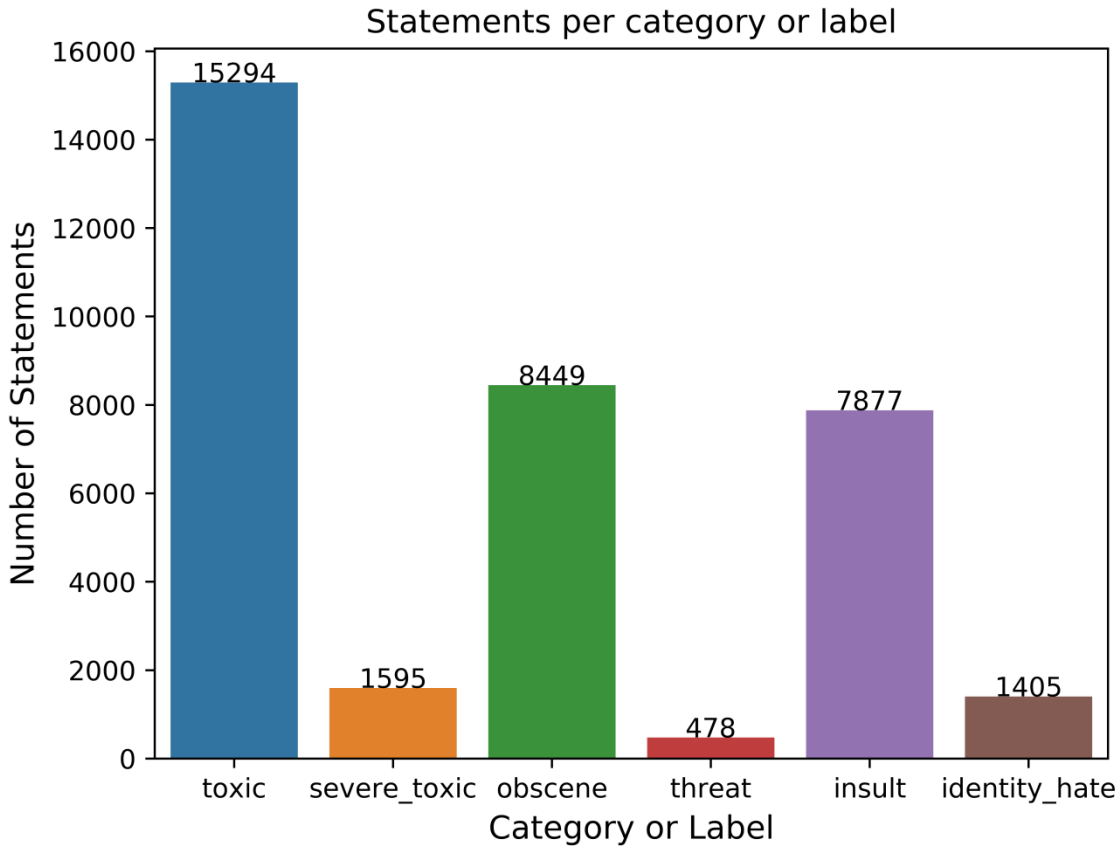

Figure 1.1: Statements per category or label

Figure 1.2: Word Cloud for 'identity_hate' category

Figure 1.3: Word Cloud for 'insult' category

Figure 1.4: Word Cloud for 'obscene' category

Figure 1.5: Word Cloud for 'severe_toxic' category

Figure 1.6: Word Cloud for 'threat' category

Figure 1.7: Word Cloud for 'toxic' category

From the word clouds for six categories above, we can observe that ‘fuck’ seems to be a common frequent word in all of them. But as we look a little deep in the visualizations we can see that there are words which distinguish categories from each other. For example, word cloud for ‘threat’ category has words ‘die’, ‘death’, ‘kill’ which don’t appear in other word clouds. Similarly if we look at ‘identity_hate’ word cloud, we can see words ‘nigger’, ‘Jew’, ‘Nazi’ which are clearly aimed at someone’s identity.

Chapter 2

Methodology

2.1 Pre-processing

In its raw form, statements cannot be used for predictive modeling. We need to clean the statements and generate features from the statements to construct a model. Below are the steps used for data pre-processing:

- Removing the leading and trailing white spaces.
- Converting all the letters to lower case.
- Removing numbers.
- Removing words which are essential components of grammar but have no semantic significance, also known as 'stop words'. For example, 'the', 'a' etc.
- Removing punctuation marks such as comma, full stop etc.
- Replacing words by their root or base form, also known as stemming. For example, 'cats' is replaced by 'cat'.
- Replacing short words like 'I'm' by 'i am', 'can't' by 'cannot'. There are other cases as well, please refer the code for more such cases.
- A peculiarity encountered was some statement contained words like 'sssssssssttttttttaaaaaaaaaaayyyyyyyyy'; this was replaced by 'stay'. If any character which appears more than two times consecutively then all those repeated characters are replaced by one character only.

After the statements were cleansed, a TF-IDF (term frequency-inverse document frequency) matrix is generated from them, the columns of this matrix serve as features for predictive modeling.

2.2 Modeling

All the results shown in the below experiments are of test data. For each experiment, data was split in 70:30 ratio, models were trained on 70 % of data and tested on remaining 30 %.

Baseline

Before running any experimental models, a baseline was established which would serve as the benchmark to compare all the experimental models. This baseline was taken from a Kaggle kernel by Jeremy Howard [1] (Jeremy's code was used to establish the baseline; code's link is given in the references and also in the python file itself).

Jeremy's pre-processing is pretty simple and interesting; he adds whitespaces before and after each special character found in the comments, so that when it splits the string to get a list of words, the special characters are listed on their own.

Jeremy created a strong baseline using a variant of [2]; it is called Naïve Bayes Logistic Regression, please go through the paper [2] for details on the Naïve Bayes Logistic Regression model. A parameter change was made in calculation of TF-IDF matrix regarding maximum number of features. Due to memory issues, I set the maximum number of features to 5000, what this did was that only top 5000 words were selected as features, rest were discarded. This model gave an overall accuracy of 59.49% approximately (overall accuracy means that all the labels of a statement have been predicted correctly, even if one label is misclassified then the statement will be treated as misclassified). Label wise accuracy and is given in the table below:

Table 2.1: Label Wise Results for Baseline

Category	Accuracy
Toxic	63.47 %
Severe Toxic	98.98 %
Obscene	94.68 %
Threat	99.7 %
Insult	93.41 %
Identity Hate	99.12 %

A slight tweaking of a Logistic Regression parameter 'C' (inverse of regularization strength) from 4 to 1 led to an overall accuracy of 68.16 %. Label wise accuracy is given in the table below:

Table 2.2: Label Wise Results for Improved Baseline(C=1)

Category	Accuracy
Toxic	72 %
Severe Toxic	98.98 %
Obscene	95.2 %
Threat	99.7 %
Insult	94.85 %
Identity Hate	99.12 %

Adding my pre-processing further improves the above baseline, it gives an overall accuracy of 72 %. Label wise accuracy is given in the table below:

Table 2.3: Label Wise Results for Final Baseline(C=1 and added my pre-processing)

Category	Accuracy
Toxic	75.76 %
Severe Toxic	98.98 %

Obscene	95.07 %
Threat	99.70 %
Insult	94.97 %
Identity Hate	99.12 %

Experimental Models

Multi-label classification is a challenging problem and a lot of debate has been going on since many years on how to tackle this problem, people have come up with different methods and different accuracy measures on this issue. Some experimental models that I have tried are presented below, these models are in no way exhaustive and there can be many more different models, the reason behind choosing these particular models is that they are easy to understand and yet perform very well as we will see below.

Only Machine Learning models do not work very well for multi-label classification problems, problem transformation techniques combined with Machine Learning models perform very well. These techniques are:

Binary Relevance: In this method, each label is treated as a separate single category classification problem; this is what was done in baseline formation. A model was created using Naïve Bayes Logistic Regression, then this model was trained separately on each label.

Classifier Chains: In classifier chains, the first classifier is trained on the input data and then each next classifier is trained on the input space and all the previous classifiers in the chain. The advantage of this method is that it takes the label dependencies into account for classification.

Label Powerset: This method transforms the multi-label classification problem into single class classification problem by treating each unique combination of labels as a different class. This distribution is shown is table 1.2.

With the above three problem transformation methods we have combined three machine learning models namely logistic regression, multinomial naïve bayes and decision trees. Below are the results for each of them:

Logistic Regression

After going through several experiments, it was found that Logistic Regression works best with classifier chains, 1 ngrams and maximum features of 2500, giving an overall accuracy of 91.95 %; this is nearly 20 points better than our baseline accuracy of 72 % which is great. Label wise accuracy for each label is given in the table below:

Table 2.4: Label Wise Results for Logistic Regression + Classifier Chains (2500 features)

Category	Accuracy
Toxic	95.66 %
Severe Toxic	99 %
Obscene	97.96 %
Threat	99.70 %
Insult	96.97 %
Identity Hate	99.21 %

Other experiments were also conducted with change in ngrams and maximum number of features but didn't give better results. In case of ngrams changing from 1 to 1 and 2 ngrams or 1 and 3 ngrams or 2 and 3 ngrams, accuracy was a bit low than above. Only model that was able to perform as good as the best model above was when maximum features were changed to 5000 and rest all features were kept same. The reason why it was not selected as the best model is that it did not provide any improvement in overall accuracy and took more time than above model (as it was working with double the features). Table below gives the label wise result for 1 ngrams and maximum features of 5000:

Table 2.5: Label Wise Results for Logistic Regression + Classifier Chains (5000 features)

Category	Accuracy
Toxic	95.72 %
Severe Toxic	99 %
Obscene	97.93 %
Threat	99.70 %
Insult	96.95 %
Identity Hate	99.21 %

Multinomial Naïve Bayes

After going through several experiments, it was found that Multinomial Naïve Bayes works best with binary relevance, 1 ngrams and maximum features of 1000, giving an overall accuracy of 91.2 %; this also beats our baseline accuracy by nearly 19 points. Label wise accuracy for each label is given in the table below:

Table 2.6: Label Wise Results for Multinomial Naïve Bayes + Binary Relevance (1000 features)

Category	Accuracy
Toxic	94.44 %
Severe Toxic	98.98 %
Obscene	97.40 %
Threat	99.70 %
Insult	96.73 %
Identity Hate	99.14 %

Like in Logistic Regression, changing ngrams has not worked with Multinomial Naïve Bayes as well. Increasing maximum features to 2500 resulting in slight improvement of overall accuracy 91.3 %, this model was not selected as the improvement is very marginal and also model takes more time as it works with more than double the features. Label wise accuracy for each label is given in the table below:

Table 2.7: Label Wise Results for Multinomial Naïve Bayes + Binary Relevance (2500 features)

Category	Accuracy
Toxic	94.90 %
Severe Toxic	98.96 %
Obscene	97.42 %
Threat	99.69 %
Insult	96.83 %
Identity Hate	99.16 %

Decision Trees

After going through several experiments, it was found that Decision Trees works best with label power set, 1 ngrams and maximum features of 1000, giving an overall accuracy of 90.84 %; this also beats our baseline accuracy by nearly 19 points. Label wise accuracy for each label is given in the table below:

Table 2.8: Label Wise Results for Decision Trees + Label Powerset (1000 features)

Category	Accuracy
Toxic	93.19 %
Severe Toxic	98.97 %
Obscene	97.10 %
Threat	99.70 %
Insult	96.07 %
Identity Hate	99.15 %

Decision Trees with label power set and 2500 features performed slightly better than the above model with overall accuracy of 90.85 % but like in previous cases it was not selected as the difference in overall accuracy was very marginal and also it took more time than the previous model as it worked with more than double the number of features. Label wise accuracy for each label is given in the table below:

Table 2.8: Label Wise Results for Decision Trees + Label Powerset (2500 features)

Category	Accuracy
Toxic	93.16 %
Severe Toxic	98.98 %
Obscene	97.11 %
Threat	99.70 %
Insult	96.12 %
Identity Hate	99.14 %

All the above results for decision trees are for max_depth 3. A problem encountered in decision tree which was not encountered in logistic regression and multinomial naïve bayes is overfitting. With default parameters decision tree model was overfitting, so to counter this problem max_depth was set to 3 after trying several other values.

Chapter 3

Conclusion

In conclusion, out of all the models shown here and others not shown here but tried logistic regression with classifier chains, 1 ngrams and maximum features of 2500 emerges as the best model, giving an overall accuracy of 91.95 %. It not only outperforms other models but also is a very simple model rather than being complex and it also doesn't suffer from disadvantages like overfitting.

Except 'toxic' label, other labels seem to be pretty easy to predict, it may also be the case that as 'toxic' label statements are most out of all other labels that's why it's a bit difficult to predict, plus it also has more commonality with all other labels than other labels having with each other. This needs deeper analysis and would be interesting to look further into.

Pre-processing and regularization does make significant difference as can be seen in baseline improvements. Jeremy's baseline gave an overall accuracy of 59.49 %; just increasing the regularization strength improved the accuracy to 68.16 %. When my pre-processing was used with better regularization, overall accuracy of baseline was 72 % which is quite impressive, as we have just performed pre-processing and increased regularization without use of any complex models.

I have tried simple and limited number of models, but there are many complex models out in the world like deep learning models, this could also be done in future that improve the accuracy above 92 % using deep learning models.

Here we have only used TF-IDF matrix for features, other features should also be tried, but the results again prove how powerful TF-IDF matrix is as it alone has resulted in more than 90 % accuracy on different models.

Not much can be inferred about different problem transformation methods as all of them have performed well with different Machine Learning models.

Overall accuracy measure that we have used is considered a bit harsh as it deals in extremes (a statement is properly classified only if all the labels are correct). There are other measures which look at number of labels properly classified that's why we have given a table showing label wise accuracy for each model so that our performance has some detail and are not entirely opaque.

Lots of great insights we have got through this project and lots of interesting questions to work further upon.

References

1. https://www.kaggle.com/jhoward/nb-svm-strong-linear-baseline

2. Baselines and bigrams: Simple, good sentiment and topic classification. In Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2, 2012.