Information Retrieval and Text Analytics

- Toxic Comment Classification Challenge -

Amarvir Singh
s2131196
a.singh.9@umail.leidenuniv.nl

Georgios Kyziridis
s2077981
g.kyziridis@umail.leidenuniv.nl

Ioannis Chios
s2149133
i.chios@umail.leidenuniv.nl

October 14, 2018

1 Introduction

As we all understand and experience, the occurrence of bad comments, which are often categorized as "hate", "toxic", "insulting" or "threat", is an increasing problem on social media platforms and portals like Yahoo! Answers, Quora and Wikipedia. At first, anyone would think that we should convince people against posting this kind of stuff but then, since we cannot tell every single one of the internet users to not post bad views/speech, it's better if we can devise and implement a system that prevents users from posting bad comments.

Now the problem here is that users always find a workaround to this kind of systems, for instance using numbers like 3 instead of E, 4 instead of A and so on, but still having a system at the first place is what could solve a major of the problem. This is the motivation we took while doing this Kaggle Challenge on Toxic Comment Classification where we had majority of unlabeled data and we wanted to build a model to classify these unlabeled comments into different categories. Who knows one day, there might exist an actual system which we believe is not so far to be seen implemented.

2 Data Exploration

2.1 Toxic-Comments Dataset

This specific dataset consists of a large number of Wikipedia comments which have been labeled by human raters about each comment's toxicity. Train-set and test-set include 159571 and 153164 comments respectively. The purpose is to build a model that classifies the comments of the test-set into six different categories regarding the types of toxicity. More precisely, the basic goal of this assignment is to train a Natural Language Processing model which predicts the probabilities of each type of toxicity for each comment. The following table provides information about the label-frequencies in the train-set. It can be easily observed that the most multitudinous class is "Toxic" which is present in 15294 comments. This is reasonable because toxic is a very general label compared to the others.

Obscene	Insult	Toxic	Severe toxic	Identity hate	Threat
8449	7877	15294	1595	1405	478

Table 1: *Train set Class-Frequencies*

The graph below visualizes the correlation between the different classes. If two classes are strongly correlated (> 0.5), then the color of the corresponding square is more yellow, on the contrary if two classes are not correlated then the color is blue.

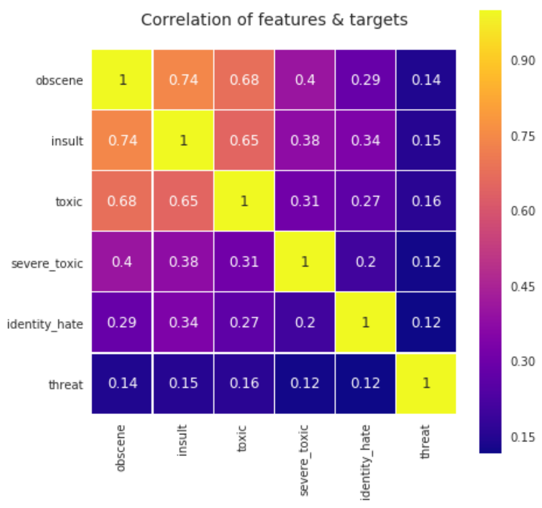

Figure 1: *Correlation of training data*

2.2 Imbalanced Data

The peculiarity of the dataset is that there are many comments which are not labeled. More precisely, 89.83% of the whole dataset consists of comments which are not included in any class. This fact would probably have a strong impact on our classification model. There are many solutions for the problem of imbalanced data such as "Undersampling/Oversampling" which are common methods to bring some artificial balance in the data. Unfortunately in this case this is very difficult due to the linguistic nature of the data. Additionally, we could not throw out the unlabeled comments because we would lose significant information. The plots below, visualize the frequencies of each class in the train-set.

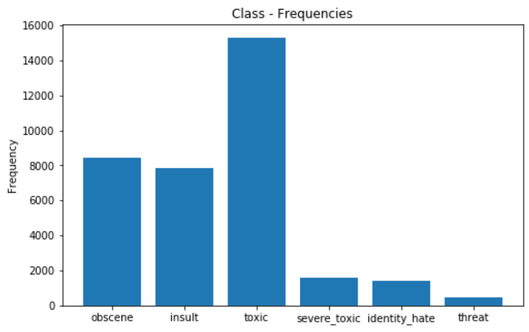

Figure 2: *Frequencies of each class*

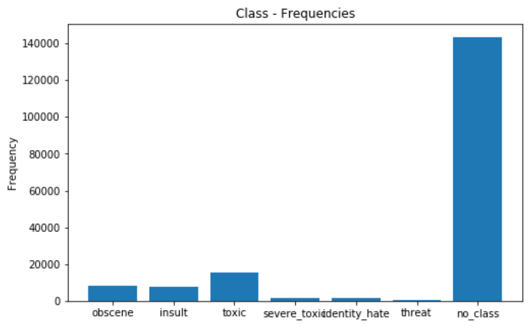

Figure 3: *Frequencies of imbalanced data*

3 Data Manipulation

3.1 Scikit-learn

The basic library we used in order to implement the algorithms described in this paper is Scikit-learn. The Scikit-learn project is a general purpose machine learning library written in Python. It provides efficient implementations of many well known machine learning algorithms, and a very easy to use API that makes it accessible even to non-machine learning experts. While scikit-learn focuses on ease of use and is mostly written in high level languages, care has been taken to maximize computational efficiency.

In our problem, in order to make our algorithm as efficient as possible we made use of Scikit-learn in various cases. To begin with, we used the TfidfVectorizer for the preprocessing and the feature extraction, as it is the fastest alternative that we are aware of, for the implementation of the TF-IDF algorithm. Furthermore, we utilized the LogisticRegression and the MultinomialNB classes to build each one of our classifiers respectively. Finally, we used various methods from sklearn.metrics (roc curve, roc auc score,etc) to measure the Mean Area Under the ROC Curve.

3.2 Preprocessing

In this subsection we will describe how we processed our data in order to be able to proceed with the classification. As we mentioned above both our training and our test set consist of approximately 160.000 comments each. These comments although are not as "noisy" as other text datasets retrieved from online platforms (Twitter etc.) still require some basic preprocessing.

Therefore, we performed some of the most common techniques in Natural Language Processing. To begin with, we tokenized each review and removed the stopwords. The aforementioned step could have been avoided, because some stopwords may occur in "toxic" phrases but we decided to remove them in order to reduce the size of the corpus and be able to process our data faster. Furthermore, we removed all numbers and punctuation from the comments since we do not believe that they are going to improve our classifiers.

One of the most popular and useful Natural Language Processing methods, that we deliberately omitted, is stemming. Hence, our main objective was to detect "toxic" words or phrases in order to classify them in different types of "toxicity" we decided not to use it in our problem. The reason behind this approach is that all stemmers that we are aware of are not able to handle curse words correctly, not to mention that many comments contain a lot of slang expressions. As a result many words that could be valuable for our classifier would have been reduced to the wrong stems and that would negatively affect our models.

3.3 Feature Extraction

To extract useful features from the corpus and build a feature vector for each comment we implemented a "Bag of N-grams" model with TF-IDF weighting. In more details, for each comment we have a vector with a length of 5000 features which denotes the TF-IDF weighting for each of the 5000 most frequent unigrams and bigrams in our corpus for the specific comment. We decided to use both bigrams and unigrams for our model instead of the common "Bag of Words" model because bigrams provide better results for the classification between different types of toxicity.

The TF-IDF weighting of the features is still a state of the art approach that almost always produces better results than the normal term frequency. This weighting technique assigns less weight to n-grams that occur more frequently across comments because they are generally not so good distinguishers between any pair of comments, especially if they are both "toxic", and a high weight to more rare n-grams.

$$tf - idf_{t,d} \;=\; tf_{t,d} \times idf_t \;=\; tf_{t,d} \times log(\frac{N}{df_t})$$

From the above formula we can observe that the way to achieve this weighting is the Inverse Document Frequency. Document Frequency (df_t) of t is the number of documents that contain term t. Additionally, df_t/N may be seen as the likelihood of t's occurrence in all documents. Therefore, we define idf as the logarithm of the aforementioned likelihood and $tf - idf$ of a term as the product of its tf weight and its idf weight.

4 Multi-label Classification

There are particular types of problems, where certain set of target variables are cast-off to measure the data. They are known as multi-label classification problems. A label may be a property to one or even more categories which are determined upon what the dataset is about and what labels are we considering to cluster the data into.

There are three methods to unravel a multi-label classification problem which are:

- Problem Transformation
- Adapted Algorithm
- Ensemble approaches

The interesting part here is finding ways to transform our problem domain. What we try to achieve is devising ways for transformation of the multi-label problem(s) into a single-label problem(s). There are distinct methods of achieving this. Since we used Problem Transformation for our dataset, a brief description about the types/methods is given below.

The distinct types of Problem Transformation are:

- Binary Relevance
- Classifier Chains
- Label Powerset

4.1 Binary Relevance

A very straightforward technique, it primarily treats each label as a discrete single class classification problem. This is what we used in problem transformation of the domain because of it's easy implementation and decent efficiency/accuracy. For example:

X	Y1	Y2	Y3
X1	1	0	0
X2	0	1	0
X3	0	0	1

Table 2: *Sample Problem*

By using Binary Relevance, the above table is broken down into distinct single class problem.

X	Y1
X1	1
X2	0
X3	0

Table 3: *X and subset Y1*

X	Y1
X1	0
X2	1
X3	0

Table 4: *X and subset Y2*

and so on.

To measure the accuracy of our predictions, we implemented the accuracy score metric. This function determines the subset accuracy. This implies that the estimated set of labels must exactly match with the convincing set of labels.

4.2 Classifier Chains

The methodology in this type functions in a way such that the first classifier is trained just on the input data and then the following classifiers are trained on the input space and all the previous classifiers forming a chain. This is known as Classifier Chains.

X	Y1	Y2	Y3	Y4
X1	1	0	1	0
X2	0	1	0	1
X3	0	0	1	0

Table 5: Sample Problem

First step - Compute one independent operation.

X	Y1
X1	0
X2	1
X3	0

Table 6: *X and subset Y1*

The next step then forms a chain with the previous step including the input space.

X	Y1	Y2
X1	1	0
X2	0	1
X3	0	0

Table 7: Forming chain with Y1 while computing Y2

The similarity to binary relevance is quite high. However, the only difference that exists is that it forms chains in order to encourage label correlation. But if the dataset we are acting upon has no label co-relations, it results in bad accuracy, even less than Binary Relevance method.

4.3 Label Powerset

The problem in this method is transformed into a multi-class problem using only one multi-class classifier, thereby training on every unique possible combination that is observed in the training data.

- X	Y1	Y2	Y3	Y4
X1	1	0	0	1
X2	0	1	0	0
X3	0	0	1	1
X4	1	0	0	1
X5	0	0	1	1
X6	1	0	1	1

Table 8: Sample Problem

Since X1 and X4 have same set of variables, they can be coupled together as a single class. The same implies for for X3 and X5.

X	Y1
X1	1
X2	2
X3	3
X4	1
X5	3
X6	1

Table 9: Classifying as per classes

Label powerset provides a unique class to every possible label combination that exits in the training set. This results in the highest accuracy amongst all the three methods discussed. The only disadvantage this method has is that the number of classes are directly proportional to the training data, hence it increases if the training data increases resulting in realtively huge increment in the number of classes. It increases the complexity of the model, and results in a lower accuracy.

5 Experimentation & Results

5.1 Logistic Regression

In this section, the process of modeling will be addressed. As it has been mentioned previously, the goal was to build a model in order to predict the probabilities of each class for each comment. The first model we built was logistic regression. In this approach, each class was considered as independent and consequently we trained a different model for each one of the classes. More precisely, the train-set was split into 70% and 30% train and validation set respectively. The former used for training the model for each class and the latter used for model evaluation. So this approach is based on multiple logistic regression models which predict if a comment belongs to a specific class or not for each one of the six different classes. In conclusion we consider the multi-label problem as six binary independent problems which are solved using independent logistic regression models. The following formula describes logistic regression using probabilities (odds).

$$logit(p) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_n X_n \tag{1}$$

$$odds = \frac{p}{1-p} = \frac{probability \quad of \quad presence}{probability \quad of \quad absence} \tag{2}$$

$$\textit{From equation (1),(2) : } logit(p) = \log(\frac{p}{1-p}) \tag{3}$$

From equation (1) it can be easily observed that logistic regression is a basic linear model which describes the relation between the target characteristic (class: Y) and a set of independent variables (features: X_i). Logistic regression generates the coefficients β_i and the standard errors (residuals) as well. So all in all, logistic regression model is a formula which predicts a logit transformation of the probability of presence of the target characteristic. Furthermore, in classic ordinary linear regression models, the parameters are chosen regarding the minimization of sum of squared errors. In this case of logistic regression, the parameters are chosen with the intention of maximize the likelihood of observing the sample values.

5.2 Multinomial Naive Bayes

One of the models we used for our prediction was the Multinomial Naive Bayes classifier, which is one of the most commonly used classifiers for text classification. Multinomial Naive Bayes is a probabilistic method where the probability of a document d being in class c is computed as

$$P(c|d) = P(c) \prod_{1 \leq k \leq n_d} P(t_k|c)$$

Furthermore, $P(t|c)$ is the conditional probability of a term t occurring in a document of class c and $P(c)$ is the prior probability of a document occurring in class c. The aforementioned probabilities normally provide sufficient evidence to classify a document in the correct class.

However, as we mentioned above, in our problem a document can belong to multiple classes. Therefore, we do not need to find the maximum likelihood given from the Multinomial Naive Bayes in order to classify each comment to a particular class.We just need to predict the probability for each of the six possible types of comment toxicity.This is easily achieved by using the Multinomial Naive Bayes combined with the binary relevance in order to estimate the likelihood for each class independently.

5.3 Results

As mentioned above, the final goal of this project is to classify correctly each comment to each class according to the type of toxicity and produce the probabilities that predict the different classes on the test-set. In order to evaluate the quality of the classifier/model we measure out the following metrics: training accuracy, test accuracy and area under roc curve. Training accuracy is a percentage of true positive predictions using the fitted model for predicting the data from the train-set. The percentage is produced by calculating the number of correct predictions for each class according to the ground-truth. Test or validation accuracy, is again a percentage but in this case measures the number of correct predictions according to the validation-test. So first the model was trained and fitted into the features of each comment as the independent variables X_i and the corresponding class as the dependent variable Y. This process, as it had been mentioned previously, took place six times, one for each class. The following tables provide information about training and validation accuracy for each class in two different models.

-	Training Accuracy	Validation Accuracy
obscene	0.983	0.978
Insult	0.976	0.970
Toxic	0.964	0.955
Severe Toxic	0.992	0.989
Identity Hate	0.994	0.991
Threat	0.998	0.997

Table 10: *LogReg Accuracy for each class*

-	Training Accuracy	Validation Accuracy
obscene	0.973	0.972
Insult	0.969	0.968
Toxic	0.951	0.947
Severe Toxic	0.990	0.989
Identity Hate	0.991	0.991
Threat	0.996	0.996

Table 11: *NB Accuracy for each class*

As it can be easily observed, validation accuracy is very close to the accuracy of the training set. This means that, the models provide robust predictions and they can be generalized into various datasets of the same kind.

Except from accuracy, there is another measure for evaluation called MAURC which means "Mean Area Under Roc Curve". ROC curve is basically used in binary classification problems and provides information about the quality of the model-classifier according to the relation of True-Positive and False-Positive predictions. More precisely, ROC curve is a plotted function of the false-positive rate for different cut-off points each one represents a ratio of sensitivity/specificity which will be described in the following formulas. Because of the nature of our dataset and our classifier, the final measure is the mean area under the six different ROC curves which are plotted regarding the six different models.

$$Sensitivity = \frac{True\ \ Positive}{True\ \ Positive + False\ \ \ Negative}$$
$$Specificity = \frac{True\ \ Negative}{True\ \ Negative + False\ \ Positive}$$

- **Sensitivity** is the probability that a predicted test result will be positive when the target characteristic (class) is present in ground-truth.
- **Specificity** is the probability that a predicted test result will be negative when the target characteristic (class) is absent in ground-truth.
- **ROC curve** visualizes the relation between True Positive (Sensitivity) rate and False Positive (1-Specificity) rate.

The following plots visualize six different ROC curves for the corresponding class using the outcome of Logistic Regression and Naive Bayes classifier. Mean Area Under ROC Curve using logistic regression model with binary relevance as multilabel representation is 0.961, while using Naive Bayes model is 0.955.

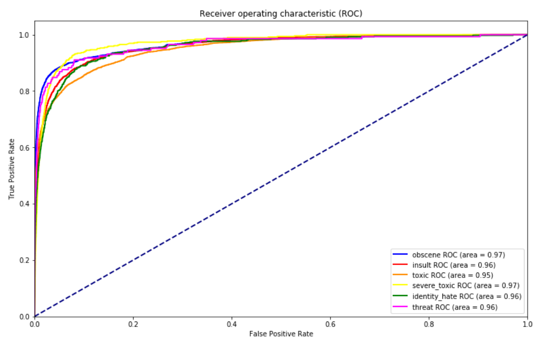

Figure 4: *Log Reg ROC Curve*

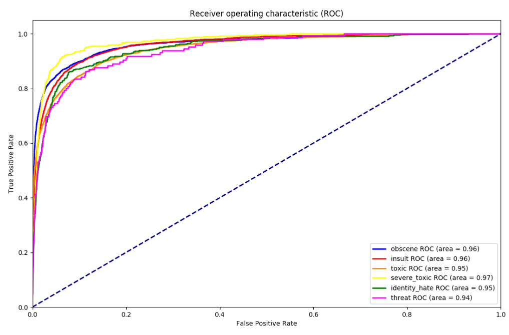

Figure 5: *Naive Bayes ROC Curve*

The results of the above plots are quite adequately because all curves tend to be near to the upper-left corner of the plot covering enough area. This means that the built models are quite robust and they can produce satisfactorily results which is very interesting because both models are very simple and naive. Nevertheless, both models produced good results.

6 Final Remarks

According to Figures 4 and 5, it is obvious that Logistic Regression produced better results than Naive Bayes. The former seems to be more appropriate model for this kind of classification and despite the fact that the classes were considered as independent, the model generated quite nice results. We can also cast an eye over the Tables 10 , 11 and understand thoroughly the quality of logistic regression in each class. The final score from Kaggle submission using the predictions of logistic regression was 0.956.

7 References

- Multi-Label Classification: An Overview
- Efficient Multi-label Classification with Many Labels
- Study and Analysis of Multi -Label Classification Methods in Data Mining
- Introduction to Information Retrieval *C.Manning 2009*
- Scikit-learn