DataSet Descrption 

The data contains a large number of Wikipedia comments which have been labeled by human raters for toxic behavior i.e. Whether a data point (comment) is toxic in six different labels.

The labels are toxic, severe\_toxic, obscene, threat, insult,

identity\_hate. The dataset has a train , test and testLabels files

. The train data files in which a comment is present in a label is shown by 1 otherwise 0 is filled. The test data file has comments only. The testLabel file has labels for comments in test files based on unique comment ids. A label can have 1 when a comment is present in a particular label, 0 when a comment is not present in a particular label, -1 indicates it was not used for scoring.The data is taken from Kaggle.com.

Methology

**ClassifiersUsing Weka**

Weka is a Data Mining tool that is developed in java. Weka provides GUI for applying standard data mining tasks like data preprocessing, clustering, classification, regression, visualization, and feature selection.

Input to weka-

The Attribute-Relation File Format (arff) is the standard format as the input file to the weka tool so we transform our train.csv to train.arff file and then open this file in weka.

## Using Weka NaiveBayesMultinomialText

Multinomial Naive Bayes for text data. It Operates only on String attributes. Other types of input attributes are accepted but ignored during training and classification

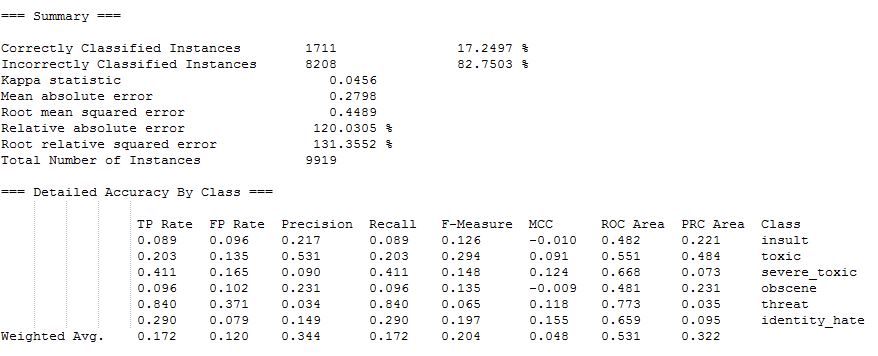
Comment classification can be mapped to this technique as there is a set of comments that are already labeled, the task is to predict the label for a new comment into one of the existing labels. We have a set of comments that are labeled into six labels i.e. toxic, severe\_toxic, obscene, threat, insult,

identity\_hate. The algorithm works estimate it the conditional probability of a particular word/term/token given a class/label, as the relative frequency of the term in comments belonging to class c. http://blog.datumbox.com/wp-content/uploads/2013/09/naive-bayes-maths7.png

**http://blog.datumbox.com/machine-learning-tutorial-the-naive-bayes-text-classifier/**

We Apply this classification algorithm on our dataset in Weka, the results we got are –

1.10fold



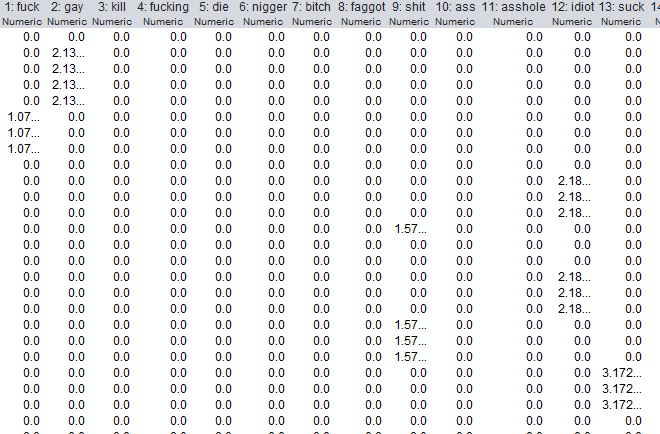
From the result above the Correctly Classified Instances are 40.46% with less true positive for every label except ‘threat’. So this algorithm works on the classification problem, but accuracy is very bad. So now tried to find the Numeric data corresponding to each comment or word in order to train the model on numeric data. So tried the next approach.

## Using Weka StringToWordVector

StringToWordVector is a filter in Preprocess tab in Weka that is used to convert string attributes into a set of numeric attributes representing word occurrence information from the text contained in the strings. Both cleaning and tokenization can be done using this filter. To make the provided comment classifiable, we need to do Feature extraction that is converting the normal text to a set of numeric features that can be used for the training of the classifier.

This filter includes the option to add stopwords file, words which we want to remove from our dataset on tokenization. It also includes the option to convert to word vector using the TF/IDF scores of each word in comments. For our model we have used this these two options of StringToWordVector . And then we do the Attribute Selection i.e. selecting the required features using the feature visualization method. The output dataset from this filter and feature selection is as shown in fig 3 in which each word is a feature and the matrix has TF/IDF scores

.



**Classifier**

**NaiveBayesMultinomial-**

Naive Bayes is a family of algorithms based on Bayes theorem in which the trained model predict the class of a given sample. These classifiers are the probabilistic classifier, which calculates the probability of each category using Bayes theorem, and the class with the highest probability will be predicted class.

One of the Classifier of Naïve Bayes family is NaiveBayesMultinomial classifier which is suitable for classification with discrete features like word counts or tf/idf scores for text classification. The multinomial distribution normally requires integer feature counts, but it works fine with fractional counts.

The data Matrix which comes as the output of StringToWordVector which works as the input for the NaiveBayesMultinomial. Now we train the model on 80 % of the data and keep 20%(known as 80-20 split) for the test purpose, to calculate accuracy. The output is that the model predicts with 38.9% accuracy.

This model is also not predicting the class of the test comments properly. So we understand that there is a problem with the approach which we are using for classification or there is some problem in understanding the problem.

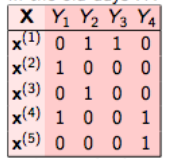
Then we ponder upon the data and came with a problem that the one comment can be a part of atmost six classes and this model trains that comment in all the classes in which it belongs. So the model is not getting trained properly and while predicting, the model will predict it into wrong class .

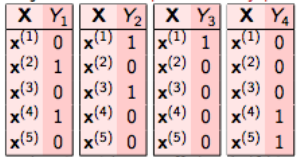
So we decided to use Multi-label classification for training out data i.e Binary Relevance and Classifier-Chains.

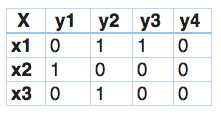
**Multilabel Classification**

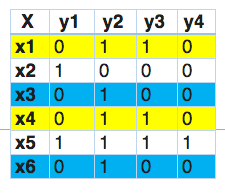
In our dataset, a single comment can have multiple classes. For example, the same instance can be obscene, threat and insult. To tackle this, we have to apply Multilabel classification techniques. Multilabel classification problems are the problems in which an instance can assign to various categories,  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. This method can be carried out in three different ways as:  
1. Binary Relevance  
2. Classifier Chains  
3. Label Powerset

3.1 Binary Relevance  
This technique treats each label as a separate single class classification problem. For example, let us consider a case as shown below. We have the data set like this, where X is the instance and Y’s are the classes.

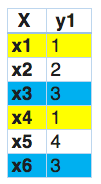


In binary relevance, this problem is broken into 4 different single class classification problems as shown in the figure below.  
  
  
  
Drawback:- It doesn’t consider labels correlation because it treats every target variable independently.  
  
  
3.2 Classifier Chains  
The first classifier is trained just on the input data and then each next classifier is trained on the input space and all the previous classifiers in the chain.   
Let’s try to this understand this by an example. In the dataset given below, we have X as the input space and Y’s like the labels.

  
  
In classifier chains, this problem would be transformed into 4 different single label problems, just like shown below. Here yellow-coloured is the input space and the white part represent the target variable.

This is quite similar to binary relevance, the only difference being it forms chains in order to preserve label correlation. So, let’s try to implement this using multi-learn library.  
Advantage:- It considers dependencies between labels. It forms chains in order to preserve label correlation.  
  
3.3 Label Powerset  
In this, we transform the problem into a multi-class problem with one multi-class classifier is trained on all unique label combinations found in the training data. For example  
  
  
In this, we find that x1 and x4 have the same labels, similarly, x3 and x6 have the same set of labels. So, label powerset transforms this problem into a single multi-class problem as shown

below

  
  
  
So, label powerset has given a unique class to every possible label combination that is present in the training set. This approach does take possible correlations between class labels into account. More commonly this approach is called the label-powerset method because it considers each member of the power set of labels in the training set as a single label.  
Drawback :- This method needs the worst case (2^|C|) classifiers and has a high computational complexity. However when the number of classes increases the number of distinct label combinations can grow exponentially. This easily leads to a combinatorial explosion and thus computational infeasibility.

Result

