Toxic Comment Classification

1.INTRODUCTION

1.1. Abstract

Social networking sites are the source of most of the recent trend. Almost every human is one way or another attached and affected by these sites. Leading platforms gives people freedom to express themselves through posting of comments and various media. Although that is good in an idea world, where no one is expected to abuse, such freedom but in real world often exactly opposite is observed. Such abuse of freedom often leads to hate spreading, racial slurring or verbal assault. These dangers refrain many people from sharing their opinion or sharing any media for that matter which is harmful for leading platform as well as our society. Google along with Jigsaw started working together to solve this problem[1].Their idea was to build an application that could detect and remove or limit verbal abuse which crosses the 'terms of use' of a particular site. Their service is called Conversational AI

Conversational AI uses machine learning, which provides a distinct advantage over other technical solutions, and detect comments that contain toxic content. Conversation AI's API (works only on English comments right now) has been used in real world to detect toxic comments. Natural language processing along machine learning is a challenge and hence an interesting area that gives a lot of scope to experiment. The same reason inspired us to take this project. We wish to test different machine learning algorithms and text processing methods which perhaps give us comparable performance that of 'Conversational AI'. Our solution makes use of natural language processing tools for preprocessing the data and machine learning approaches were used to train a model that could detect the toxicity of the comments.

Wikipedia provides a dataset which contains comments posted by users. Each of those comments are voted by users for its toxicity. Thus a single comment may have multiple toxicity at same time - And that is where things gets complicated. Semantic has lots of considerations when it comes to converting text data into numerical data like sequence and order of word, bold words, obscurely used abusive words which won't be detected by a dictionary.

1.2. PROBLEM OVERVIEW:

A common feature that every social media provides is a section where users can comment on someone's post. It has been seen that there has been a misuse of the comment section, and some people use it to spread hate comments, and also threaten the author. A way to say if the comment is toxic can be a game changer and people can post/share their views & opinions without any fear.

Our aim is to build a bot that can predict the probability of a comment in any social media is toxic. The toxicity is divided into 6 categories:

1. Toxic
2. Severe_toxic
3. Obscene
4. Threat
5. Insult
6. identity_hate.

1.3. Problem Break down

- Extract comment from dataset / sites
- Identify if the comment is toxic
- If it is, classify into one of 6 classes
- Also predict the probability with which the toxic comment belongs to each class
- Extract a report in a readable format

The problem can be seen as both classification and regression problem. We are supposed to classify the comments as well as find the probabilities. Also, as the dataset is labelled, it is a supervised learning problem.

1.4. Proposed Methods

We have split the data into train set and test set with 80:20 proportion. Further, we divided the train data into train and validation dataset.

Overview of tasks performed:

1. Split the data to Train, Test and validation

2. Method

: a. Convert text/comments to TF-IDF using Train data

b. Generate TF-IDF for Test and Validation using features from Train

c. Build Classification Models on Train

1. Logistic Regression
2. Naive Bayes
3. Support Vector Machines(SVM)

d. Evaluate on Validation data and tune the models. Repeat to find best parameters.

e. Test performance of final model on Test dataset

2. Data Processing

The data consist of text comments identifier and its toxicity level that many users has recorded. Text comments contained may symbols, emoji, dates, codes (E.g. UTC) and escape characters like '\n', '\t other than normal words.

1 Case of alpha bate doesn't help so we are converting to lower case letter,

2 multiple space converted to single space

3.char other than alpha bate are removed

4. stop word removed ,

5.removed links,

2.1 TD/IDF

For converting connects into numerical features we decided to used unigram model that consider a separate word as a feature , this is particularly helpful due to the fact that model will see many more event of a particular word and its corresponding target value than sequence of words. In other words given the medium size of corpus (159571 comments), single word feature model can be more general and represent other dataset as a result we have better trained model with less over fitting.

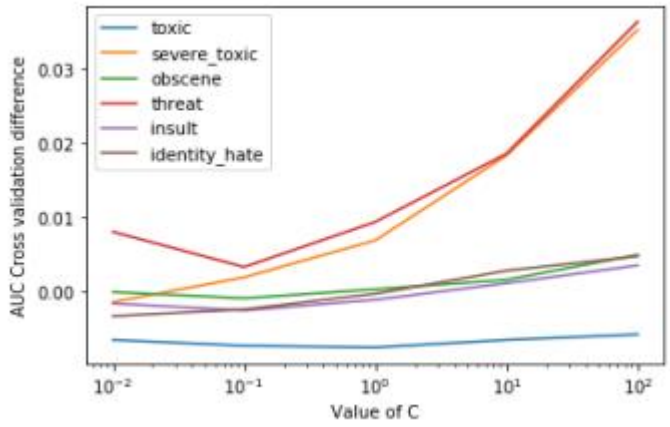
We converted comments to TF_IDF matrix using libraries provided by Sk-learn. TF-IDF converted whole document (collection of comments) into N number of unique terms T={t1,t2,t3,t4……tN} and note their respective frequencies F={f1,f2,f3…..fN}. In our case value of N is1134255 creating feature space of that value. At first it seem impossible to have huge number of unique words present in comments as the generally we don't use these many words. But as we can see in the following screenshot many obscure words or the

words which has no meaning (perhaps in any language) or just string of same characters are used in comments. To make training process computationally lighter and easier we decided to remove some of the words from the document. We removed the words that are used very commonly and have mundane meaning. For example, in a sentence “I have a pen” the words ‘i’, ‘have’, ‘a’ holds no derogatory meaning and used to make sentence grammatically perfect. As a result there was considerable reduction in number of features.

3. Modelling

3. 1. LOGISTIC REGRESSION:

Logistic regression is a statistical **model** that in its basic form uses a **logistic** function to **model** a binary dependent variable, although many more complex extensions exist. In **regression** analysis, **logistic regression** (or **logit regression**) is estimating the parameters of a **logistic model** (a form of binary **regression**).



ADVANTAGES:

- Logistic Regression is **one of the simplest machine learning algorithms** and is easy to implement yet provides great training efficiency in some cases. Also due to these reasons, training a model with this algorithm doesn't require high computation power.
- The predicted parameters (trained weights) give **inference about the importance of each feature**. The direction of association i.e. positive or negative is also given. So we can use logistic regression to find out the relationship between the features.

- This algorithm allows models to be **updated easily to reflect new data**, unlike decision trees or support vector machines. The update can be done using stochastic gradient descent.
- Logistic Regression **outputs well-calibrated probabilities** along with classification results. This is an advantage over models that only give the final classification as results. If a training example has a 95% probability for a class, and another has a 55% probability for the same class, we get an inference about which training examples are more accurate for the formulated problem.
- In a **low dimensional dataset** having a sufficient number of training examples, logistic regression is **less prone to over-fitting**.
- Rather than straight away starting with a complex model, logistic regression is sometimes **used as a benchmark model to measure performance**, as it is relatively quick and easy to implement.
- Logistic Regression proves to be **very efficient** when the dataset has **features that are linearly separable**.

DISADVANTAGES:

- Logistic Regression is a statistical analysis model that attempts to predict precise probabilistic outcomes based on independent features. On **high dimensional datasets**, this may lead to the model being **over-fit on the training set**, which means overstating the accuracy of predictions on the training set and thus the model **may not be able to predict accurate results on the test set**. This usually happens in the case when the model is trained on little training data with lots of features. So on high dimensional datasets, Regularization techniques should be considered to avoid over-fitting (but this makes the model complex). Very high regularization factors may even lead to the model being under-fit on the training data.
- **Non linear problems can't be solved** with logistic regression **since it has a linear decision surface**. Linearly separable data is rarely found in real world scenarios. So the transformation of non linear features is required which can be done by increasing the number of features such that the data becomes linearly separable in higher dimension

3.2 Naive Bayes

Naive Bayes uses the technique for prediction where it assumes probability of every feature to be independent of each other. The parameters taken into consideration were

alpha and fit_prior Alpha is an additive smoothing parameter where 0 means no smoothing and fit_prior indicated whether the class prior probabilities are to learnt or not. If it is false uniform prior would be used

Bayes Theorem:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Using Bayes theorem, we can find the probability of **A** happening, given that **B** has occurred. Here, **B** is the evidence and **A** is the hypothesis. The assumption made here is that the predictors/features are independent. That is presence of one particular feature does not affect the other. Hence it is called naive**.**

Types of Naive Bayes Classifier:

Multinomial Naive Bayes:

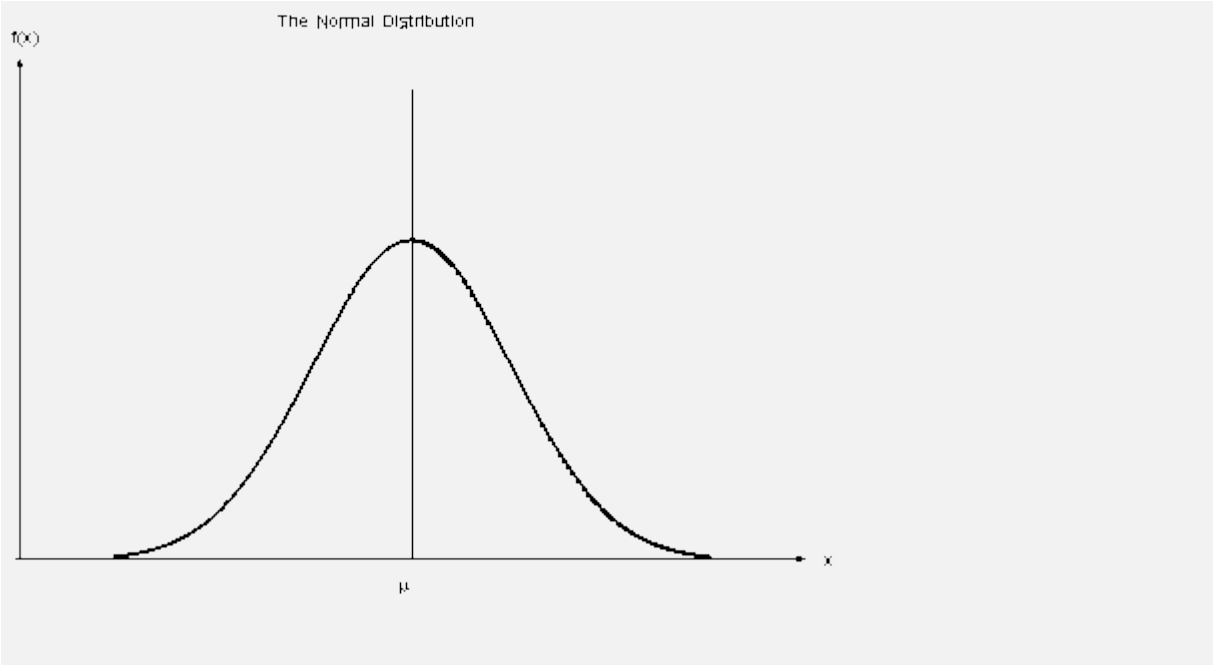
This is mostly used for document classification problem, i.e whether a document belongs to the category of sports, politics, technology etc. The features/predictors used by the classifier are the frequency of the words present in the document.

Bernoulli Naive Bayes:

This is similar to the multinomial naive bayes but the predictors are boolean variables. The parameters that we use to predict the class variable take up only values yes or no, for example if a word occurs in the text or not.

Gaussian Naive Bayes:

When the predictors take up a continuous value and are not discrete, ASSUME we that these values are sampled from a gaussian distribution.



Gaussian Distribution(Normal Distribution)

Since the way the values are present in the dataset changes, the formula for conditional probability changes to,

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} exp\left(-\frac{(x_i-\mu_y)^2}{2\sigma_y^2}\right)$$

3.3 Support Vector Machine