GEN 511 - MACHINE LEARNING
PROJECT

Toxic Comment Classification

Team Name: *Akinator*
Mohammad Khalid – IMT2017028
Anurag Pendyala – IMT2017504
K. Sailesh – IMT2017524

Contents

1 Abstract

Social Media has changed the way people interact. In the past decade, Texting and Chatting has taken over the preferred to talking for communication. Emails have also become the standard for formal communication all over the world.

Online forums, because of their convenience, vivid diversity and accessibility, have changed the way people get their daily feed and keep up to things going on in a particular community. Significant internet forums communities have converged around topics ranging from medicine to technology, and vocations and hobbies.

Healthy discussions on various topics are very important. It gives the user more information about that particular topic. But this can turn difficult as well. Off-late, it can been seen that, there are enough instances where people have face abuse and harassment online. This may stop the people to express their ideas and opinions on the topic. Many open discussion platforms struggle to stop these abusive opinions.

Hence, they try to work it down manually or just shutdown the comment section. The latter is chosen most of the time. But if that happens, the information a user gets is only a one-way interaction. The user can not ask any questions or present his ideas. This is a loss for most of people who want to learn something new or express something that they already know.

Many people have tried to come up with a solution to filter out these offensive and hurting comments. It is an ever-growing Machine Learning problem. These kind of language related problems come under the category of Natural Language Processing(NLP).

2 Problem Statement

Build a multi-headed model capable of detecting six types of of toxicity in texts like Toxic, Severe Toxic, Threat, Obscenity, Insults, and Identity-based hate. Predict the probability of the comment belonging to a particular class.

Toxic Comment Classification – http://www.kaggle.com/c/iiitb-toxic-comment

3 Data Set

Wikipedia is an online encyclopedia where people generate and share information. It has a comment section as well that lets people comment on articles and share their ideas. Often than not some or the other comments pop up which destroy the whole purpose of having such a thing in the first place.

A data Set with Wikipedia comments which have been labeled by human raters for toxic behavior is used.

The types of toxicity are:

- toxic
- severe_toxic
- obscene
- threat
- insult
- identity_hate

Every data point has an ID, comment text and 6 classes given above. These classes are specified for the training set and are to be predicted for the testing set.

- *train.csv* has the training set, contains comments with their binary labels. There are around 120,000 points with 8 features.
- *test.csv* has test data with around 47,000 points with an ID and text.

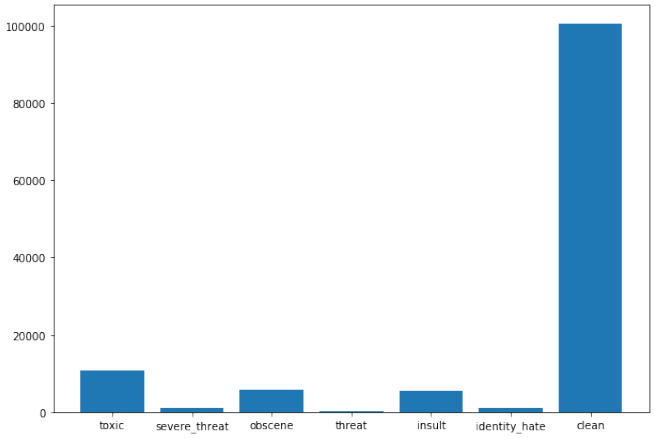
4 Exploratory Data Analysis

The training data set has around 120,000 data points and 8 columns. These include ID, comment_text and 6 levels of toxicity one-hot encoded. A new feature named *clean* was added to the data set. It's field value is 1 if the data point does not belong to any of the 6 levels of toxicity. If not, it is left at 0.

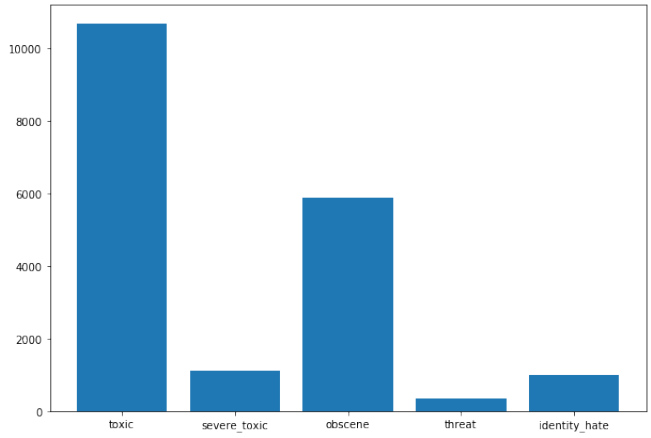
The following observations have been made from the frequency plots, correlation plots and Venn diagrams of the distribution of data points in the data set.

The given data set is highly unbalanced.

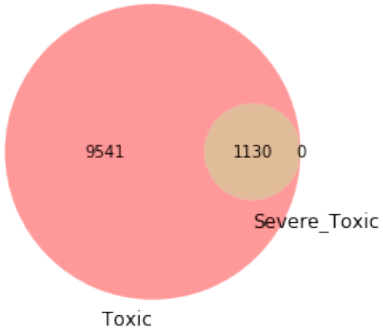
- Around 100,000 out of 120,000 data points belonged to *clean* class. Making the data set heavily skewed towards the clean class.



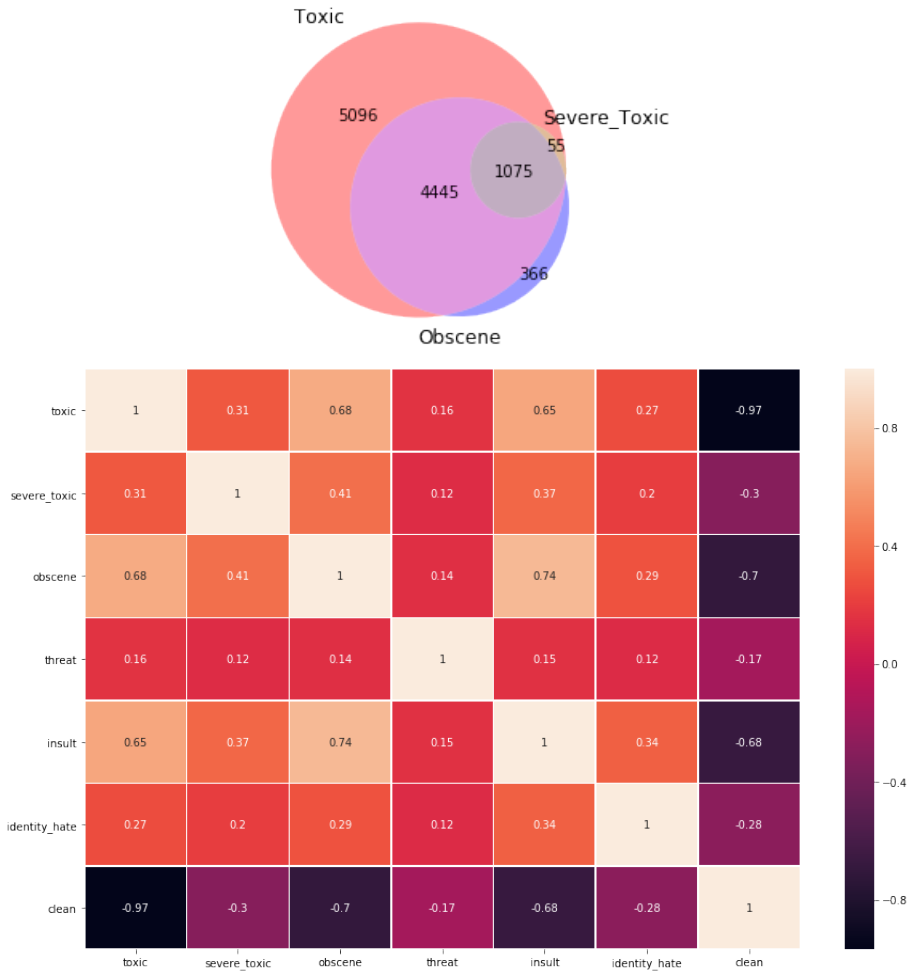
- Apart from the data points belonging to clean class, majority of the data points belonged to the toxic class.



- The class **severe toxic** was a proper subset of the **toxic** class this is clearly evident from the plot below and class labels.



- Most of the data, apart from the clean data, belonged to the intersection of three classes, **Toxic, Severe Toxic** and **Obscene**.



The following methods were used to tackle and resolve the above mentioned issues.

- **RE-SAMPLE THE DATA** : There are 2 majors approaches to re-sample the data. The data from the minority class data can be replicated and new points can be synthesized to balance the data set. The other is to under sample the majority class data.

  Problem faced : On over sampling, the correlation between features is lost. On under sampling, the number of training points reduced drastically. This could lead to the model not having enough data to learn and predict correctly.

- **ADD ADDITIONAL DATA** : Adding more data points to the data set can help balance the data set. The data being dealt with comes under a popular category of *Hate Speech*. The link for the additional data used to train the model has been attached in the references section.

5 DATA PREPROCESSING

After exploring the data and fixing the issues related to the distribution, Now comes the part where noise/redundancies in the data is to be eliminated.

Data preprocessing is an important step because having clean data will help improve the models ability to learn

NLTK contains a suite of libraries and programs for symbolic and statistical Natural Language Processing (NLP) programming language. Most of the data preprocessing is being done with help of libraries and packages from NLTK. The following steps have been taken to clean the data:

- **CONVERTING THE TEXT TO LOWER CASE**
  - The case of the words does not affect the toxicity of the sentence in any way thus making this attribute redundant. Hence, all the words have been converted to lowercase.
- **STOP WORDS REMOVAL**
  - Words like *this, an, is*, etc which are inevitably present in almost every sentence do not make any contribution to toxicity of a sentence. They also clutter the feature vector if frequency is used as a feature.
  - List of stop words from NLTK package has been used to identify and remove stop words from the training data.
- **SPELL CORRECTION**
  - The data contains a lot of text which cannot be found in a dictionary. These include a combination of words without spaces in between and Misspelled words
  - SpellCheck : SymSpell and PySpellCheck have been used to correct spellings of the words. These replace a word with its nearest possible substitute in case of a misspelling.
  - Words with length more than 20 were removed. This did not help much.
  - Words which have a letter repeating more than thrice continuously have been removed.
- **REMOVAL OF PUNCTUATION AND NUMERICAL CONTENT**
  - Punctuation and numerical data just like stop words are redundant and were removed.
- **REMOVAL OF LINKS, HTTP FORMATTED TEXT**
  - Links contain texts which are not cognizable and would not contribute to toxic comments. They would induce more confusion than do any help.

- **LEMMATIZATION AND STEMMING**
  - Stemming and Lemmetization play an important part in text processing. Words like *ask* and *asked* make the same sense(unless one is trying to identify the tense or context) thus reducing them to one feature reduces the complexity and redundancy. Stemming reduces word-forms to (pseudo)stems, whereas lemmatization reduces the word-forms to linguistically valid lemmas.
  - Stemming
    * Porter stemmer and Snowball stemmer have been used for stemming.
    * These improved the performance when compared to before.
    * This technique is not always good as it might be ineffective in the case of (eat, ate) or (see, saw), in the latter case the algorithm might consider 's' as a valid prefix.
  - Lemmatization
    * Lemmatization from NLTK and TextBlob have been used to perform Lemmatization.
    * These further improved the performance when compared to stemming.
    * Lemmatization with POSTAG (Parts of Speech Identification) gave the best results as chopping and changing is done more sensibly with help of parts of speech.

6 FEATURE EXTRACTION

Text as it is(after preprocessing) can not be used directly used for training. A large number of models are optimized to perform better on numerical data. A few approaches have been tried to convert the text to numerical data.

- **BAG OF WORDS** : This method involves creating a vector of all the unique words which occur in the data. Then each sentence is mapped to a feature vector which contains the frequencies of the words in the sentence. Basic method. Not very efficient
- **TF-IDF VECTORIZER** : 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 or corpus.

  The formula for weight score of a word in the feature vector is:

$$w_{i,j} = tf_{i,j} \times log\left(\frac{N}{df_i}\right)$$

$tf_{i,j}$ = Number of occurrences of i in j.

df_i = Number of documents containing i.

N = Total number of documents.

The terms with higher weight scores are considered to be more important.

- Bag of n-grams is a natural extension of bag of words. An n-gram is simply any sequence of n tokens in a sentence.

Additional features like length, punctuation's, length of sentence, Case of the words were used in addition to the already available columns.

The above mentioned methods give huge feature vectors of the order of a lakh.

Word tokenization and char tokenization with varying parameters have been used to extract features from text.

PCA used here to reduce the number of features for quick computation and reduction in complexity of the model.

7 Model Building

In order to check the performance of the model, the training data has been split. 80% of the total data has been used for training the model and the remaining 20% has been used to test it. The parameter ROC is used to check the performance while validating.

Binary classification on each class has been preferred over one versus rest classifier as there is no limitation that a comment belongs to only one class. Also the data is highly skewed.

The following models have been used to classify the text into different classes.

- Logistic Regression
- Random Forest Classifier
- ComplementNaiveBayes
- SVM
- Ridge Classifier
- SGD Classifier

Ensemble methods are techniques that create multiple models and then combine them to produce improved results.

VotingEnsembler and EasyEnsembleClassifier have been used in addition to the above mentioned classifiers.

EasyEnsembleClassifier takes a base classifer and creates similar models to that of the base classifier. The predictions are made by wrapping around these classifiers.

The above listed models have been each used as base classifier to create a new EasyEnsemblerClassifier

Voting classifier is not an actual classifier but a wrapper for set of different ones that are trained and valuated in parallel in order to exploit the different peculiarities of each algorithm.

Voting Classifiers with the above listed models have also been used.

8 Analysis

Models have been tested with various parameters and hyper-parameters

The addition of features like length, punctuation in a sentence has not given any significant boost to the model as these parameters are not strictly different between toxic and normal sentence

PCA has reduced the computation to a large extent while reducing the performance of the models as projecting 1000's to features to a few small is showing its effect.

Lemmetization with POSTAG has been the best of the Text Normalization technique of all the ones used.

Ridge classifier has not be used to make a submission because it predicts only the binary labels and not the probability of belonging to a class.

The Table below contains the scores for a few models with different parameters and hyper parameters.

Sno	Model	Feature Parameters	Hyper Parameters	ROC Score
1	Logistic Regression (without PCA)	Word Tokens 50k Char Tokens 25k Word ngram 1,1 Char ngram 2,4	penalty='l2' solver='warn'	98.5

2	Logistic Regression (without PCA)	Word Tokens 50k Char Tokens 25k Word ngram 1,2 Char ngram 2,6	penalty='l2' solver='warn'	98.1
3	Logistic Regression (with PCA)	Word Tokens 50k Char Tokens 25k Word ngram 1,1 Char ngram 2,4	penalty='l2' solver='warn'	96.7
4	SGD Classifier (without PCA)	Word Tokens 50k Char Tokens 25k Word ngram 1,1 Char ngram 2,4	alpha=0.0001 epsilon=0.1	96.2
5	ComplementNB (without PCA)	Word Tokens 50k Char Tokens 25k Word ngram 1,1 Char ngram 2,4	alpha=1.0 class_prior=None fit_prior=True norm=False)	94.3
6	Random Forest (without PCA)	Word Tokens 50k Char Tokens 25k Word ngram 1,1 Char ngram 2,4	criterion='gini' min_samples_split=2	95.8
7	Random Forest (with PCA)	Word Tokens 50k Char Tokens 25k Word ngram 1,1 Char ngram 2,4	criterion='gini' min_samples_split=2	95.1
8	Voting Classifier1	Word Tokens 50k Char Tokens 25k Word ngram 1,1 Char ngram 2,4	Logistic Regression(1) EasyEnsembler1 (10) Random Forest (6) EasyEnsembler3 (11) SGDModel (4) EasyEnsembler4 (12) CNBmodel (5) Voting = Hard	97.9
9	Voting Classifier2	Word Tokens 50k Char Tokens 25k Word ngram 1,1 Char ngram 2,4	Logistic Regression(1) EasyEnsembler1 (10) Random Forest (6) EasyEnsembler3 (11) SGDModel (4) EasyEnsembler4 (12) CNBmodel (5) Voting = Soft	97.3

10	EasyEnsembler1 (with Log reg)	Word Tokens 50k Char Tokens 25k Word ngram 1,1 Char ngram 2,4	C=2 solver= 'sag' max_iter=500	98.2
11	EasyEnsembler3 (with RandomForest)	Word Tokens 50k Char Tokens 25k Word ngram 1,1 Char ngram 2,4	criterion='gini' max_depth=100	96.3
12	EasyEnsembler4 (with SGD classifier)	Word Tokens 50k Char Tokens 25k Word ngram 1,1 Char ngram 2,4	alpha=.0002 max_iter=500 penalty="l2"	97.2

9 Conclusion

Extraction of proper features and preprocessing data plays a major role in building a good model. Doing data analysis before training the model is very important for better prediction as handling skewed data involves additional steps. More the data used for training the better the model generalizes.

This problem can be extended to also classify texts, posts and other text based modules to make reading a better experience.

10 References and Citations

- Lemmatization Examples
- Text Classification
- Spellcheck using python
- Text Preprocessing
- Word Embeddings Exploration and Exploitation
- Implement Text Classification in python
- Twitter comments data set from HateSpeechData