Toxic Comments Classification

Advait Lonkar
IMT2017002
International Institute of Information Technology, Bangalore

Gandharv Suri
IMT2017017
International Institute of Information Technology, Bangalore

Mili Goyal
IMT2017513
International Institute of Information Technology, Bangalore

***Abstract*—The problem originates from the various online forums, where people participate and make commments. The hosting organizations are constantly on the lookout for abusive, insulting or even hate-based comments to ensure that the conversations on their forums stay civil.**
We are presenting a machine learning model using traditional methods to look into this problem and try to come with a solution. The task is to build a model which could predict whether a comment is clean, provided various categories into which the comments are divided into : toxic, severe toxic, obscene, threat, insult, and identity hate.
Keywords : NLP, Stemming, Lemmatization, TF-IDF Vectorizer.

I. Introduction

Cyberbullying is a form of harassment using electronic means. Also known as online bullying, it has become increasingly common among teenagers.
Cyberbullying is an important new kind of bullying, with some different characteristics from traditional bullying. Cyberbullying has clearly diversified beyond bullying by text messages or emails. Some cyberbullying can combine the anonymity of the aggressor found in conventional indirect aggression with the targeted attack on the victim found in conventional direct aggression [1].

One such form of cyberbullying is the conversational toxicity on the online forums of various websites. Conversational toxicity is an issue that can lead people not to genuinely express themselves and to stop seeking others' opinions out of fear of abuse/harassment .
The goal of this project is to identify toxicity in the comments collected from various online forums, which could be used to prevent users from posting hurtful/abusive comments, and create more civil arguments when engaging with other participants [2].

II. Problem Statement

Given a comment_text, which was commented by a user on an online platform, predict whether the comment is clean or not, provided if it belongs to one more categories :
toxic, severe-toxic, obscene, threat, insult, identity-hate.
This is a binary-classification problem with multiple labels.
This means that any comment will either belong to one or more categories mentioned above or else it will be labelled *clean*.
This problem falls under the category of **Natural Language Processing**.

III. Data

We have a dataset of 111,698 training samples. The data we have, is from the live competition - Toxic Comments Classification on Kaggle [3]. We use python's libraries matplotib and seaborn to visualize the data.
The dataset consists of the following fields:

- id : An 8-digit integer value which identifies which person wrote the comment.
- comment_text : A text field, which can be of multiple lines, containing the exact comment.
- toxic : binary label containing 0 or 1 (0 for no and 1 for yes)
- severe_toxic : binary label containing 0 or 1
- obscene : binary label containing 0 or 1
- threat : binary label containing 0 or 1
- insult : binary label containing 0 or 1
- identity_hate : binary label containing 0 or 1

Out of all these fields, the comment_text field will be preprocessed and fit into various models/classifiers to predict whether the comment is clean or not.
An observation on the frequency of occurence of multi-labelled data shows that most of the comments are clean.

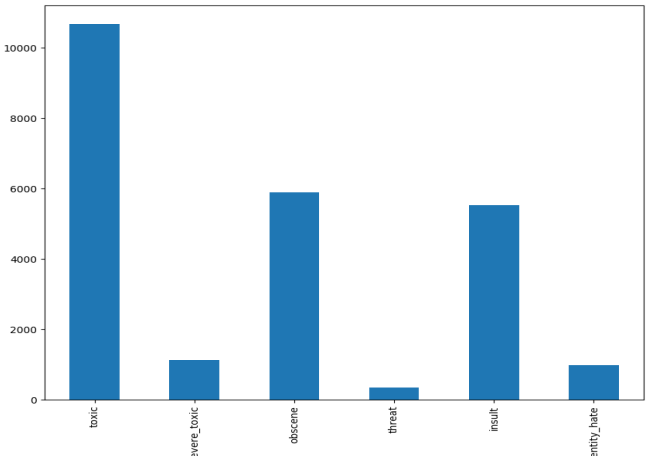

Fig. 1: Frequency of labels.

As shown in Fig. 1, most of the labelled comments belong to the toxic category. Obscene and insult categories too have a good share of comments. While severe-toxic, threat and identity-hate categories make up for the remaining comments with very few comments distributed amongst them.

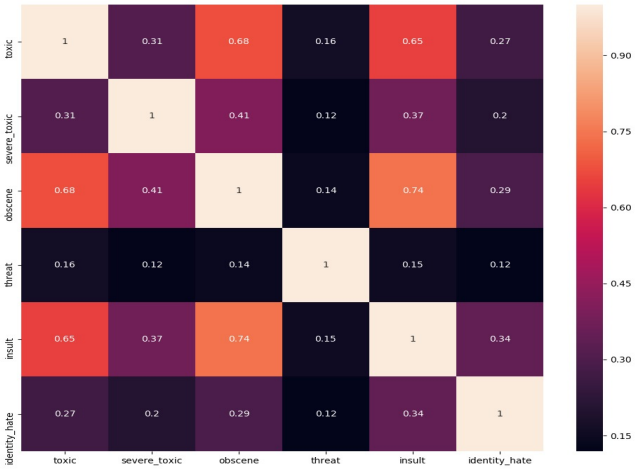

Fig. 2: Correlation matrix for all the labels.

The correlation matrix shown in Fig. 2 shows the correlation coefficient between any two labels - which in turn dependence of the variables on each other. For example, the labels threat and obscene have a correlation coefficient of 0.74 ,and toxic and obscene have a correlation coefficient of 0.68, showing strong dependence whereas the labels threat and insult have a correlation coefficient of 0.15, showing weak dependence.

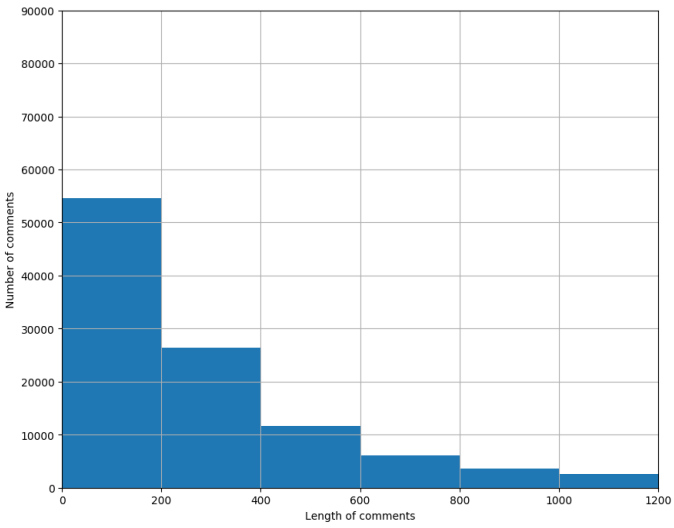

Fig. 3: Length of comments.

Fig. 3 shows the distribution of number of comments against the length of the comments.
The majority of comments are less than 400 characters long. Therefore, all the comments above 400 characters in length make up the outliers for the data. This is because the comments with less than 400 charcters characterize the dataset and the longer comments may not fit the characteristics.
Hence the longer comments ($>$ 400 characters) need to be removed.

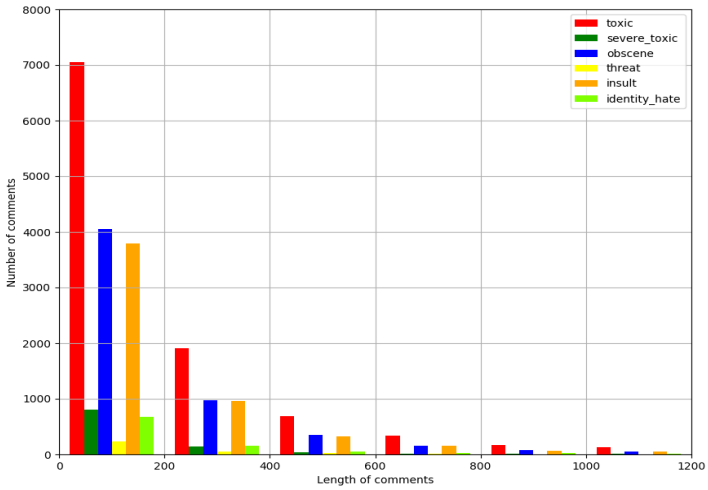

Fig. 4: Length of classified comments.

Fig. 4 shows the distribution of classified comments against the length of comments, with a bucket size of 200 i.e. the comments are classified for the lengths in the range of 0-200, 200-400, 400-600 etc. characters.
This ditribution clearly shows that most of the toxic comments are shorter than 400 characters in length. So it strengthens the claim to remove comments longer than 400 characters.

IV. Data Pre-preocessing

Before any pre-processing, we removed all the comments longer than 400 characters.
Then, pre-processing involved the following steps:

1) **Basic Cleaning of text :**
   a) *Lowercasing all the alphabet* : All the comments are converted into lowercase, since varying capitalization of the same words in a big dataset may lead to different types of output.
   b) *Noise removal* : Noise removal is about removing characters digits and pieces of text that can interfere with your text analysis.
      i) Removing html markup.
      ii) Removing non-ASCII characters and digits.
      iii) Removing excess whitespace.
2) **Removing stop-words** : Stop words are a set of commonly used words in a language. Examples of stop words in English are a, the, is, are and etc. The intuition behind using stop words is that, by removing low information words from text, we can focus on the important words instead [4].
3) **Stemming** Stemming is the process of reducing inflection in words (e.g. troubled, troubles) to their root form (e.g. trouble). The root in this case may not be a real root word, but just a canonical form of the original word. Stemming uses a crude heuristic process that chops off the ends of words in the hope of correctly transforming words into its root form [4]. We used the Snowball Stemmer to achieve it.

4) **Lemmatization** Lemmatization on the surface is very similar to stemming, where the goal is to remove inflections and map a word to its root form. The only difference is that, lemmatization tries to do it the proper way. It doesnt just chop things off, it actually transforms words to the actual root. For example, the word *better* would map to *good* [4].

V. Features Extraction

TF-IDF Vectorizer: It is a numerical statistic that is intended to reflect the importance of a word in a document of a collection or corpus. TF-IDF vectors are calculated in the following way:

1) Calculate term frequency (TF) in each document.
2) Calculate the inverse document frequency (IDF): Take the total number of documents divided by the number of documents containing the word.
3) Iterate each document and count how often each word appears.
4) Calculate TF-IDF: multiply TF and IDF together.

$$TF(t) = N(t)/N.$$
$$IDF(t) = log_e(D/D(t)).$$
$$TF - IDF = TF \times IDF$$

where,
N(t) = Number of times term t appears in a document,
N = Total number of terms in the document,
D = Total number of documents,
D(t) = Number of documents with term t in it.

To extract the features from the comments we used TF-IDF vectorizer over the combined text of train and test in order to capture all the words occurring in the data with maximum features as 30,000 for an n gram of (1,1). In addition to using TF-IDF vectorizer for the words we also employed it on character n-grams of these words (1-4 including spaces) with maximum features of 23,000 to model the types of conscious or unconscious abbreviations or reductions of offensive words by the users. Then stacked both these matrices using scipys hstack.

VI. Model Building

We used the VotingClassifier ensembling technique. The idea behind the VotingClassifier is to combine conceptually different machine learning classifiers and use a majority vote or the average predicted probabilities (soft vote) to predict the class labels. Such a classifier can be useful for a set of equally well performing model in order to balance out their individual weaknesses.
The following models were used as a part of the VotingClassifier ensemble [5]:

1) **EasyEnsembleClassifier (base-estimator - LogisticRegression)** :
The EasyEnsembleClassifier allows to bag AdaBoost learners which are trained on balanced bootstrap samples. But in this case, we used LogisticRegression as the inner learner for this bagging algorithm.

2) **EasyEnsembleClassifier (base-estimator - SGDClassifier)** :
The EasyEnsembleClassifier allows to bag AdaBoost learners which are trained on balanced bootstrap samples. But in this case, we used SGDClassifier as the inner learner for this bagging algorithm.

3) **LogisticRegression**
Logistic regression, despite its name, is a linear model for classification rather than regression. Logistic regression is also known in the literature as logit regression, maximum-entropy classification (MaxEnt) or the log-linear classifier. In this model, the probabilities describing the possible outcomes of a single trial are modeled using a logistic function.

4) **RandomForestClassifier**
A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.

VII. Results and Conclusion

We got a final accuracy of 98.94 % (as displayed on the Kaggle leaderboard) through the VotingClassifier's result, which placed our team on the 4^{th} place on the leaderboard of the IIIT-B Toxic Comments Classification competition hosted on Kaggle.

TABLE I: Models and their accuracies

Classifier	Accuracy
EasyEnsembleClassifier (LogisticRegression)	98.75%
EasyEnsembleClassifier (SGDClassifier)	98.87%
LogisticRegression	98.99%
RandomForestClassifier	99.12%

The detailed results are shown in TABLE I.
The individual models used in the VotingClassifier all show accuracies of 98.2% to 99.1%.

References

[1] Peter K Smith, Jess Mahdavi, Manuel Carvalho, Sonja Fisher, Shanette Russell, and Neil Tippett. Cyberbullying: Its nature and impact in secondary school pupils. *Journal of child psychology and psychiatry*, 49(4):376–385, 2008.
[2] Kevin Khieu and Neha Narwal. Detecting and classifying toxic comments.
[3] Dataset from Kaggle competition :. https://www.kaggle.com/c/iiitb-toxic-comment/data.
[4] Aditya Jain, Gandhar Kulkarni, and Vraj Shah. Natural language processing. *Int. J. Comput. Sci. Eng*, 6(1), 2018.
[5] Scikit-Learn. https://scikit-learn.org.