CSCI 6509: Advanced Topics in Natural Language Processing

Toxicity Detection using NLP

Group 4

Sahil Fruitwala, B00844489
Abhijeet Singh, B00833813
Joel Joseph Thomas, B00843518
Punarva Vyas, B00841566

Table of Contents

ABSTRACT

In today’s world, humans from all over the world use online portals and services to tackle their real needs. Especially, it is considered as an integral part of people where their businesses are operated online. The E-commerce industry is one of the fastest and largest growing industry in the world. In spite of the fact, E-commerce industry is constantly challenged and under pressure with customer’s product or service, feedbacks and reviews. The positive and derogatory feedbacks and reviews of their products is crucial for companies which help them to analyze and provide useful recommendations for improvisation and enhancement. Likewise, social networking websites and other online communities are also associated with mixture of positive and negative comments from different users on various things. In this project, we have developed a system with the Identification block which would identify any kind of derogatory or disrespectful comments. This system is a necessity for social networking websites and online communities existing in the digital world. This project focuses on analyzing any piece of textual data consisting of various forms of online harassments and distinguish it as toxic or non-toxic.

INTRODUCTION

In the world we live right now, every business uses sentiment analysis on their products to understand the insights of what customers or users think of their products provided. For example, Amazon which is an E-commerce company does sentiment analysis on their products using customer comments and reviews. By performing those analysis on all of their products, they get to recognize whether the products are successful or not. At first, there are three main kinds of sentiments which are taken into consideration for analysis i.e. positive, negative and neutral sentiments. However, as the Natural Language Processing (NLP) field and Natural Language Understanding (NLU) progressed, two more sentiments toxicity and sarcasm were included in the list of sentiments. In the past few years many researches were done and still some are in progress on sarcasm detection and generation. On the other hand, the researches performed on toxicity detection were limited and narrow. The purpose of Toxicity detection is to identify hate, threat, abusiveness and many more in the language. The implementation of Toxicity detection can be used in various online platform such as social networking and E-commerce websites. For example, when someone reports a tweet or a comment, the toxicity detection system will detect the toxicity in text algorithms with the help of NLP word embedding techniques and later those tweets or comments are discarded automatically. It is important to find the most suitable model for Toxic detection in order to achieve the best results, and to do so we have experimented and performed trial runs on various models. After completing the runs for various machine learning algorithms along with NLP word embedding techniques, we have identified the best working one by comparing the confusion matrixes along with precision and recall scores of each model.

PROBLEM DEFINITION

Online harassment or cyberbullying have their roots from most of the online discussion forums which gives users the opportunity to express feeling of hate and direct personal attacks. Those threats, insults, profanity on online forums have been generally distinguished as toxic languages. Online spaces like Twitter, Wikipedia, and Google have termed the word toxicity for negative online interaction. However, recent study of toxicity has formalized it as “comments that are rude, disrespectful or otherwise likely to make someone leave a discussion” [1]. The increase in verbal abuse, disagreement, and harassment in social media platforms have prompted people to stop expressing their thoughts and opinions. As a result, it is a constant challenged to facilitate communications effectively and also being provoked to limit their service or also shut down user comments. In recent years, prevalence and consequence of online harassment or cyberbullying have provoked the research and industrial communities to create efficient computational systems which would identify and delete the toxic comments entered by various users [2]. And in order to develop such tool/service, we have initially obtained the sample dataset from Kaggle which was originally prepared by Jigsaw/Conversation AI team. After attaining the dataset, we performed data cleaning, data pre-processing and Word embedding which are various NLP techniques to eliminate data inconsistencies and create vectorized features of textual comments. Those features have been later assigned to various Machine learning models to predict whether the user comment is toxic or not. This model helps in detecting toxicities in the comments entered by various users [3]. In future, for healthy online conversation we will consider deploying this system which would distinguish toxic comments and eliminate them.

RELATED WORK

Toxicity detection of text in NLP is complex due to the absence of inflection and emotions. The modern developments in NLP research have seen increasing curiosities in evaluating the negative and positive sentiment of words or phrases. Nonetheless, efficiency and exactness of outcomes are usually influenced by false sensibilities of sarcasm and this flaw is often left unnoticed and not rectified.

In sentimental Analysis, Toxicity Detection is one of the most challenging problems across various languages. Chun-Che Peng *et al*. implemented a project which would distinguish sarcastic comments from user inputted texts with the help of are various machine learning classifiers such as Naive Bayes & Kernel Support Vector Machine. By using the Naive Bayes classifier with TF-IDF features the results obtained were inefficient. After a thorough analysis, the Naive Bayes approach did not consider contextual clues, feature types such as sentiment contrast, the order of words, etc. Both the models Naive Bayes and Kernel Support Vector Machine misclassify most of the sarcastic data. The role of the features is very essential for the system in order to perform successfully and enhance the quality of sarcasm detection [4].

Similarly, William Warner presented the paper on Hate Detection on the World Wide Web has also implemented various features and ML models such as POS and SVM respectively to predict whether the text is a hate speech or not. The prediction results of this project yielded a good accuracy score. However, the precision and recall scores were comparatively low. We will be studying and implementing the various features and models in our project to get the desired result and predictions [5].

Yin *et al.* proposed a research paper in 2009 which adopted supervised learning for harassment detection. In this paper, initially, sentiment and contextual features were given as inputs to Support Vector Machine algorithm for detecting repugnant and obscene comments in chat rooms and discussion forums. However, we observed that the approach of supplementing sentiment and contextual features to a TFIDF model returned better results [6]

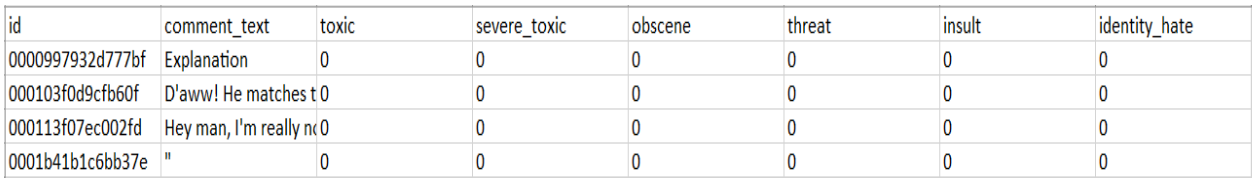
METHODOLOGY

Every experiment is performed by keeping a main goal in one's mind and our paper aims to detect the toxicity of a given user input text. To perform toxicity detection, we have initially gathered the dataset from external sources and followed by the implementation Data pre-processing to eliminate data discrepancies. Once the data has been cleaned and preprocessed, we have applied Word Embedding Techniques to it in order to create features which are to be fed to various selected machine learning algorithms to perform prediction on whether the given input text is toxic or not. We have divided the process of toxicity detection into various categories. These categories are as follows:

1. Collecting Datasets
2. Data Preprocessing (Data Cleaning)
3. Data Visualization
4. Feature Engineering
5. Implementation of Machine learning algorithm.

1. Collecting Datasets

The selection of a proper dataset is as much important as finding a correct algorithm to solve the problem. If the obtained dataset is not suitable for the chosen problem, then the outcome of the entire project tends to be less meaningful and eventually the model will become a failure. Therefore, it is very crucial to find the matching dataset for this problem, and in order to do so we have browsed through various websites such as Google Dataset Search, The Big Bad NLP Database, UCI Machine Learning Repository, Kaggle, etc. After reviewing all the websites and its datasets, we identified the most suitable and reliable dataset provided by the organization Jigsaw/Conversation AI team on Kaggle. Here is the sample overview of dataset acquired as shown in Figure 1.



id	comment_text	toxic	severe_toxic	obscene	threat	insult	identity_hate
0000997932d777bf	Explanation	0	0	0	0	0	0
000103f0d9cfb60f	D'aww! He matches t	0	0	0	0	0	0
000113f07ec002fd	Hey man, I'm really n	0	0	0	0	0	0
0001b41b1c6bb37e	"	0	0	0	0	0	0

Figure 1 Overview of Data

2. Data Preprocessing

When it comes to data science or machine learning solutions, Data cleaning and Data preprocessing are one of the first few steps to be dealt with. Data pre-processing is a data mining technique where crucial and cleaned information is fetched from the raw data from various different sources. The data obtained from various sources usually consists of various redundancies, inconsistencies and missing values. It is a vital step to fill the gaps in the dataset with the use of Data cleaning and Data preprocessing techniques. By eliminating those data discrepancies from the dataset, we have improved the quality of the data to produce optimized and accurate results with good scores [7].

In this paper, we have performed multiple processes of cleaning and preprocessing techniques to enhance the data quality. Firstly, we have removed columns from the dataset which are unnecessary for our analysis and followed by converting multiclass classification to a binary classification problem which was done by ignoring the column data with multiclass label data since the number of values for some of the target classes were minimal in comparison to the entire dataset.

Once the unnecessary columns are ignored, we have performed cleaning of data to remove any data inconsistencies. As part of Data cleaning we have performed the following on the comment text attribute,

- Removed the URLs from comment text.
- Removed trailing whitespaces, tabs and new lines
- Removed emoticons and special characters or non-ASCII characters
- Converted the words to lowercase and removed contractions

To remove URLs, whitespaces, emoticons and others, we used regular expression with Pandas library which gave faster and better result than normal iteration over every data row.

And as part of Data preprocessing, we have used NLP techniques such as Lemmatization and Stopwords removal. To perform these Data preprocessing techniques, we have used various modules of NLTK and Pandas library. NLTK in python is one of the most suitable platforms used for working with human language data. In addition to the in-built Stop word library, we have created a custom list of stop words manually that are meant to be removed from the data. But, in prior to Stop words removal, we have created another custom list of contraction words that are to be de-contracted. The removal of Stop words is done since some words in the text have no meaningful representation of the document and occupy unnecessary space in the database. At every instance of Stop word removal, we have also applied Lemmatization techniques that converts the inflected words to its root dictionary form in a document. For example, we have used WordNetLemmatizer module from NLTK to lemmatize the words in the comment text.

3. Data Visualization

Data visualization is the process of representing data in a graphical way and it is the integral part of any experiment to visualize and have a clear understanding of the dataset. It is also true that human brain adapts patterns more through visualization than simple columnar view. In our project, we have implemented various visualizations of data before cleaning, after cleaning and preprocessing dataset. Before cleaning the dataset, we have implemented a histogram to show on how the length of comment text was distributed over whole corpus.

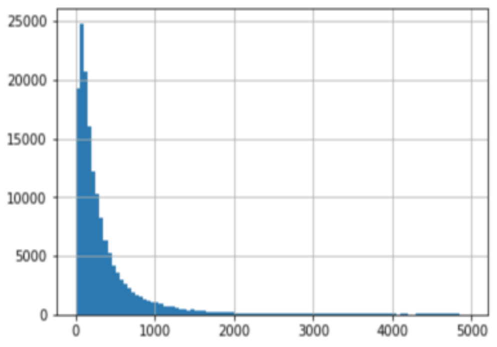


Figure 2 Comment length Distribution

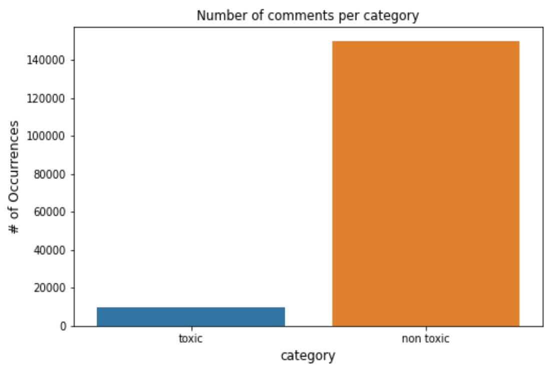


Figure 3 Number of comments per category

After visualizing the data, we were able to obtain some insights and understanding of how data is distributed among different categories and size of average length of individual comment. Similarly, after preprocessing of data through above mentioned processes we have obtained output as binary classified data which is shown in the above Figure 3 and we could clearly identify the dataset procured for this project is highly imbalanced with most of it belonging to the non-toxic class with 94% and very little belonging to the toxic class with 6%. This makes it difficult to predict the toxicity of the given input text. On the other hand, we have also provided visualization for the top 100 frequent occurring words in a bar chart and Word Cloud before and after data preprocessing.

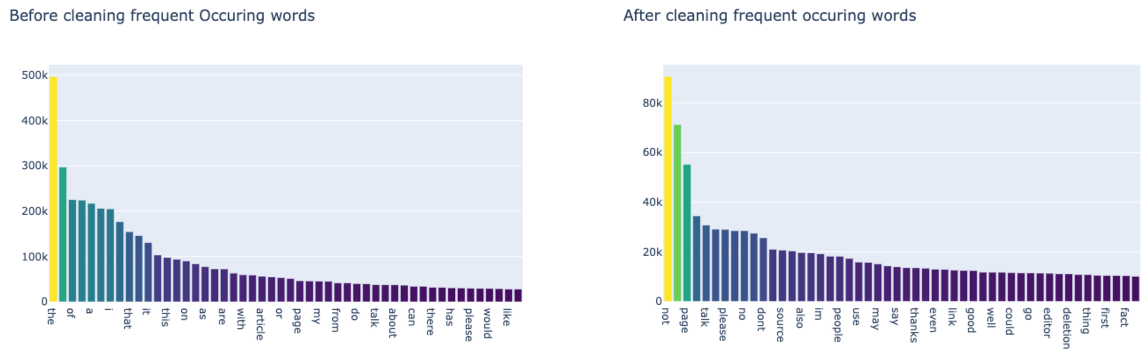


Figure 4 & 5 Frequent occurring words before and after cleaning & processing the data

Word Cloud

It is also a visualization of the most frequently occurring words. The weight of words in word cloud are based on the frequency of the words present in the dataset

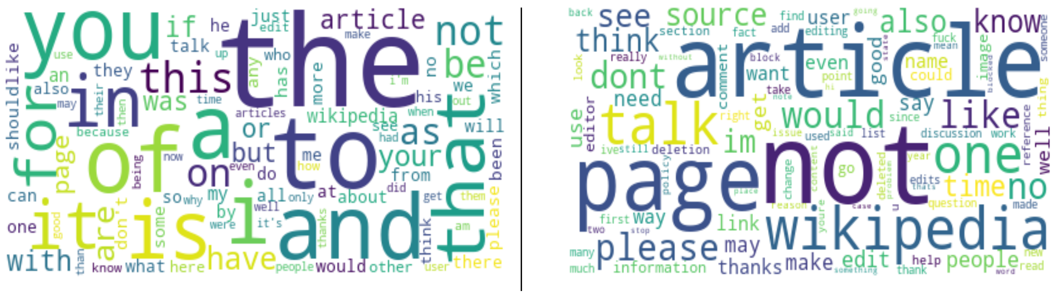


Figure 6 & 7, World Cloud for Frequent occurring words before and after cleaning & processing the data.

From Figures 4 & 6, we could see the most frequent occurring words are ‘the’, ‘of’, ‘an’ and so on. These words are considered to be Stop words. On the other hand, from figures 5 & 6, we are displaying the list of words that are to be treated for our analysis by removing Stop words and lemmatizing the remaining word to its root form.

We have implemented these data visualizations with the help of various modules present in the chart_studio, plotly, matplotlib & wordcloud. After preprocessing the data, we have assigned it to various Word Embedding Techniques in NLP for feature engineering.

4. Feature Engineering

Feature engineering is a process of getting insights or feature using domain specific knowledge from raw data and it is used to improve performance of machine learning models. We have applied Bag of Words (BoW), Term Frequency–Inverse Document Frequency (TF-IDF) techniques on our comment text attribute to create features. Bag of Words (BOW) is an NLP technique used to extract features from a given text documents [8]. This model represents a bag of words by keeping its multiplicity and ignoring the rest. This feature is commonly used in document classification and training machine learning algorithms [9]. By implementing BoW, we have created a dictionary of all the unique words occurring in the comment text of the training dataset [8].

The second feature engineering is performed with TF-IDF which is termed as Term Frequency-Inverse Document Frequency which is a statistical approach that is in a collection or corpus and is designed to indicate how essential a word is to a document. It is mostly used as a weighting factor in the field of text mining and information retrieval [10]. We have implemented the TF-IDF algorithm to transform the text data into numerical weighted representation. This technique is broadly implemented to derive features across various NLP applications

The last feature engineering is derived from SpaCy which is an open-source Python library that is mainly used in NLP applications similar like NLTK. It is implemented to provide understanding from unstructured data by performing statistical modeling and advanced deep learning approach by analyze large quantity of textual data [11]. Spacy has a set of inbuilt features such as tokenization, POS tagging, Similarity, text classification, etc. In Spacy, we have tokenized the complete string text into individual words and then created a vector for each of it. Later the created vectors we multiplied with IDF values to obtain the final features which are to be fed to various Machine learning algorithms.

To obtain the feature engineered data, we have divided the preprocessed dataset into train and test data. Then, the train data has been passed to CountVectorizer class of Scikit-learn's feature_extraction module for Bag of Words. In this, we provided the max features as 5000. We have also used bigram property in course of feature extraction process. Later, we have passed the data to transform and fit method of CountVectorizer which produce train feature engineered data. The Fit method used to learn vocabulary from the document provided which is the training data in our case, whereas, transform method convert document text to document matrix and return this matrix [12]. Similarly, we have followed the same procedure with TfidfVectorizer module as well.

As Figure 3, depicts that data is highly unbalanced so after feature engineering process, we used sampling methods to improve data quality and balance it. In our case, we have used the over-sampling method since the number of toxic comments in our dataset obtained was very less. To perform over sampling, we have used the "over_sampling" module of "imbalanced-learn" python library. We have adopted Synthetic Minority Over-sampling Technique (SMOTE) to oversample our data [13].

5. Implementation of Machine Learning algorithm

After creating the features i.e. vector representation of words. We will be feeding those features to the classification algorithms. However, there are many classification algorithms out there but choosing the best suitable one is important for our task. Based on the previous researches and analysis performed in Sentiment study, we have shortlisted the Machine learning algorithm.

Machine learning model is essential component which classifies the data according to their features. To obtain the best prediction results, we tried four classifier algorithms with three feature engineering methods to decide the most efficient one. Support Vector Machine (SVM), Random Forest, Naive Bayes and AdaBoost are the Machine learning model which were used to predict the result of our project.

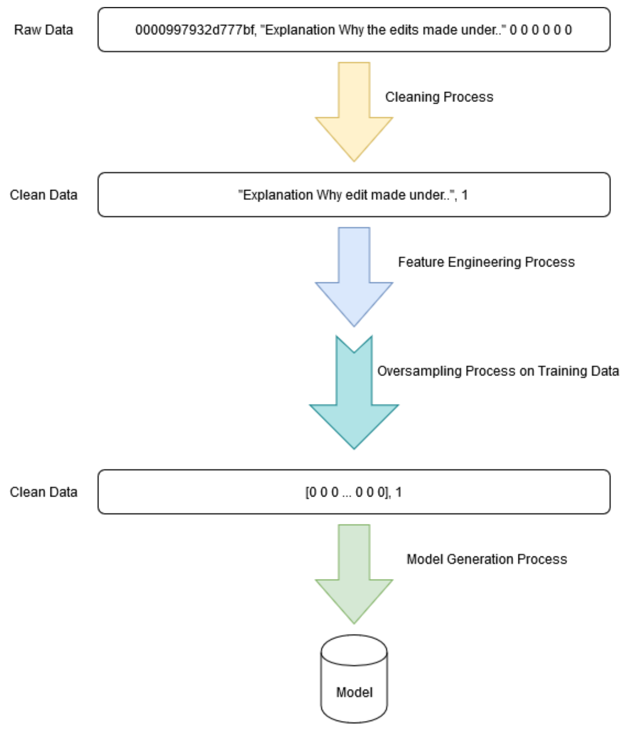
In order to choose the best parameters for our various machine learning models, we have used hyperparameter tuning with RandomizedSearchCV. The "fit & "score" are the two main methods of RandomizedSearchCV which are used to obtain the best parameters for training the models in our project. [14]. Support Vector Machine (SVM) is a discriminative algorithm where the output is an optimal hyperplane which distinguishes each class [15]. In this algorithm, SGDClassifier is used along with RandomizedSearchCV for Hyperparameter tuning. The alpha parameter for hyperparameter tuning is considered as a regularization parameter and also as an initial learning rate [16]. The other parameters used in this algorithm are "penalty" and "loss" which are set to default values such as 12 and hinge respectively.

The second algorithm, Random Forest exhibits ensemble learning methods which contains numerous individual decision trees i.e. each individual tree in the algorithm gives out a predicted class and the number of trees that has the highest number of votes for a particular class become the prediction of our model [17]. We have used RandomForestClassifier with RandomizedSearchCV for Hyperparameter tuning along with "n_estimators" and "max_depth" parameters to capture information and learn about the data.

Naïve Bayes classifiers are a collection of probabilistic classifiers with fixed independence assumptions among features. It works on the principle of Bayes theorem i.e. it illustrates the probability of an event based on the prior evidence [18]. We used the Multi-nominal Naïve Bayes classifier for classifying discrete features and performed hyperparameter tuning. The alpha parameter used for hyperparameter tuning is regarded as a smoothing parameter which is used to solve the problem of zero probability [19].

AdaBoost (Adaptive Boosting) is a boosting machine learning meta-algorithm which groups numerous weak classifiers into a strong classifier. It provides more weight on instances that are difficult to classify. Similarly, we have used hyperparameter tuning to get the best parameter. The parameters used in this algorithm are n_estimators, learning_rate and loss. The learning rate shrinks the contribution of each regressor [20].

We have implemented all the machine learning algorithms chosen for this project with the help of Python library SCIKIT-LEARN. The algorithms have been used with various combination of all features obtained from BoW, TF-IDF and SpaCy. For Spacy, we have adopted Inverse Document Frequency values from TF-IDF features along with SVM machine learning algorithm to obtain the prediction results for toxicity in a given text. The Figure 8 demonstrates the general overview of how model generating pipeline works.



Raw Data
0000997932d777bf, "Explanation Why the edits made under.." 0 0 0 0 0 0
Cleaning Process
Clean Data
"Explanation Why edit made under..", 1
Feature Engineering Process
Oversampling Process on Training Data
Clean Data
[0 0 0 ... 0 0 0], 1
Model Generation Process
Model

Figure 8 Model Generation Pipeline

EVALUATION

Classification problems deals data of different classes and to provide the optimal solutions, there are multiple Machine learning algorithms such as Logistic regression, K-nearest neighbour, Decision tree, Naive Bayes etc. Since the execution and prediction style of each and every algorithm is different, it is important and decisive to find the most efficient algorithm to solve a specific problem. Nevertheless, generating the model which can predict the right class is not sufficient enough and cannot be considered as the best performing model. In order to evaluate models, there are some evaluation methods which can help us to choose most suitable algorithm and therefore we have used Model Evaluation Metrics to evaluate the results from every model [21]. The Model Evaluation Metrics used in our project contains confusion, precision and recall matrix to evaluate the final results obtained from the model. While evaluating the accuracy of the classification on imbalance data can prove to be decisive, which is true in our case. Since the collected data was imbalanced, we will be evaluating all the matrix scores to decide the best performing algorithm among all. The confusion matrix will display the prediction result in four categories True Positive, False Positive, False Negative & True Negative respectively. The four sections represent the classes that are correctly and incorrectly classified.

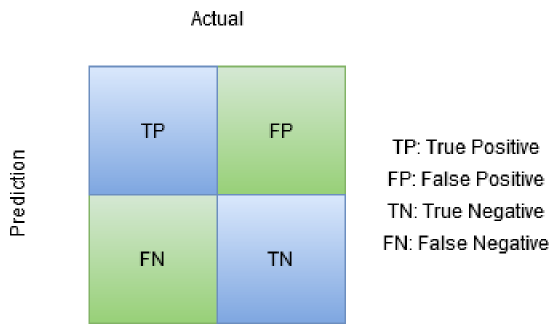


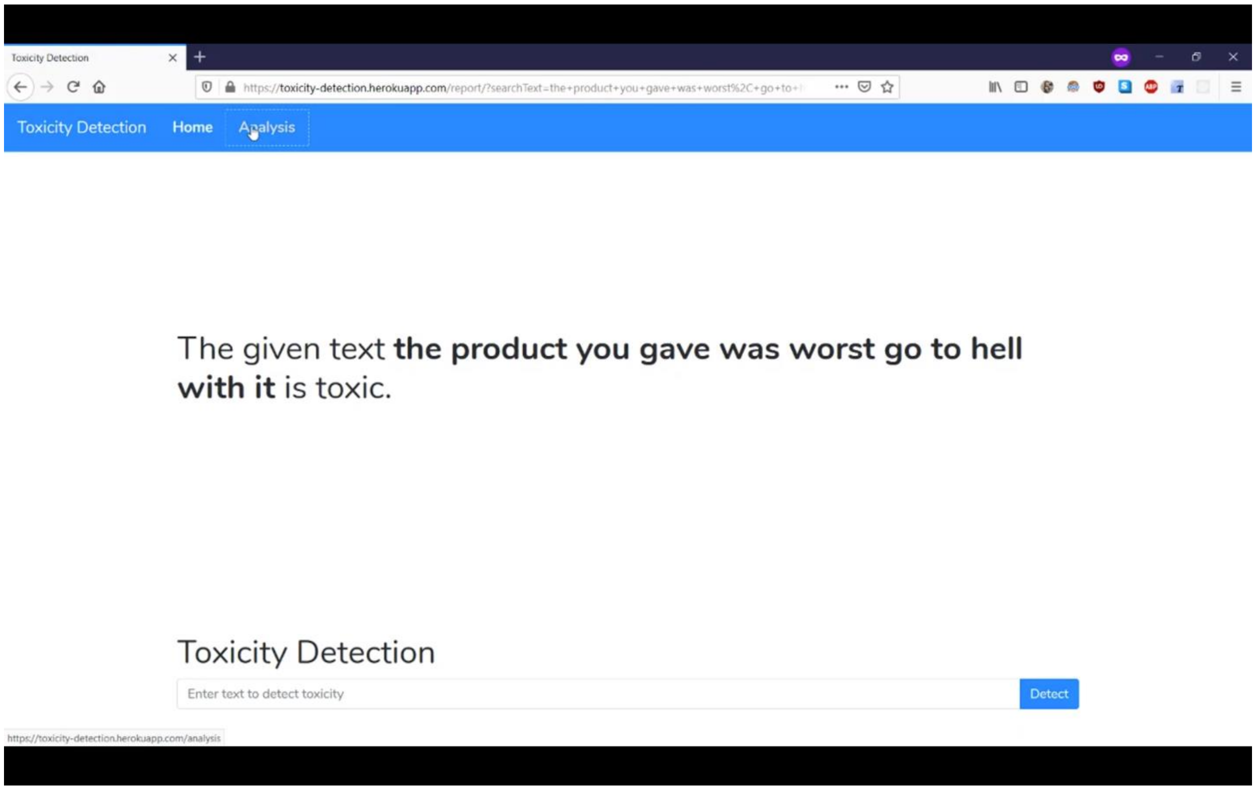
Figure 4 Confusion Matrix in General

Based on the evaluation metrics used, we could identify the best performing Machine learning algorithms based on the precision and recall scores. As we can see in the table below, we have chosen SVM and Naïve Bayes with TF-IDF feature engineered data has predicted the best results with good precision and recall scores. On the other hand, precision scores for SVM, AdaBoost and Random Forest are very low for Toxic Class. The model with the least accurate predictions were made by the SVM with SpaCy features which predicted all the datapoints belonging to Non-Toxic class. The evaluation matrices for all the models have been attached in the Appendix.

Algorithms	**Precision**		**Recall**	
	0 (Non-Toxic)	**1 (Toxic)**	**0 (Non-Toxic)**	**1 (Toxic)**
SVM with BoW	98%	26%	72%	87%
SVM with TF-IDF	**98%**	**52%**	**91%**	**83%**
Random Forest with BoW	95%	21%	71%	67%
Random Forest with TF-IDF	97%	30%	80%	76%
Naïve Bayes with BoW	96%	62%	96%	63%
Naïve Bayes with TF-IDF	**98%**	**50%**	**90%**	**84%**
AdaBoost with BoW	98%	14%	34%	93%
AdaBoost with TF-IDF	95%	79%	98%	52%
SVM with SpaCy	90%	-	100%	0%

Table 1 Comparison of various models results

To provide real-time example of the model, we have created the prototype of which has a user front-end UI that allows a user to enter a text in the textbox and submit it. Once the user has submitted it, the text data will be passed to Feature engineering and Machine learning module. Based on the model's performance, the text will be predicted as either 1 or 0. Then, the UI will retrieve the prediction results from the model stating whether the given text is toxic or not.


Toxicity Detection
https://toxicity-detection.herokuapp.com/report/?searchText=the+product+you+gave+was+worst%2C+go+to+
Toxicity Detection
Home
Analysis

The given text **the product you gave was worst go to hell with it** is toxic.

Toxicity Detection

Enter text to detect toxicity
Detect
https://toxicity-detection.herokuapp.com/analysis

CONCLUSION

Therefore, in this paper we have solved the binary classification problem of predicting whether a given user input text is toxic or not. In order to obtain a successful system, we have performed a combination of various tasks such as Data cleaning, Data preprocessing, Feature Engineering and application of Machine Learning models with hyperparameter tuning. Therefore, by observing the recorded results in confusion, precision and recall matrices, we have identified two best combination of word embedding technique and Machine learning algorithms such as SVM with IDF and Naïve Bayes with TFIDF. The two models consisting of precision and recall scores for toxic and non-toxic as.

SVM (TF-IDF):

Precision:	Toxic class: 52%	Non-Toxic: 98%
Recall:	Toxic class: 83%	Non-Toxic: 91%

Naïve Bayes (TD-IDF):

Precision:	Toxic class: 50%	Non-Toxic: 98%
Recall:	Toxic class: 84%	Non-Toxic: 90%

For Future scope, we have identified that we will be applying few concepts and rules of deep learning neural network.

REFERENCES

[1] S. K. B. Z. R. P. Hossein Hosseini, "Deceiving Google’s Perspective API Built for Detecting Toxic Comments," IEEE, Seattle, WA, 2017.

[2] L. D. O. R. R. R. N. T. Lora Aroyo, "Crowdsourcing Subjective Tasks: The Case Study of Understanding Toxicity in Online Discussions," in *Companion Proceedings of the 2019 World WideWeb Conference*, NewYork, USA, 2019.

[3] Kaggle, "www.kaggle.com," Kaggle, 2019. [Online]. Available: https://www.kaggle.com/c/jigsaw-unintended-bias-in-toxicity-classification. [Accessed 4 April 2020].

[4] M. L. J. W. P. Chun-Che Peng, "Detecting Sarcasm in Text: An Obvious Solution to a Trivial Problem," IEEE, 2015.

[5] W. Warner, "Detecting Hate on the World Wide Web," IEEE, New York, NY, 2018.

[6] Z. X. L. H. L. E. A. K. B. D. D. Dawei Yin, "Detection of Harassment on Web 2.0," IEEE.

[7] R. Miller, "ceoworld.biz," ceoworld.biz, 13 December 2019. [Online]. Available: https://ceoworld.biz/2019/12/13/data-preprocessing-what-is-it-and-why-is-important/. [Accessed 5 April 2020].

[8] P. Dubey, "freecodecamp.org," freecodecamp.org, 18 December 2018. [Online]. Available: https://www.freecodecamp.org/news/an-introduction-to-bag-of-words-and-how-to-code-it-in-python-for-nlp-282e87a9da04/. [Accessed 5 April 2020].

[9] K. J. M. M. H. S. M. D. B. L. B. Kamran Kowsari, "Text Classification Algorithms: A Survey," *MDPI,* p. 68, 2019.

[10] S. Lakshmanan, "medium.com," medium.com, 16 May 2019. [Online]. Available: https://medium.com/@swethalakshmanan14/how-when-and-why-should-you-normalize-standardize-rescale-your-data-3f083def38ff. [Accessed 5 April 2020].

[11] A. Lee, "towardsdatascience.com," towardsdatascience.com, 22 December 2019. [Online]. Available: https://towardsdatascience.com/so-whats-spacy-ad65aa1949e0. [Accessed 5 April 2020].

[12] Scikit-Learn, "CountVectorizer," [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html. [Accessed 30 Mar 2020].

[13] G. Lemaitre, F. Nogueira, D. Oliveira and C. Aridas, "SMOTE," [Online]. Available: https://imbalanced-learn.readthedocs.io/en/stable/generated/imblearn.over_sampling.SMOTE.html. [Accessed 31 Mar 2020].

[14] scikit-learn.org, "scikit-learn.org," scikit-learn.org, [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html. [Accessed 5 April 2020].

[15] S. Patel, "medium.com," medium.com, 3 May 2017. [Online]. Available: https://medium.com/machine-learning-101/chapter-2-svm-support-vector-machine-theory-f0812effc72. [Accessed 5 April 2020].

[16] V. Patlolla, "towardsdatascience.com," towardsdatascience.com, 28 November 2017. [Online]. Available: https://towardsdatascience.com/how-to-make-sgd-classifier-perform-as-well-as-logistic-regression-using-parfit-cc10bca2d3c4. [Accessed 5 April 2020].

[17] towardsdatascience.com, "towardsdatascience.com," towardsdatascience.com, 12 June 2019. [Online]. Available: https://towardsdatascience.com/understanding-random-forest-58381e0602d2. [Accessed 5 April 2020].

[18] A. Chavan, "medium.com," medium.com, 20 January 2019. [Online]. Available: https://medium.com/@akshayc123/naive-bayes-classifier-nb-7429a1bdb2c0. [Accessed 5 April 2020].

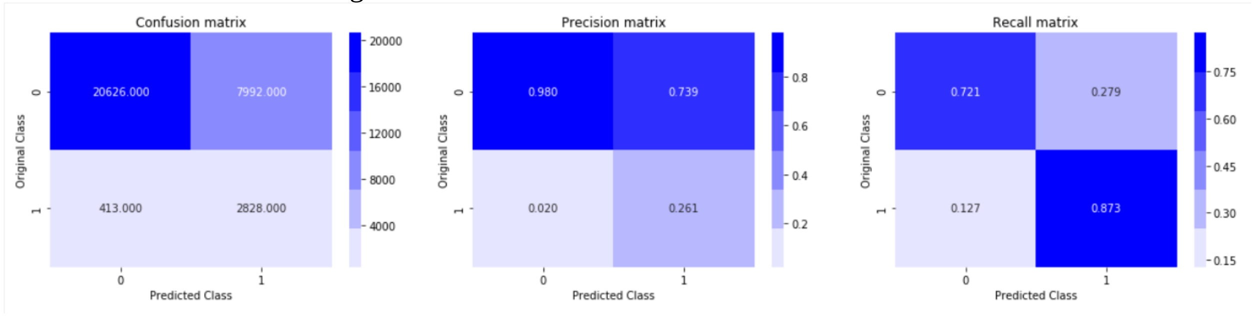
[19] Y. S, "towardsdatascience.com," towardsdatascience.com, 8 September 2019. [Online]. Available: https://towardsdatascience.com/introduction-to-na%C3%AFve-bayes-classifier-fa59e3e24aaf. [Accessed 5 April 2020].

[20] scikit-learn.org, "scikit-learn.org," scikit-learn.org, [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html. [Accessed 5 April 2020].

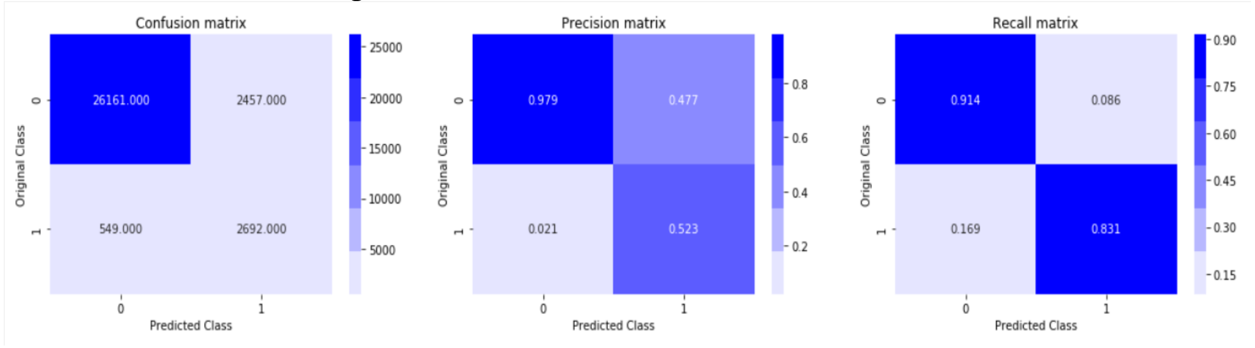
[21] L. VanCauwenberge, "www.datasciencecentral.com," www.datasciencecentral.com, 20 February 2016. [Online]. Available: https://www.datasciencecentral.com/profiles/blogs/7-important-model-evaluation-error-metrics-everyone-should-know. [Accessed 5 April 2020].

APPENDIX

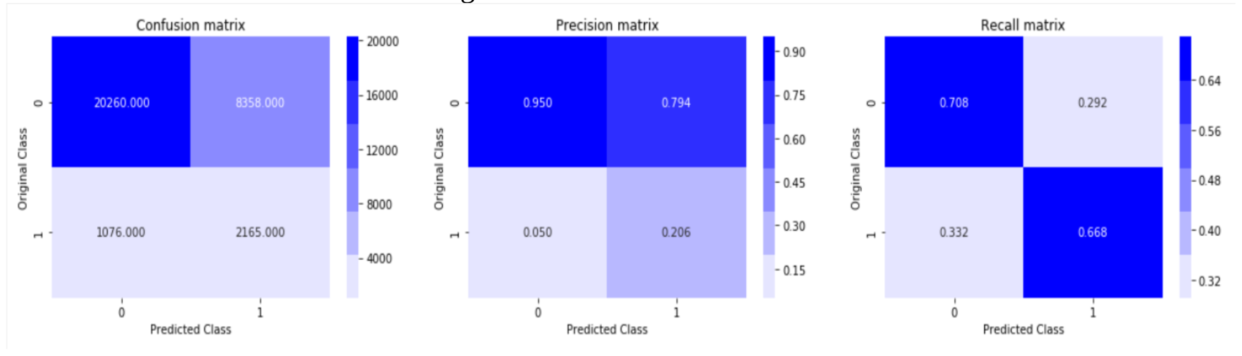
Evaluation Matrix of SVM algorithm with BoW



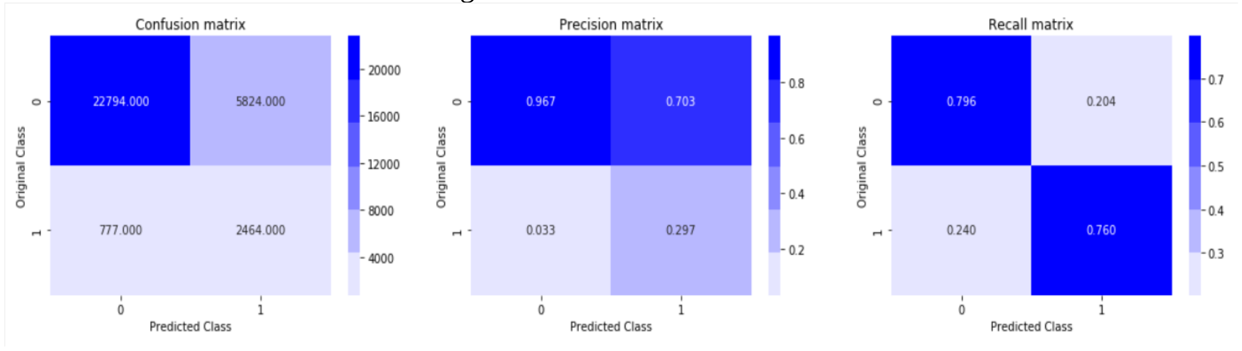
Evaluation Matrix of SVM algorithm with TF-IDF



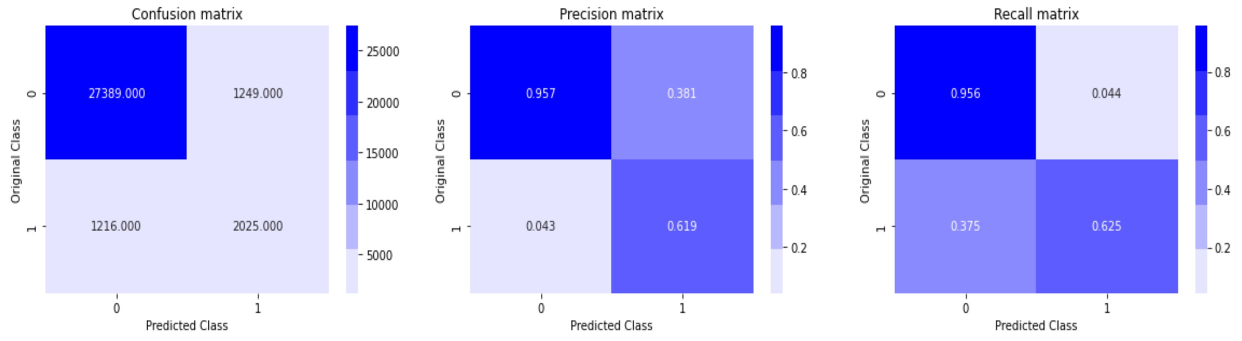
Evaluation Matrix of Random Forest algorithm with BoW



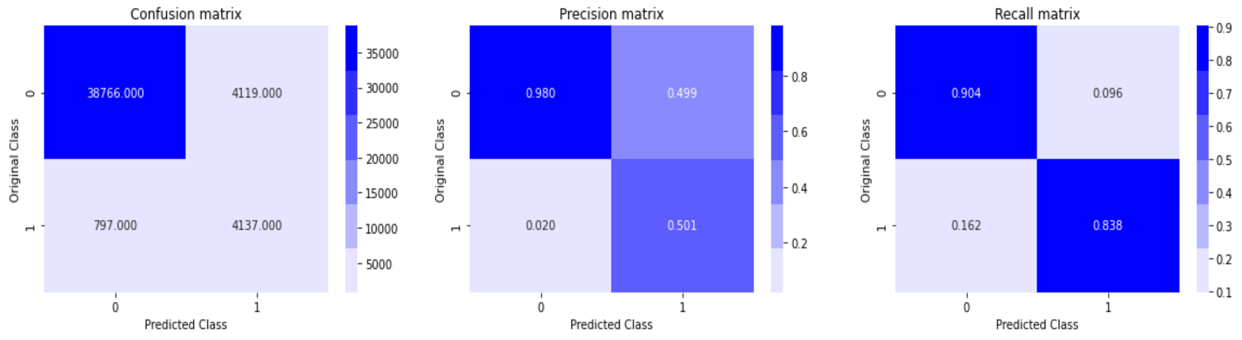
Evaluation Matrix of Random Forest algorithm with TF-IDF



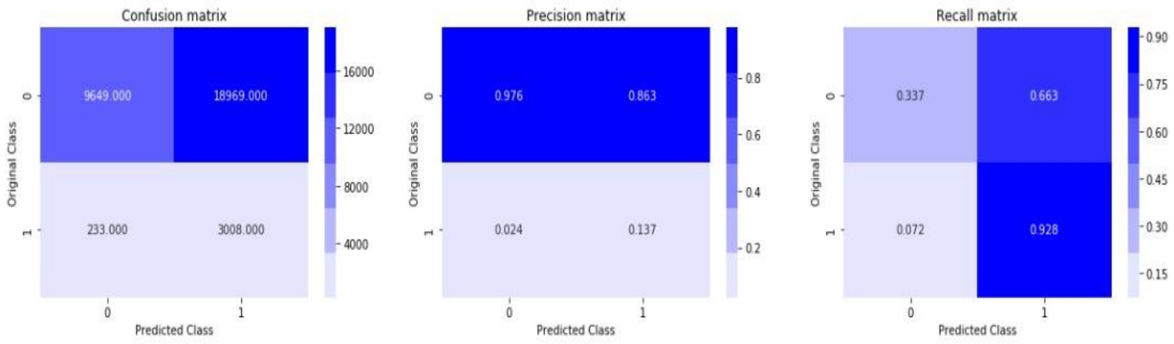
Evaluation Matrix of Naïve Bayes algorithm with BoW



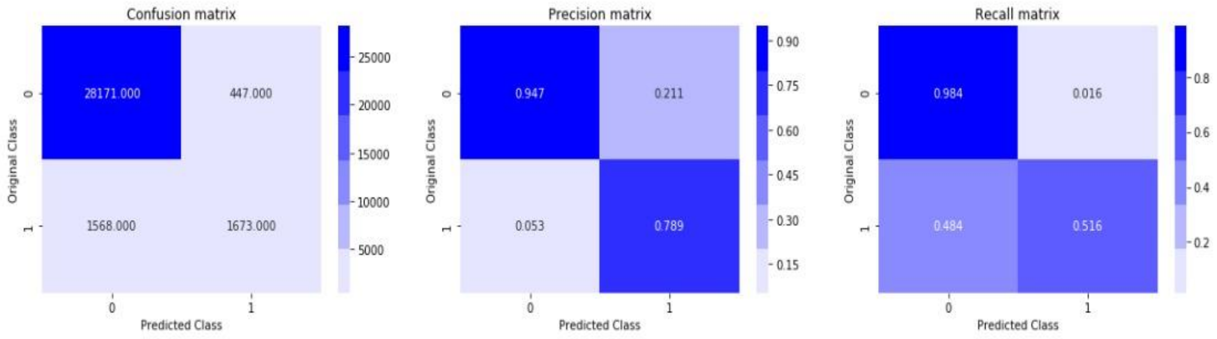
Evaluation Matrix of Naïve Bayes algorithm with TF-IDF



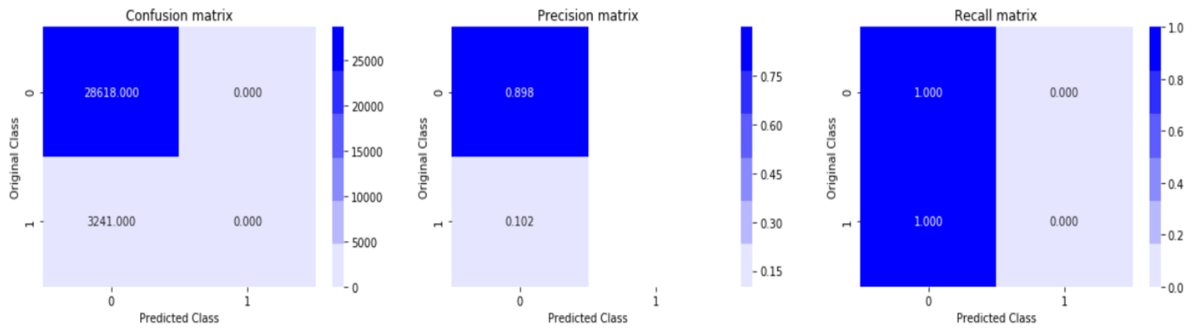
Evaluation Matrix of AdaBoost algorithm with BoW



Evaluation Matrix of AdaBoost algorithm with TF-IDF



Evaluation Matrix of SVM algorithm with TF-IDF and SpaCy


Confusion matrix
Original Class
Predicted Class
Precision matrix
Recall matrix