Toxic Comments Classification

Faranak Abri, Rakshit Gupta, Prathamesh Dixit, Nitish Ranjan
Computer Science Department, San Jose State University
{*faranak.abri, rakshit.gupta, prathamesh.dixit, nitish.ranjan*}*@sjsu.edu*

***Abstract*—Social media has become an essential part of our lives these days. People are interacting with one another more than ever. Not only are these social networking websites a medium for communication, but they have also become a hub for news circulation, education, and also e-commerce. With more and more users adding up, there is a rise in the number of people writing inappropriate content. This project aims to identify and flag all such comments posted on the internet and classify them into the 6 classes of toxicity - toxic, severe toxic, obscene, threat, insult, and identity hate. In this project, various machine-learning models have been deployed using Natural Language Processing (NLP) techniques. Further, the performance of these models is analyzed using the F1-score as the evaluation metric.**

***Index Terms*—Natural Language Processing, Normalization, Data Visualisation, TF-IDF**

[1]

I. Introduction

The Internet's ability for individuals to exchange ideas is built on platforms that aggregate user information. Websites such as Wikipedia, blogging sites, communities, and internet forums are prime examples. The worry remains that not everybody who uses the Internet wants to interact courteously; some utilize it as a way to vent their wrath, insecurities, and bigotry. The issue is that people routinely write inappropriate content. In order to maintain a positive community, both the toxic content and the users posting it need to be removed quickly. However, businesses lack the funds to employ full-time moderators to scrutinize each comment. So, in order to automate this process, machine learning (ML) can be employed in building models with top-performing methods and parameters by investigating the efficacy of various Natural Language Processing (NLP) strategies [1].

Wikipedia relies on user debate to curate and approve material because it is entirely user created. The issue with this is that people routinely publish inappropriate things, and in order to maintain a pleasant community, both the users who post the toxic content and it must be immediately deleted. Due to this issue, a sizable open dataset of labeled Wikipedia Talk Page comments was created. This dataset will be used for the project. Through Kaggle, you can access the dataset. The project will concentrate on a seventh label that indicates the overall toxicity of the remarks out of the six labels in the dataset that represent subcategories of toxicity.

Support Vector Machines are supervised learning models with associated learning algorithms that analyze data for classification and regression analysis [1].

[1]https://www.overleaf.com/read/dbknkkydxtwq

Machine learning techniques are often used to analyze and predict the toxicity of text. In this paper, three methods of machine learning were used to classify the sentiment of the reviews. They are - Logistic Regression, SVMs and Multinomial Naïve Bayes. The accuracy of these methods has been studied to evaluate their performance.

The remainder of this report is organized as follows: the related work is reviewed in Section II, the dataset is described in Section III, while the preprocessing techniques implemented are listed in Section IV. The prediction models are detailed in V. Section VI provides a comprehensive comparison between the models implemented. The last 3 sections, Sections VII, VIII, and IX illustrate the computational resources used, the Graphical User Interface built, and the conclusion of this project.

II. Related Work

Online abuse is prevalent around the clock which can be done anonymously, and one single incident can spread like wildfire, as suggested in [2]. This problem led the Conversation AI team, owned by Alphabet, to develop a large open dataset of labeled Wikipedia Talk Page comments, which will be the dataset used for the project. Over the past several years, there has been more data and content published online, and not paying attention to the increasing amounts of inappropriate content will undoubtedly lead to a toxic internet culture. The most crucial task is sorting texts into positive and negative categories and examining the polarity of the negative evaluation. We use interdisciplinary approaches from natural language processing and machine learning to effectively identify, understand, and communicate extracted opinions and observed polarity [2]. The ambiguity, constant change, and poorly defined characteristics of human language make NLP, a branch of AI concerned with giving computers the ability to understand text and spoken words highly difficult field to work in. Based on ML and NLP techniques proposed in [2], [3], autonomous cyberbullying content identification system needs can be built. It will serve as a way to decrease and end cyberbullying. It can be done in several ways, like identifying insult-related terms with the Multiple Correlation Coefficient (MCC) algorithm. In reality, even with ML and NLP, existing insult detection methods have relatively low recall rates. A number of methods including methods like Light BGM have been employed thus far as we see in [4] but the results haven't been up to the mark. Evolutionary prototyping, incremental prototyping, throwaway prototyping, and extreme prototyping are the major prototyping types now in use. Extreme prototyping, on the other hand, is primarily used for web-based applications that divide web development into three phases that are comparable to the three levels of prototyping: low, mid, and high-fidelity models as inferred from [3]. Effectiveness and an aesthetically pleasing feel are lacking in the current systems. In order to make the system easier for the user to operate, Graphical User Interface (GUI) can be incorporated [2]. Applying the same idea to all social media and other applications work is another improvement that needs a lot of work [2]. Instead of deploying the entire program to the cloud server, improvements can be made to link the application to it [4]. The prediction also has to be improved. [4] For this, the dataset should be thoroughly divided with properly defined labels. Since it is an NLP problem, vectorization of the data and testing of multiple classification algorithms is needed [3].

III. Dataset Description

In this prediction experiment, we used the Toxic Comment Classification dataset obtained from Kaggle [5].

This dataset contains 159,571 comments synthesized from Wikipedia. The data consists of one input feature, the string data for the comments, and six labels for different categories of toxic comments: toxic, severe toxic, obscene, threat, insult, and identity hate. The term ”toxic” is not a blanket description but rather a subclass with a lot of overlap. As a result, we add a seventh label named ”any label” to describe a comment’s overall toxicity. Any comments that have been marked as toxic will be referred to as such, with the word ”toxic” itself and any additional labels appearing in quotation marks.

A. Data Visualization

The dataset is highly imbalanced as is evident from figure 1 that the number of comments belonging to the classes ’severe_toxic’, ’identity_hate’, and ’threat’ is very low compared to that of the other 3 classes. We can see the data composition to be:

- Toxic: 9.58%
- Severe_toxic: 1%
- Obscene: 5.29%
- Threat: 0.3%
- Insult: 4.94%
- Identity_hate: 0.88%

A notable point to be observed is that the ”Toxic” label is not a catch-all label but has a large amount of overlap. For e.g., the ”severe_toxic” label is not a sub-category of the “toxic” label. This means that a comment can be labeled as ”severe_toxic” without being labeled ”toxic”. This kind of overlap is found in the entire dataset.

B. Python Libraries Used

The following python libraries have been used for the visualization of datasets:

- MatplotLib
- Seaborn

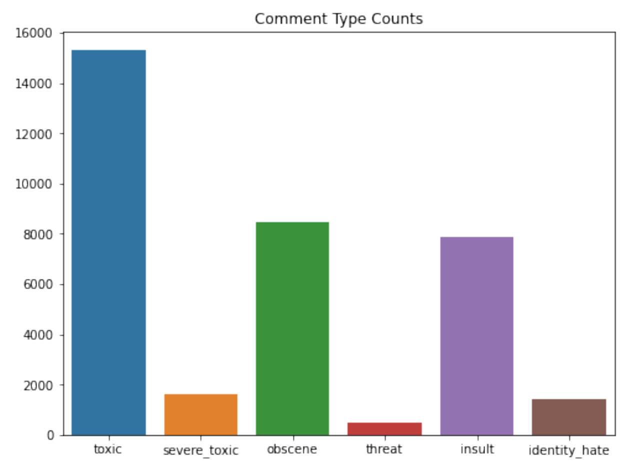


Fig. 1: Data-distribution of the Wikipedia dataset

IV. Preprocessing

A. Background

To perform pre-processing on the data, i.e., to transform data in a way that the features have similar distributions, the data has been normalized. We have used the TfidfVectorizer of the sklearn module to vectorize the data.

- Removal of Punctuations- Punctuation marks used at the start and end of the comments are removed along with additional white spaces.
- Word Tokenization- Each individual comment is tokenized into words and stored in a list for easier retrieval.
- Removal of stop words- Affixes are removed from the stem.
- Strip IP Addresses- The IP Address from each comment is removed before training the models.

B. Feature Engineering

Additional linguistic features were added to the dataset to improve the performance of ML models. Below is the description of each feature.

- Comment length - Clean comments on average are longer than toxic comments.
- Capitalization percentage – Toxic comments are more likely to have caps in them.
- The mean Word length – Toxic comments on average use more abbreviations such as ”u” instead of ”you”.
- The average number of exclamation points – Toxic comments use more exclamation marks - ”!”.
- The average number of question marks – Toxic comments have more frequent question marks - ”?”.
- any_label - This is a new ”catch-all” target variable, created by taking a logical OR of the original target variables. It has been added for hyper-parameter tuning and evaluating the overall performance of models.

After creating the new features, these are then normalized using the MinMaxScaler function of sklearn.preprocessing module.

C. Python Libraries Used

The following python libraries have been used in the pre-processing, modeling, and evaluation of the dataset:

- Numpy - Used for array computations
- Pandas - Used to read data from the dataset and for operations like constructing the dataframe and concatenating dataframes.
- sklearn - A vast library used for classification reports, confusion matrices, and splitting datasets into train and test sets
- NLTK - Library to perform tokenization
- pickle - Library to save models
- sparse - To efficiently store 2-Dimensional matrices

V. Prediction Models

We have used the following models to classify the comments under the six classes of toxicity

We split the dataset into 3 parts – a Training Set, a validation set, and a Test Set in the ratio of 60:20:20. We analyzed the comments with 3 models – Logistic Regression, Support Vector Machines, and multinomial Naïve Bayes. The validation set is used for tuning the hyper-parameters of the models and TF-IDF Vectorizer, which are then later tested using the test set. Upon training the models, it was found that there was hardly any overfitting of data. Hence we did not need to perform feature selection on the dataset.

A. Evaluation Metrics

We have displayed a Confusion Matrix. A Confusion Matrix, also known as an error matrix, is a specific table layout that allows visualization of the performance of an algorithm. Each row of the matrix represents the instances in an actual class while each column represents the instances in a predicted class, or vice versa – both variants are found in the literature. The name stems from the fact that it makes it easy to see whether the system is confusing two classes. In our case, since the data is highly imbalanced, and one label represents less than a third of a percent of the data, it's too easy to get a high score even with hardly any true-positive predictions. Hence, F1 Score is used for evaluating the performance of models. It severely penalizes models that just predict everything as either positive or negative with an imbalanced dataset. The F1 score can be interpreted as a harmonic mean of precision and recall, where an F1 score reaches its best value at 1 and the worst score at 0. The relative contribution of precision and recall to the F1 score are equal. The below formula explains how the F1 score can be calculated for each target variable class.

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} \tag{1}$$

B. Model Tuning

The models are tuned to perform best on the given data using different combinations of the n-grams, max_features for the TfidfVectorizer. The performance of each n-gram combination is evaluated using the validation set and a baseline model.

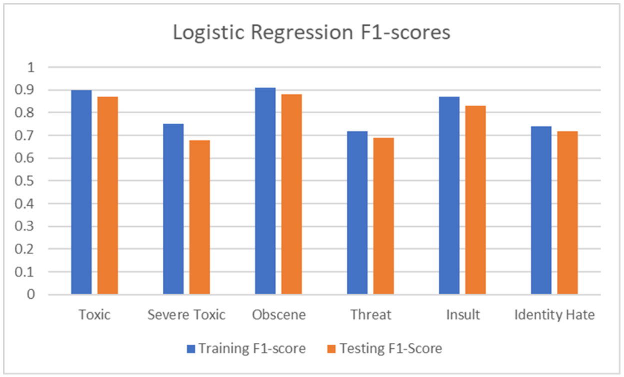


Fig. 2: F1 scores for Logistic Regression

Vectorizer with 20,000 unigram and bigram word features combined with 10,000 tri, quad, and pent-grams gave the best results. The F1 scores for each combination tried can be viewed in Table I. After selecting the optimum Vectorization parameters, each individual model is also tuned for the best performance using the 5-fold cross-validation technique.

Word N-gram	**Word Max Features**	**Char N-gram**	**Char Max Features**	**F1 Score**
1,2	30,000	-	-	0.63
1	20,000	-	-	0.64
1,2	20,000	-	-	0.65
1,2	20,000	3,4,5	10,000	0.71

TABLE I: F1 scores of N-grams

The best performance of each model was delivered when using the below hyper-parameters.

- Linear Regression:
  - C (l2 penalization factor): 1.2
- SVM:
  - kernel: linear
  - C (l2 penalization factor): 0.5
- Naive Bayes:
  - Aplha (Laplace Smoothing): 1.0

C. Logistic Regression

The logistic model is a statistical model that models the probability of an event taking place by having the log odds for the event be a linear combination of one or more independent variables. The algorithm has been used to train different multiple models for each toxicity class and their F1 scores can be seen in Figure 2. The overall performance of the model can be analyzed using the confusion matrix of "any_label" target class, displayed in Figure 3.

D. Support Vector Machine

SVM is a supervised learning method that looks at data and sorts it into one of the multiple classes. An SVM outputs a map of the sorted data with the margins between them as

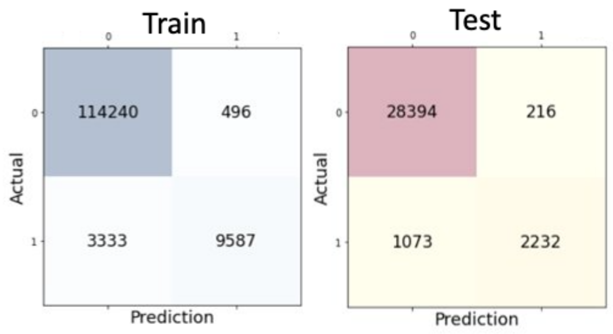


Fig. 3: Confusion Matricesfor 'any_label': Logistic Regression

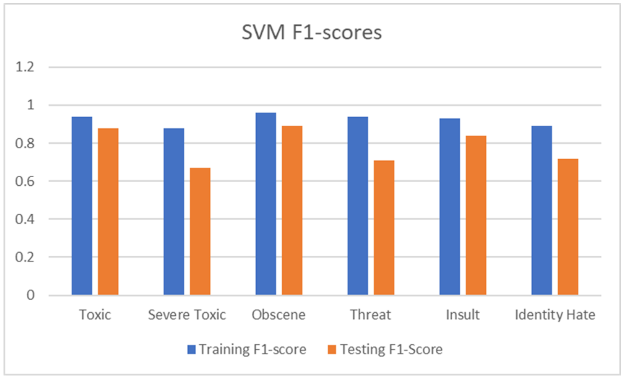


Fig. 4: F1 scores for Support Vector Machine

far apart as possible. The algorithm has been used to train binary classifiers for each of the target variables. The F1 scores of each class can be seen in Figure 4 and their overall performance can be analyzed using the confusion matrix in Figure 5.

E. Multinomial Naïve Bayes

A Naive Bayes classifier is a probabilistic machine learning model that's used for the classification task. The crux of the classifier is based on the Bayes theorem.

$$P(A|B) = \frac{P(A) * P(B|A)}{P(B)} \tag{2}$$

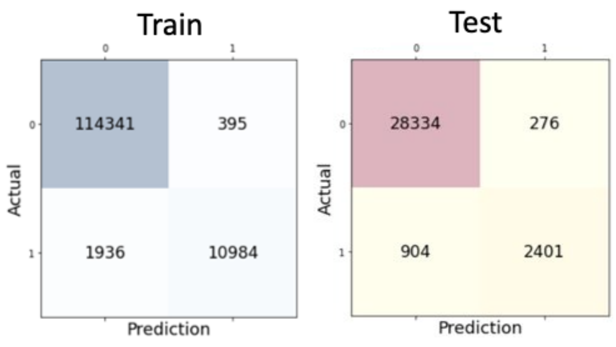


Fig. 5: Confusion Matrices for 'any_label': Support Vector Machine

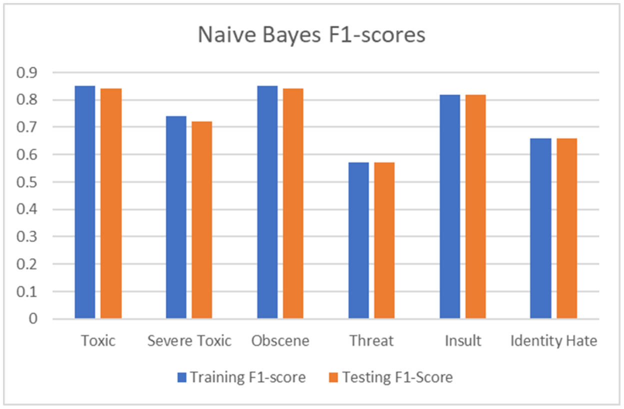


Fig. 6: F1 scores for Multinomial Naïve Bayes

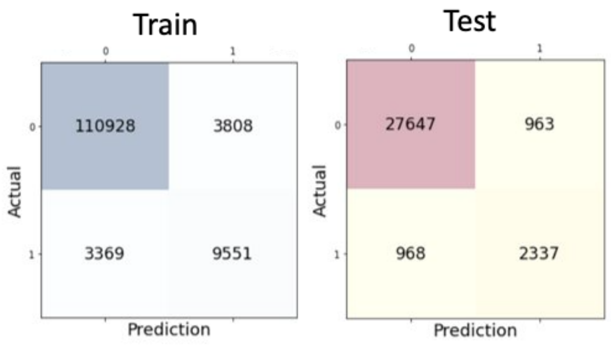


Fig. 7: Confusion Matrices for 'any_label': Naïve Bayes

The algorithm has been used to train binary classifiers for each of the target variables. The F1 scores of each class can be seen in Figure 6 and their overall performance can be analyzed using the confusion matrix in Figure 7

VI. Comparison and Analysis

The comparison of various models is illustrated in Figure 8 and Tables III and II.

For the dataset, the highest F1 score is observed with the Support Vector Machine model. As is evident from the F1 score comparison analysis in Figure 8, the best-performing model was the Support Vector Machine model. F1 scores are 0.95 and 0.89 for the "any_label" training and testing sets respectively. A similar trend was observed across all labels in the datasets.

The Linear Regression model performed close to the Support Vector Machine model. Since the Support Vector Machine model is tuned for linearity, the similarity in performance is anticipated. The F1 scores are 0.91 and 0.88 for the "any_label" training and testing sets respectively.

The Multinomial Naïve Bayes model performed well for half the labels as is evident from Tables III and II. However, it struggles with 'Severe Toxic", "Threat", and "Identity Hate" labels. These classes are a minority in the dataset as observed in Figure 1.

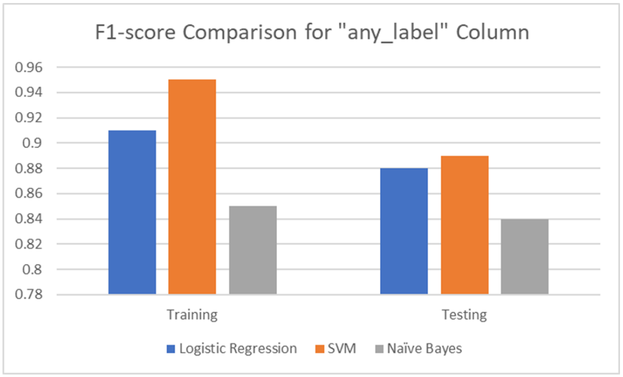


Fig. 8: F-1 score comparisons

Label	**Logistic Regression**	**SVM**	**Naïve Bayes**
Toxic	0.9	0.94	0.85
Severe Toxic	0.75	0.88	0.74
Obscene	0.91	0.96	0.85
Threat	0.72	0.94	0.57
Insult	0.87	0.93	0.82
Identity Hate	0.74	0.89	0.66

TABLE II: Training F1 scores of Multiple Classes

Label	**Logistic Regression**	**SVM**	**Naïve Bayes**
Toxic	0.87	0.88	0.84
Severe Toxic	0.68	0.67	0.72
Obscene	0.88	0.89	0.84
Threat	0.69	0.71	0.57
Insult	0.83	0.84	0.82
Identity Hate	0.72	0.72	0.66

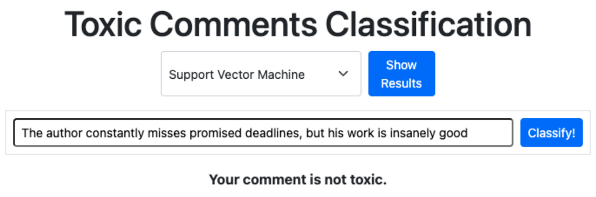
TABLE III: Testing F1 scores of Multiple Classes

VII. Computational Resources Used

Our project has been executed using Google Colab, a free cloud service that offers Jupyter notebooks via remote servers. Google Colab (Free version) offers 12GB RAM and a hard disk capacity of 108GB with Python 3 installed on it.

VIII. Graphical User Interface

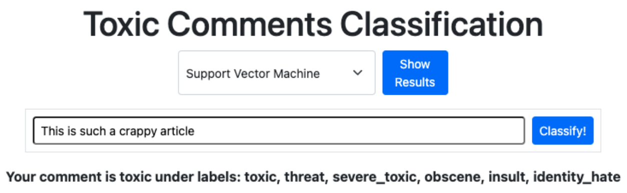
A Graphical User Interface (GUI) has been built upon the trained models using the pyscript python library, HTML, CSS, and JavaScript. The GUI enables users to fetch the evaluation results for each model trained in the project. It also enables users to view the classification of custom comments in real-time. These functionalities are depicted in Figures 9 and 10.



The author constantly misses promised deadlines, but his work is insanely good

Your comment is not toxic.

Fig. 9: GUI: Non-Toxic Comment



This is such a crappy article

Your comment is toxic under labels: toxic, threat, severe_toxic, obscene, insult, identity_hate

Fig. 10: GUI: Toxic Comment

IX. Conclusion

The final model selection is based on the best model after calculating the accuracy and the F1 scores for Logistic Regression, SVM, and Naive Bayes models, and testing for underfitting and overfitting. The confusion matrix allowed us to calculate the accuracies and the F1 scores.

For the dataset, the SVM model with a linear kernel showed the best F1 scores. The model shows an impressive accuracy of 96%. Hence, this model can be used to identify toxic comments. From Figure 8, it can also be concluded that the linear models perform better for this task than the more complex non-linear models such Naive Bayes.

In the future, given that we have more computational power, we would like to compare our performance with neural network models such as RNNs or pre-trained models such as BERT.

References

[1] J. Su, V. Vysotska, A. Sachenko, V. Lytvyn, and Y. Burov, "Information resources processing using linguistic analysis of textual content," in *2017 9th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*, vol. 2, 2017, pp. 573–578.

[2] A. A. Putri Ratna, A. Kaltsum, L. Santiar, H. Khairunissa, I. Ibrahim, and P. D. Purnamasari, "Term frequency-inverse document frequency answer categorization with support vector machine on automatic short essay grading system with latent semantic analysis for japanese language," in *2019 International Conference on Electrical Engineering and Computer Science (ICECOS)*, 2019, pp. 293–298.

[3] K. Dubey, R. Nair, M. U. Khan, and P. S. Shaikh, "Toxic comment detection using lstm," in *2020 Third International Conference on Advances in Electronics, Computers and Communications (ICAECC)*, 2020, pp. 1–8.

[4] M. Vichare, S. Thorat, C. S. Uberoi, S. Khedekar, and S. Jaikar, "Toxic comment analysis for online learning," in *2021 2nd International Conference on Advances in Computing, Communication, Embedded and Secure Systems (ACCESS)*, 2021, pp. 130–135.

[5] "Toxic comment classification challenge." [Online]. Available: https://www.kaggle.com/competitions/jigsaw-toxic-comment-classification-challenge/data