CE807-24-SP – Assignment Report

Student ID: 2310246 Email: md23681@essex.ac.uk

Abstract

Online platforms should censor or delete toxic speech to fulfill legal compliance and keep users happy. In this assignment, we deal with the problem of automatic extraction of negative comments online forums automatically by text classification. A dataset was given to us as well, and we looked into using both generative and discriminative methods to determine which machine learning models were most effective.

Our task would be creating, testing, and scoring such text classifiers so as to exactly determine whether a piece of text is harmful or not. Our studies equally focus on the implementation of both construction-modeling and choice-justification, in order to offer a fundamental understanding of the classification problem. The accuracy and the consistency of our results are assured through the dataset which is provided to us for the purposes of training, validation, and testing throughout the assignment.

By empirical comparison we show the working of our models in detecting toxic comments, spotlighting their effectiveness and constraints. We speak about how model selection, feature engineering, and hyperparameter tuning contribute to the overall improvement in classifier performance.

Overall, the assignment gives heaps of practical knowledge on the subject and constitutes a good process of the artificial intelligent toxic comment detection directed at the safety of the online community.

Materials

Below are the links to get access to the following materials:

- Zoom Presentation
- Google Colab
- Google Drive

1 Task 1: Model Selection

The models that I have used under Generative and Discriminative methods for text classification are:

- Generative method - Naive Bayes
- Discriminative method - Logistic Regression, Support Vector Machine (SVM)

1.1 Summary of 2 selected Models

"Generative classifiers learn a model of the joint probability, p(x,y), of the inputs x and the label y, and make their predictions by using Bayes rules to calculate p(y|x) and then picking the most likely label y. Discriminative classifiers model the posterior p(y|x) directly, or learn a direct map from inputs x to the class labels" [3].

Naive Bayes, Logistic Regression, and Support Vector Machines are widely used models for toxic comment classification tasks, each offering unique strengths in terms of simplicity, interpretability, and performance. Their application in identification of toxic comments in online platforms contributes to enhancing user experience, complying with legal requirements, and fostering a safer online environment.

1.2 Critical discussion and justification of model selection

For an algorithm that needs to run quickly each time a remark is placed on one of the most well-known websites on the Internet, deep learning techniques might be too resource-intensive. Efficiency would be a crucial necessity if this were a real-world business issue. Furthermore, stacking would be necessary to modify the model to accommodate supplementary information; this is an untidy workaround that makes training and prediction more difficult.[4]

Transparency is another factor to be taken into account, and it is the primary reason why neural networks were not chosen for this application. To make sure the model is not picking up bias related to race, gender, sexual orientation, culture, other unanticipated categories from the data curators, I want to be able to readily audit it. A third party will find it much simpler to examine the effects of particular features on the model and make the necessary corrections to mitigate bias by using Naive Bayes, Logistic Regression, and SVM.[4]

The notable disparity in vocabulary frequency distributions between comments suggests that Naive Bayes might perform well under generative method. Under discriminative method, SVM might outperform Logistic Regression in my opinion.[4]

Support vector machine models are distinct in that they are not impacted by outliers and determine the boundaries between classes by measuring the distances between each class's nearest point and the dividing line. I used the linear kernel here. The most effective kernel among them is the linear kernel, which also has a high resistance to overfitting. In support vector machines, the kernel trick is to project data with nonlinear or complicated boundaries into a higher dimension where they can be separated linearly, and then use a linear model algorithm to build a hyperplane between the data points. [4]

2 Task 2: Design and implementation of classifiers

Let's discuss the complete design and implementation of the classifiers which includes:

- Dataset details with table
- Model implementation details like preprocessing, hyperparameters etc

2.1 Dataset details

The dataset contains 3 files: train set, validation set, and test set. Train set is used for training the models, validation set for evaluating the performance of the models, and test set

to produce model output.

The train set has 4 columns: comment id, comment, split, and toxicity; and 8699 rows. Our focus is on the comment and toxicity columns. Comment column contains toxic and non toxic comments. Toxicity column has 2 classes: 'toxic' class labelled as 1 and 'non toxic' class labelled as 0. So, basically we are performing binary classification, trying to classify a comment either as toxic or non toxic. The validation set also has same 4 columns as train set with 2920 rows. The test set has no toxicity column. We will train our generative and discriminative models and generate the toxicity labels under 'out label model Gen' and 'out label model Dis'.

The shape of the dataset is as follows:

- Train set: (8699, 4)
- Validation set: (2920, 4)
- Test set: (2896, 5)

Dataset	Total	% Toxic	% Non Toxic
Train	8699	13.00	87.00
Valid	2920	13.15	86.85
Test	2896	xxx	xxx

Table 1: Dataset Details

The distribution of Toxicity in Percentage in train set shows 87 percent of data labelled as 0 (non toxic) while only 13 percent of data labeled as 1 (toxic). This means that the amount of toxic comments is very less which means there is a high imbalance of classes.

The distribution of Toxicity in percentage in validation set shows 86.85 percent of data labelled as 0 (non toxic) while only 13.15 percent of data labeled as 1 (toxic). This means that the amount of toxic comment is very less indicating high imbalance of classes in validation set as well.

Some undersampling or oversampling should be performed to remove imbalance.

2.2 Data preprocessing

Data preprocessing is the process of cleaning, transforming, and preparing raw text data that is suitable for analysis and modeling tasks. It is a crucial step to implement in our dataset to achieve more reliable insights and predictions.

For my dataset, I have tokenized text and returned a non-unique list of tokenized words by undertaking the following preprocessing steps:

- Converted the text to lowercase
- Striped punctuation and special characters using regular expressions
- Removed stopwords using NLTK's English stopwords list
- Filtered out non-ASCII characters
- Lemmatized each word to its base form
- Dropped words with a length less than 3

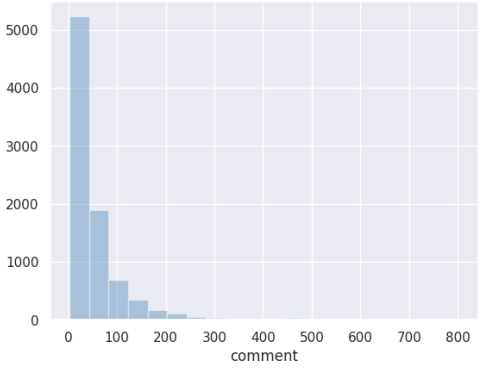

Figure 1: plot showing comment length frequency

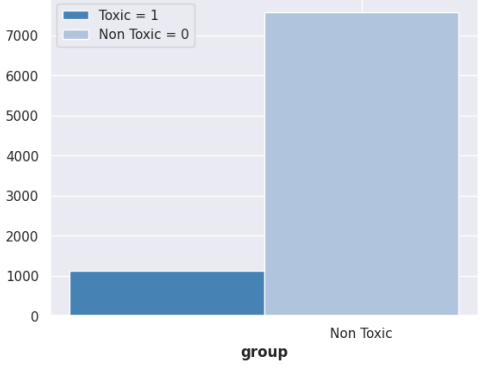

Figure 2: plot showing labeled data frequency

2.3 Data visualization

After preprocessing the data, I performed data visualization. First, I plotted a graph to check the comment length frequency as shown in Figure 1. I observed that most of the comments are short with only a few comments longer than 100 words.

Then, I plotted a graph for the labeled data frequency as shown in Figure 2. I observed that there is significant class imbalance since majority of the comments are labelled as non toxic. 7568 rows out of 8699 rows in training data do not have any label as 1. There is a high chance of overfitting and getting a good accuracy with a model which predicts almost all comments as non toxic. The problem is to fit the data such that it predicts the toxicity accurately.

Then, I plotted a graph Visualizing the most common words contributing to Toxicity as shown in Figure 3.

2.4 Undersampling

While RandomOverSampler, RandomUnderSampler, and SMOTE (Synthetic Minority Oversampling Technique) are effective methods for doing oversampling or undersampling, they

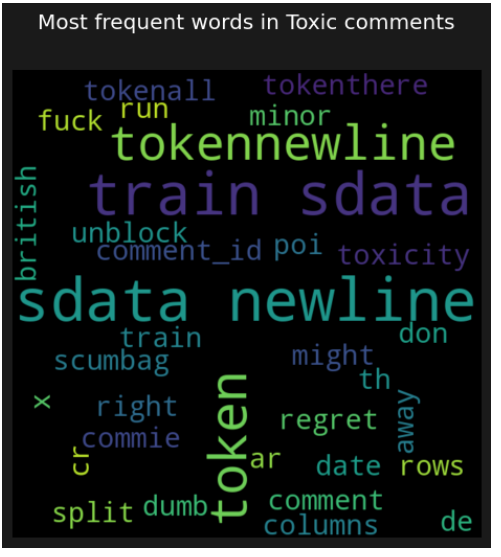

Figure 3: plot showing most frequent words in toxic comments

are less effective when the dataset is severely imbalanced and contains few samples for a given class. Therefore, these methods do not match this dataset well[2].

I executed undersampling by removing a couple of data rows. I removed 35 percent of the comments with all zero labels. This figure was used to avoid drastically reducing the dataset while maintaining a sufficient number of constructive comments for the model to be trained effectively.

After performing undersampling, the shape of the train set is (6099, 4). The distribution of toxicity is slightly more balanced than before with 81.5 percent non toxic data and 18.5 percent toxic data. Now my dataset is ready for feature extraction.

2.5 Feature extraction

Vectorization is performed by converting preprocessed text data into numerical features using a technique called Bag-of-Words (BoW). I also tried using TF-IDF (Term Frequency-Inverse Document Frequency) but did not get better results.

In addition to being the simplest technique for extracting features from text input, the boW is also straightforward to use and comprehend. For text classification and language modeling, the boW is highly appropriate and practical. To implement BoW, the 'CountVectorizer' package is utilized. Word occurrence is computed by CountVectorizer, which also creates a backup word matrix database. The word pool, or boW, consists of all features that have been labeled with labels indicating the occurrences of each category feature [1].

2.6 Hyperparameters

To get optimal hyperparameters for the models, I have utilized Grid Search technique, GridSearchCV().

Multinomial Naive Bayes: An instance of the MultinomialNB Naive Bayes classifier is created. The fit prior parameter determines whether or not to learn class prior probabilities, and the alpha value for Laplace smoothing are two of the parameters that make up the grid of parameters that the Naive Bayes model searches across. A five-split stratified K-fold cross-validation approach is defined. Using the F1-score as the scoring measure to optimize for, GridSearchCV is configured to conduct a grid search over the provided parameter grid. The best hyperparameters obtained from grid search are: 'alpha': 0.1, 'fit prior': False

Logistic Regression: The maximum number of iterations for the Logistic Regression classifier is set to 1000 when it is built. Different class weights to handle class imbalance (None, balanced) and different optimization solvers (newton-cg, lbfgs, liblinear) are defined as a grid of parameters to search over. A five-split stratified K-fold cross-validation approach is defined. Using the F1-score as the optimization metric, GridSearchCV is configured to conduct a grid search over the provided parameter grid. The best parameters obtained from grid search are: 'class weight': 'balanced', 'solver': 'newton-cg'

Support Vector Machine: With a maximum of 100,000 iterations, an instance of the Linear Support Vector Classifier (LinearSVC) is constructed. The regularization parameter 'C', the class weight (to address class imbalance), the fit intercept (fit or no fit), the intercept scaling (intercept scaling factor), and the tolerance for halting criterion (tol) are defined as a grid of parameters to search over. A five-split stratified K-fold cross-validation approach is defined. Using the F1-score as the optimization metric, GridSearchCV is configured to conduct a grid search over the provided parameter grid. The best parameters obtained from grid search are: 'C': 5, 'class weight': 'balanced', 'fit intercept': False, 'intercept scaling': 2.0, 'tol': 0.1

3 Task 3: Analysis and Discussion

For evaluating the models performance, I have computed performance metrics like Accuracy, Recall (macro), Precision (macro), F1 (macro) and Confusion Matrix.

3.1 Justification of Model's performance

The optimal results obtained after training and hyperparameter tuning of the models are shown in Table 2. Let's justify the models performance.

Model	F1 Score	Accuracy	Precision	Recall
Naive Bayes	0.4997	0.7205	0.5038	0.5054
Logistic Regression	0.4930	0.7264	0.4967	0.4955
Support Vector Machine	0.5004	0.7099	0.5061	0.5093
SoTA[SVM]	0.5004	0.7099	0.5061	0.5093

Table 2: Model Performance

Under generative method, Naive Bayes model achieves moderate performance across all metrics. It demonstrates reasonable accuracy but lacks precision and recall, indicating that it may struggle with correctly identifying toxic comments while minimizing false positives.

Under discriminative method, Logistic Regression model performs similarly to Naive Bayes in terms of F1 Score and Accuracy. It shows slightly lower precision and recall, suggesting a comparable performance level. SVM model achieves the highest F1 Score

among the three models, indicating a slightly better balance between precision and recall. However, its accuracy is slightly lower compared to Naive Bayes and Logistic Regression.

To choose the state-of-the-art (SoTA) model among the three models (Naive Bayes, Logistic Regression, and Support Vector Machine), we need to consider their performance metrics. Naive Bayes has the highest accuracy but slightly lower F1 Score compared to SVM. Logistic Regression performs similarly to Naive Bayes in terms of F1 Score and accuracy but shows slightly lower precision and recall. SVM has the highest F1 Score, indicating better overall performance in terms of balancing precision and recall.

With these comparisons, we can consider the Support Vector Machine (SVM) as the state-of-the-art model among the three. It achieves the highest F1 Score and demonstrates a good balance between precision and recall.

3.2 Example and other Analysis

Prediction of toxicity in test set:

- Naive Bayes model has generated values in 'out label model Gen' column having 619 instances as 1 (toxic) and 2277 instances as 0 (non toxic).
- SVM model has generated values in 'out label model Dis' column having 1005 instances as 1 (toxic) and 1891 instances as 0 (non toxic).

Models comparison: As shown in Table 3 I have selected 5 diverse interesting examples from the validation set with their ground truth/true label and compared it with both generative and discriminative models (Naive Bayes and SVM) output.

Comment ID	GT	Generative-NB	Discriminative-SVM
259342186	1	0	0
68802466	1	0	0
111903370	0	0	0
380105292	0	0	1
169266416	1	0	1

Table 3: Comparing generative and discriminative models with diverse examples

Analyzing the predictions made by both the Naive Bayes and SVM models:

- **Example 1** - *"SDATA 6 : NEWLINE TOKENNEWLINE TOKEN==PLEASE STOP IDIOT User: J Milburn,who is a picture deletor vandal!== : EDATA 6"*: Both models incorrectly predicted this example as non-toxic. It seems like the models failed to capture the toxicity of the content, possibly due to the context or nuances in the language used.
- **Example 2** - *"SDATA 6 : NEWLINE TOKENWait, I've got it. Roger sings the lyrics to the chorus, so in this case, Rick is doing the backup. â€" : EDATA 6"*: Similar to Example 1, both models failed to identify the toxicity in this comment. It's possible that the language used is subtle or ambiguous, making it challenging for the models to detect toxicity.
- **Example 3** - *"SDATA 6 : 2007 (UTC)NEWLINE TOKENNEWLINE TOKEN:Your continued rudeness and failure to remotely discuss your controversial administrative*

actions just confirms for me that you are a terrible administrator. Whatever I can do to get your administrative tools taken away from you, I will do. 20:25, 1 March : EDATA 6": Both models correctly identified this example as non-toxic. The content seems to express frustration but does not contain explicitly toxic language.

- **Example 4** - *"SDATA 6 : NEWLINE TOKENNEWLINE TOKENMy user page is now blank. Get over it. I called you a prick only after you had deleted the contents of my userpage and replaced them with a threatening message. Now stop threatening me or I will put one of those templates on your chat page. ( ) : EDATA 6"*: Naive Bayes correctly predicted this example as non-toxic, while the SVM model incorrectly classified it as toxic. The content may contain aggressive language, but it's subjective whether it qualifies as toxic or not.

- **Example 5** - *"SDATA 6 : NEWLINE TOKENNEWLINE TOKEN== fuck you! ==NEWLINE TOKENNEWLINE TOKENhow dare you not rsvp me you bitch! i thought we had something, I'm the best thing you'll ever have. : EDATA 6"*: Naive Bayes incorrectly classified this example as non-toxic, whereas the SVM model correctly identified it as toxic. The language used is clearly aggressive and offensive, which the SVM model captured better.

4 Summary

4.1 Discussion of work carried out

In my work, I focused on automatically detecting toxic comments in online forums using text classification techniques. I explored generative and discriminative machine learning models, including Naive Bayes, Logistic Regression, and Support Vector Machines (SVM), selecting them based on their efficiency, interpretability, and performance on imbalanced datasets like the provided one.

The dataset comprised training, validation, and test sets, with the former two exhibiting significant class imbalance between toxic and non-toxic comments. To address this, I implemented an undersampling technique, removing 35 percent of the non-toxic comments from the training set to balance the classes better.

For feature extraction, I used the Bag-of-Words (BoW) approach, a simple yet effective method for converting text data into numerical features. Although I experimented with TF-IDF, I did not observe improved results.

Hyperparameter tuning was performed using GridSearchCV, optimizing the F1-score to find the best configurations for each model. Ultimately, the SVM model achieved the highest F1-score, precision, and recall on the validation set, thus selected as the state-of-the-art (SoTA) model.

To further improve the performance of the toxic comment detection system, a few potential enhancements could be considered:

- investigating more sophisticated text representation methods, including word embeddings or contextual language models (like BERT), which could improve the models' capacity to detect subtle types of toxicity by capturing more semantic information.

- looking into ensemble techniques like stacking or boosting, which may be able to take use of the advantages of several models and even outperform the individual models.

- Including other features that could be useful in identifying harmful activity, including user metadata or discussion context.
- investigating methods for data augmentation to solve the problem of class imbalance and enhance the generalization abilities of the models.

4.2 Lessons Learned

After carefully reviewing my work, there are several areas where the methods could be improved to potentially achieve better results in the toxic comment detection task:

- **Exploring advanced text representation techniques**: Investigating more sophisticated text representation methods, like contextual language models (BERT, RoBERTa) or word embeddings (Word2Vec, GloVe), could help the models detect subtler kinds of toxicity by capturing more semantic information.
- **Investigating ensemble methods**: Can take into account putting ensemble techniques like stacking or boosting into practice, which make use of the advantages of several models. Through the amalgamation of predictions from many models (such as Naive Bayes, Logistic Regression, and SVM), the ensemble technique has the potential to surpass the individual models and attain superior performance.
- **Addressing class imbalance more effectively**: Investigating alternative methods, like class-weighted loss functions or oversampling the minority class (using SMOTE, for example) could enhance the models' capacity to learn from the unbalanced data.
- **Conducting thorough error analysis**: Carrying out a thorough error analysis to pinpoint the kinds of remarks that the models have trouble accurately classifying (e.g., context-dependent toxicity, sarcasm, and subtle forms of toxicity). This analysis may help with feature engineering and model advancements in the future.

References

[1] S. Rahamat Basha and T. Bhasara Reddy. Imbalanced text features for toxic comments classification. *International Journal for Modern Trends in Science and Technology*, 8: 313–317, 2022.

[2] Ankita Choudhury. Toxic-comment-classification, 2020. https://github.com/ankita1112/Toxic-Comment-Classification.

[3] Andrew Ng and Michael Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In T. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems*, volume 14. MIT Press, 2001.

[4] Jay Speidell. Toxic comment classification - natural language processing, 2018. https://jayspeidell.github.io/portfolio/project05-toxic-comments/.