Report of Toxic Comment Classification

PAN JUNCHEN jpan009@e.ntu.edu.sg
ZHONG YUJIA yzhong020@e.ntu.edu.sg
DANG ZHIXIN dang0034@e.ntu.edu.sg
PARIS SIRIPATANA paris001@e.ntu.edu.sg

Contribution of each member: All contributed equally

Abstract

The existence of toxic comments on the internet is a pressing issue that affects the overall online experience and can cause harm to individuals and communities. Since we understand how important it is to have a secure and happy online environment, our team has provided three NLP models (word2int, BERT, and GloVe) for identifying and categorizing toxic remarks as a way to address the issue. These models have the ability to stop the spread of destructive remarks and encourage productive debates, making the internet a safer and more pleasant environment for everyone.

1. Introduction

The rise of modern society has led to an increase in internet usage, with more people opting to communicate online as it is a more efficient way of exchanging information and emotions. However, due to platform regulations, toxic comments are usually not displayed to the public. These comments can discourage individuals from expressing their opinions and participating in discussions, leading to potential abuse and harassment. The presence of toxic comments on the internet has a negative impact on everyone's online experience and can harm individuals and communities. Therefore, it is vital to identify and address toxic comments to create a safer and more positive online environment. In this project, our team aims to present three Natural Language Processing models for detecting and categorizing toxic comments.

2. Background

The training data contains eight attributes which are id, comments_text, toxic, severe_toxic, obscene, threat, insult, and identity_hate. Our team will use models to classify toxic comments into the correct category.

To build a successful model for this project, it will be necessary to preprocess and clean the dataset carefully, as well as select and tune the Natural Language Processing models to achieve high performance on the task. This may involve exploring different types of models, such as linear models, tree-based models, or neural networks, and choosing the model that performs best on the given dataset. It will also be essential to consider how to handle imbalanced classes, as the data may be unevenly distributed across the different types of toxicity, and to carefully evaluate the model's performance using appropriate metrics.

3. Method

Word2int: Word2int is a mapping which transforms a word or string into a numerical representation. Word2int will create a list of all the one-of-a-kind words found in a text corpus and then give each word a special number. After the mapping is put together, the text corpus can be changed into numerical forms by swapping out each word with its matching numerical index.The resulting numerical forms can be used as input for various NLP tasks, such as toxic comments. It is used in this project to take the words from a text corpus and turn them into numerical forms that can be fed into models. In the constructor, the embedding layer, the LSTM layer, the dropout layer, the linear output layer, and the sigmoid layer are initialized. Embedding transforms the input into a vector representation. A dropout layer is used to prevent overfitting. After that, taking the input and converting it to a long, the forward function maps it to a vector using the embedding layer. Following the LSTM layer, the dropout and fully connected layers are applied, followed by the sigmoid layer with the final output.

BERT: BERT, also known as Bidirectional Encoder Representations from Transformers, is a pre-trained language model that uses a cutting-edge deep learning method, the "transformer," to thoroughly analyze and comprehend the meaning of words in a sentence. With its training on a massive corpus of text data, BERT has the ability to comprehend the context of words and excel in a range of NLP tasks, like text classification, answering questions, and sentiment analysis.sks, such as text classification, question answering, and sentiment analysis. And in this project, it will also be used to detect and categorize toxic comments. In our code, the constructor initializes a pre-trained BERT model before adding a linear layer for classification, followed by a sigmoid activation function.The BERT model is used to process inputs in the forward function, and then the results are passed to linear models. To get the final result, the output is passed through the sigmoid activation function. Based on the output size of the linear layer, the classifier classifies the input text into 6 different classes.

GloVe: GloVe, also known as Global Vectors for Word Representation, is a unique type of word embedding model that transforms words into vectors in a high-dimensional space. Unlike the conventional Word2Vec model, which relies on a neural network to produce word embeddings, GloVe uses a sophisticated statistical method to understand the connections between words and produce word vectors. GloVe can capture both the semantic and syntactic relationships between words, making it suitable for a range of NLP tasks, like the Toxic Comment Classification. Our code employs a vectorize_batch function, which takes in a batch of data from the DataLoader. The function tokenizes the comments, pads or truncates the tokens to a maximum length of 256, and converts them to tensors using a pre-trained vectorizer. The vectorized comments and the labels are then returned as a tuple.

For all the three models, they have the training process for a multi-label classification problem. The code contains two main functions, “CalcValLossAndAccuracy” and “TrainModel”. The first function, CalcValLossAndAccuracy, calculates the loss, accuracy, F1 score, and AUC (Area Under the Curve) of the validation data. The calculation is done using the loss_fn which is the specified loss function, the model,

4. Experiments

- ### Data Description

This report uses data from the Toxic Comment Classification Challenge on the Kaggle website as the basis for the experiment. The original dataset consisted of three files: "train.csv", "test.csv", and "test_labels.csv". The "train.csv" file contained comments with binary labels indicating whether they were toxic or non-toxic. The "test.csv" file contained only the comment text, while the "test_labels.csv" file contained the labels for the test data.

To better suit the purpose of this experiment, the original dataset was processed into two new files, "tokenized.csv" and "test_tokenized.csv", through data pre-processing. The "tokenized.csv" file represents the processed training set, while the "test_tokenized.csv" file is a combination of the "test.csv" and "test_labels.csv" files after undergoing the same pre-processing steps.

The processed data contains comments in text format and the corresponding toxic label for each comment. The data has been tokenized, with the text broken down into individual words and converted into numerical representations for analysis by machine learning models. The data has also been pre-processed to remove stop words, punctuation, and special characters and to convert all text to lowercase. The processed data provides the input for the machine-learning models proposed in this report to classify toxic comments.

- Data Exploration

1. The original training set train.csv was analyzed to determine the distribution of the toxicity labels for the comments. The number of comments for each toxicity label was counted to provide insight into the proportion of toxic and non-toxic comments in the dataset. This information can help understand the overall makeup of the data and inform the design of the machine-learning models to be used in classifying toxic comments.

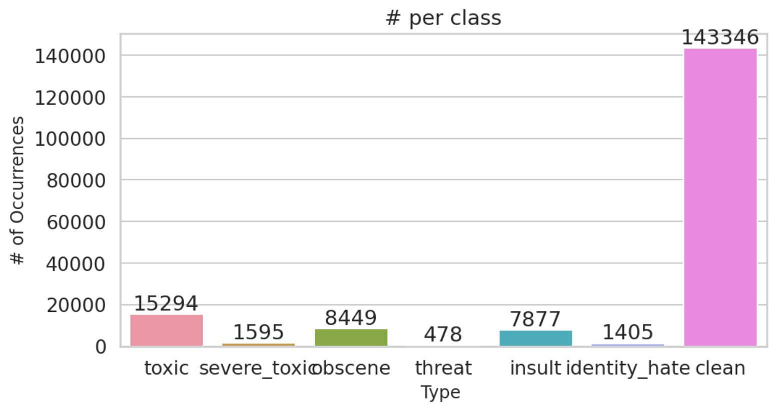

Figure 1: Count of Comments for each Toxicity Label in the train.csv

2. The following figure presents the count of comments in the train.csv with varying numbers of toxicity labels. Each comment in the train.csv may have multiple labels, and this figure provides insight into how many comments have one, two, three or more toxicity labels.

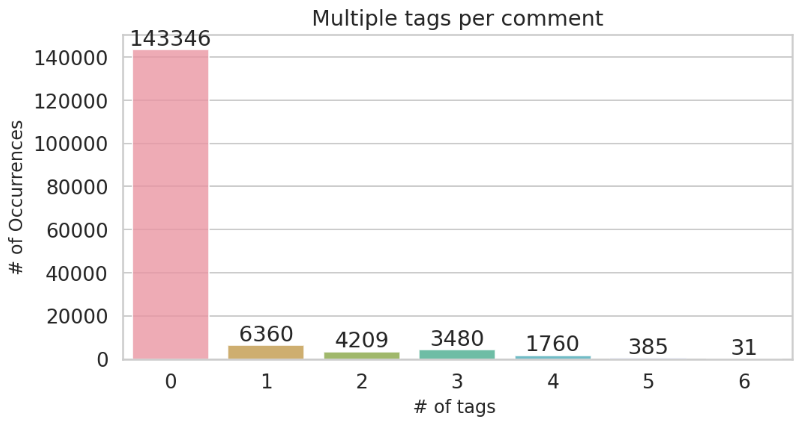

Figure 2: Count of Comments with Different Numbers of Toxicity Labels in the train.csv

3. The following figure provides a powerful visual representation of the words used in the clean and toxic comments in the train.csv. Word cloud extraction highlights the most frequently occurring words in each category, presenting a clear picture of the language used in toxic and non-toxic comments.

The word cloud of clean comments provides insight into the language used in positive online interactions. In contrast, the separate word clouds for different toxic categories demonstrate the language used in harmful comments. These figures not only serve as a clear and effective means of analyzing the data but also provide a deeper understanding of the language used in toxic and non-toxic comments. They can aid in developing effective models to identify toxic comments in the future.

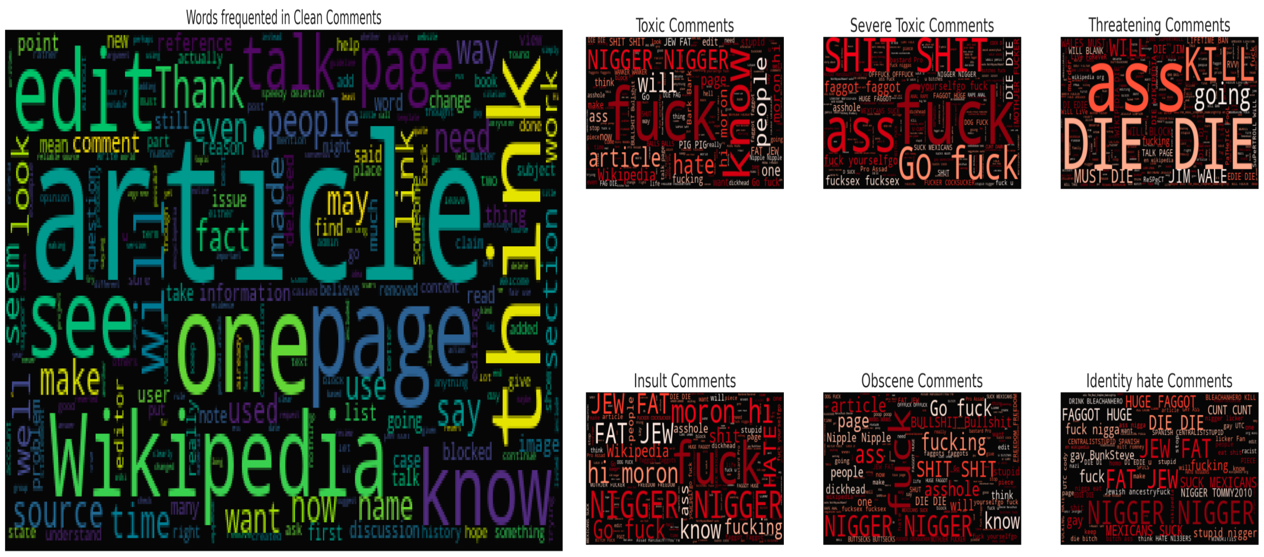

Figure 3: Word Cloud of Clean and Toxicity Comments in the train.csv

- Data Pre-processing

Data pre-processing plays a critical role in the success of the Toxic Comment Classification task. The raw text data collected from various sources must be cleaned and prepared to ensure that the machine learning models used in this task perform optimally.

In this stage, multiple pre-processing steps are applied to the text data to remove irrelevant information and improve its quality, which includes removing links, HTML tags, punctuations, numbers, non-ASCII characters, and emojis and correcting spelling errors. The removal of stop words, which have limited significance, also helps reduce the data's complexity and makes it easier to manage for further analysis.

Lemmatization is another critical step in the pre-processing stage. It reduces the words in the text to their base form, avoiding repeated words in different forms and changing the dimensionality of the data. This step also makes the words in the text in their base form, which is easier to analyze and understand the text.

The final processed data is saved as a "tokenized .csv" file, and the same applies to the test set, where "test.csv" and "test_labels.csv" are merged and saved as "test_tokenized.csv". These files serve as the input for the machine-learning models used in the classification task.

In conclusion, proper data pre-processing is indispensable in achieving precise results in the Toxic Comment Classification task. It is essential to ensure that the text data is cleaned, processed, and ready for analysis. Appropriately selecting and implementing pre-processing steps can significantly improve the performance of the machine-learning models used in this task.

- Evaluation method

The evaluation of our NLP model is a significant section. As mentioned in the previous section, our team made three models which are word2int, Bidirectional Encoder Representations from Transformers (BERT), and Global Vectors for Word Representation (GloVe). In this paragraph, the evaluation methodology and metrics for assessing the performance of our model are described.

There is an essential evaluation metric for the general machine learning model which is accuracy. By measuring accuracy, we determine how often our model correctly predicts the input text's class. In our project, we made our model to classify the toxic comments into the correct category, and which contains six classes, “toxic”, “sever toxic”, “obscene”, “threat”, “insult”, and “identity hate”. Based on the total number of samples in the evaluation set, the accuracy is calculated as the number of correctly predicted samples divided by the total number of samples. Additionally, in the model of word2int, due to the property of the model itself, the model can only classify the toxic label since it is the most common label, and the vocabulary for training is only word2int, which does not include the meaning of each word. To make a better evaluation, this project also employed another three methods which are Precision, Recall, and F1-score for the rest of the model.

- Model settings

Different models have their own model settings, and this section will demonstrate their settings. The general train and test split in this project are 95% of the data will be used as training data and the rest of the 5% of the data will be employed as testing data. For the word2int model, further data pre-processing is still needed which converts the text data into numerical format. The general architecture of this model is one embedding layer, one LSTM layer, one Linear layer and Sigmoid layer. Embedding layers are useful to map input into vector representation. And we use the Sigmoid layer since we will have binary classification. The batch size of this model has been set as 64, and the embedding size & hidden size are 256, 512, respectively. Besides, for all three models the loss function being used is binary cross-entropy loss, and we employed the Adam optimizer with the learning rate of 0.001.

- Results

Model comparison			
	Word2int	BERT	GloVe
Train Loss	0.144	0.021	0.045
Train Acc	0.963	0.993	0.984
Valid Loss	0.154	0.059	0.065
Valid Acc	0.962	0.982	0.977

Figure 4: Model comparison

To compare these three models, this report uses the Adam optimizer with epochs =15, learning_rate = 1e-6 * 5.

Based on the training and validation results provided above, BERT is performing the best with a training accuracy of 99.3% and validation accuracy of 98.2%. It has the lowest training loss (0.021) and a moderate validation loss (0.059).

Word2int is also performing well with a training accuracy of 96.3% and a validation accuracy of 96.2%. The training loss is slightly higher compared to BERT (0.144) and the validation loss is also higher (0.154).

GloVe is performing the worst among the three models with a training accuracy of 98.4% and a validation accuracy of 97.7%. The training loss is lower compared to Word2int (0.045), but the validation loss is higher (0.065).

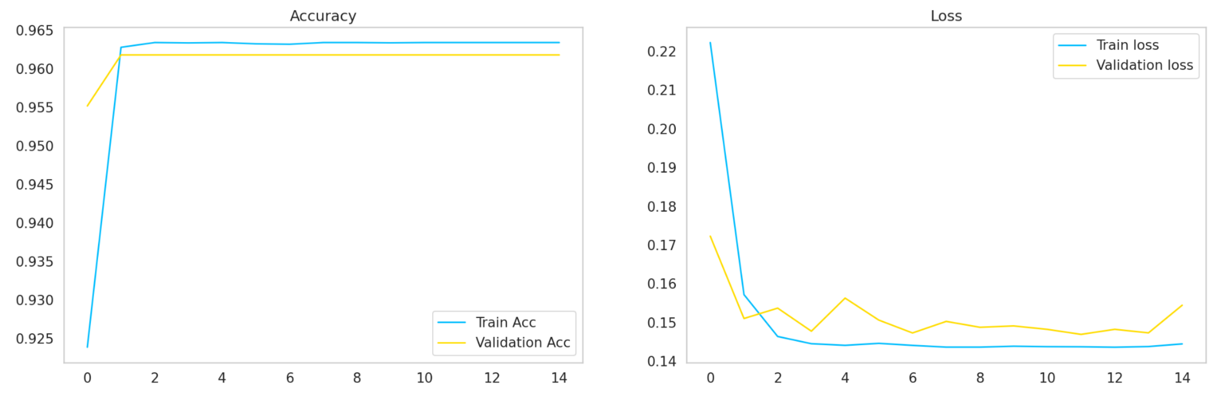

Figure 5: The Accuracy and Loss of Word2int Model

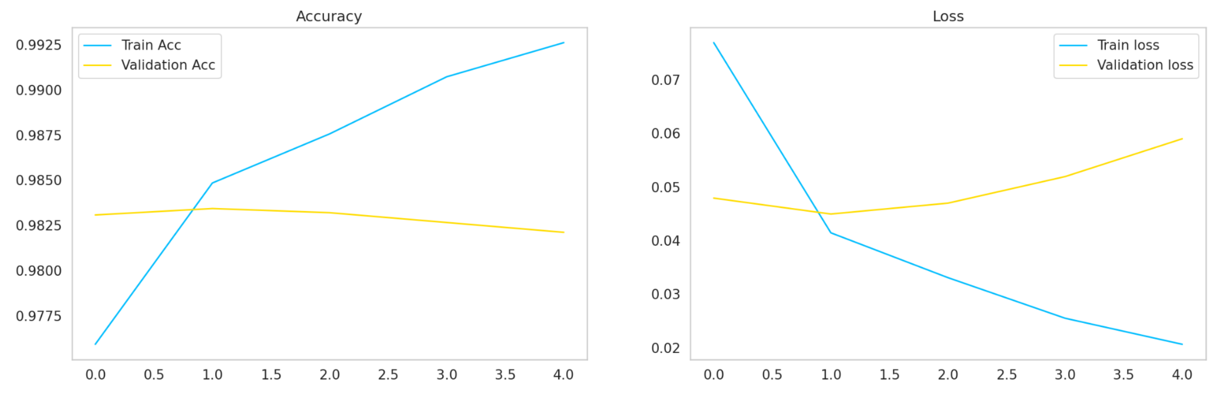

Figure 6: The Accuracy and Loss of BERT Model

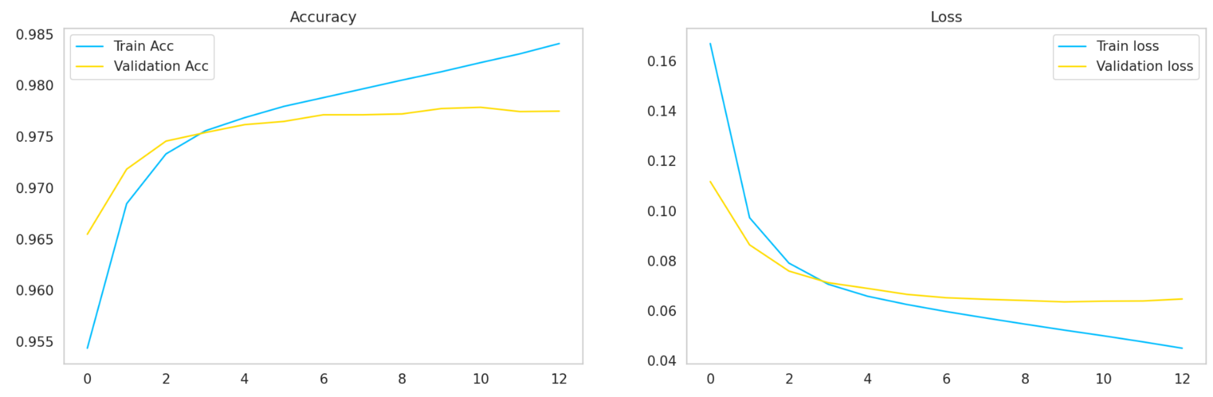

Figure 7: The Accuracy and Loss of GloVe Model

The graphics above show that the train accuracy for all three models grows as the number of epochs rises, with the Bert model showing the most improvement by reaching 99.3% at a relatively low number of epochs.

However, comparing models by accuracy is actually inaccurate, so we will use a comparison of the various metrics such as accuracy, precision, recall, and F1-score to determine their performance. The word2int model only converts text into numerical values and is not capable of comprehending the actual content, which is why it was not utilized.

Model Comparison		
	BERT	GloVeÍ
Accuracy	0.993	0.984
Precision	0.60	0.58
Recall	0.73	0.54
F1-score	0.66	0.56

Figure 8: The comparison of BERT & GloVel models.

Based on the Accuracy, Precision, Recall and F1-score results provided above, BERT is performing the best with the highest Accuracy (99.3%), Precision (60%), Recall (0.73) and F1-score (0.66). In conclusion, the project should use the BERT model to accurately classify the comments into the correct categories.

5. Analysis

The toxic comment classification system assigns comments to one of six categories: "toxic", "severe toxic", "obscene", "threat", "insult", and "identity hate". The system works by using NLP models to analyze the text of the comments and predict which category it belongs to.

When the model succeeds, it is able to accurately classify the comments into the correct categories, indicating a high level of understanding of the characteristics of each category. The model's performance can be improved by training it on a large and diverse dataset of comments, as well as by fine-tuning the model architecture and hyperparameters.

However, the model may fail or produce incorrect predictions when the text of the comments contains subtle or implicit sarcasm, irony, or uses words in a way that is not typical. Additionally, the model may struggle with identifying toxic comments that use new or rare words or phrases that it has not seen in its training data.

6. Conclusion

Based on this project, we developed a model to classify toxic comments into six categories: "toxic", "severe toxic", "obscene", "threat", "insult", and "identity hate".

The main findings of the project indicate that the Bert model was the best among the three models, with an accuracy of 99.3% and F1-score (0.66). This suggests that the Bert model was able to effectively classify the toxic comments into the correct categories.

In terms of achievements, the project successfully created a model to categorize toxic comments, which has the potential to make the internet a safer and more enjoyable place for individuals and communities.

However, there are limitations to the work as well. The model may struggle with comments that contain subtle sarcasm or irony, or use words in a way that is not typical. Additionally, the model may not be able to accurately categorize new or rare words or phrases that it has not seen in its training data.

To address these limitations, future work could include continually updating the model with new training data and fine-tuning the model architecture and hyperparameters. Additionally, incorporating other techniques such as transfer learning or ensembling multiple models may also improve the performance of the toxic comment classification system. It may also be necessary to continually update the model with new training data and to evaluate its performance on a regular basis to ensure that it is effectively categorizing toxic comments.

Reference

Khaung, K. (2021) Multi-label text classification with Bert Using Pytorch, Medium. Medium. Available at: https://kyawkhaung.medium.com/multi-label-text-classification-with-bert-using-pytorch-47011a7313b9 (Accessed: February 17, 2023).
Multi-label text classification with Bert and Pytorch Lightning (no date) Curiously. Available at: https://curiousily.com/posts/multi-label-text-classification-with-bert-and-pytorch-lightning/ (Accessed: February 17, 2023).
Omkarcgode (2022) Toxic comment classification - NLP using pytorch, Kaggle. Kaggle. Available at: https://www.kaggle.com/code/omkarcgode/toxic-comment-classification-nlp-using-pytorch/notebook (Accessed: February 17, 2023).
Solanki, S. (2022) Word embeddings for pytorch text classification networks by Sunny Solanki, Developed for Developers by Developer for the betterment of Development. CoderzColumn. Available at: https://coderzcolumn.com/tutorials/artificial-intelligence/word-embeddings-for-pytorch-text-classification-networks (Accessed: February 17, 2023).
Solanki, S. (2022) Word embeddings for pytorch text classification networks by Sunny Solanki, Developed for Developers by Developer for the betterment of Development. CoderzColumn. Available at: https://coderzcolumn.com/tutorials/artificial-intelligence/word-embeddings-for-pytorch-text-classification-networks (Accessed: February 17, 2023).
Torch (no date) torch. Available at: https://pytorch.org/docs/stable/torch.html (Accessed: February 17, 2023).
Toxic comment classification challenge (no date) Kaggle. Available at: https://www.kaggle.com/competitions/jigsaw-toxic-comment-classification-challenge/data (Accessed: February 17, 2023).