Imbalanced Toxic Comments Classification using Data Augmentation and Deep Learning

Amira Bendjama, Nikhil Muthuvenkatesh, Seyi Oyesiku, Connor Roth

***Abstract*—With the rise of cyber-bullying and online harassment, addressing toxicity in user-generated content has become a critical challenge in public online communities. This paper focuses on utilizing Wikipedia's Toxic Comment as training data to develop binary and multi-label classifiers that are capable of detecting various forms of toxicity in online discussions. The solution includes four models: a Bidirectional LSTM and BERT model for binary classification of toxic and non-toxic comments, and a BERT model and Bidirectional LSTM for multilabel classification of different types of toxicity. The bi-LSTM models effectively capture contextual information and interdependencies to accurately classify toxic comments and identify specific types of toxicity.**

***Index Terms*—natural language processing, sentence classification, multi-label classification, deep learning, long short-term memory, Bidirectional Encoder Representations from Transformers**

I. Experiment Overview

In 2018, Jigsaw Conversation AI sponsored a public competition on Kaggle. The task involved correctly classifying toxic comments scraped from Wikipedia chat rooms.

A group of researchers tried their hand at the task from an academic perspective, publishing their results. Specifically, their research involved two components: binary classification of "toxic" versus "non-toxic" and multi-label classification of the different types of "toxic" comments [1]. They were able to obtain a "0.828 and 0.872 F1-score for toxic/non-toxic classification and toxicity types prediction respectively" [1]. It is this paper that our group used as the basis of this experiment.

So, the experiment is thus: conduct both toxic/non-toxic classification and toxicity multi-label classification and attempt to beat the F1 scores obtained. For each task, the team created a bi-LSTM and BERT classification models.

II. Experimental Design

A. Dataset

The training dataset contains eight columns, including an ID column and the comment text column, which serves as the input for our modeling efforts. The remaining columns represent different labels for classification, such as toxic, severe toxic, obscene, threat, insult, and identity hate. We will further explore these labels during the EDA phase. The training dataset contains approximately 159,571 comments, while the cleaned test dataset has around 63,978 comments. We observe that the test dataset had a similar size to the training dataset initially but was reduced after removing unscorable comments.

B. Exploratory Data Analysis

To gain a deeper understanding of the dataset, we review examples of training comments and analyze their lengths. This analysis reveals that comments are, on average, approximately 394 characters long, with a wide variation indicated by the large standard deviation. We visualize the distribution of comment lengths using a histogram. Additionally, we explore the labels themselves to understand their descriptive statistics and interactions. We create a new label column, "none," to better understand the distribution of labels and observe that there is an overlap among the labels. This indicates the need for a multi-label classification approach. Finally, we check for null or missing values in the dataset, and fortunately, no such values are found.

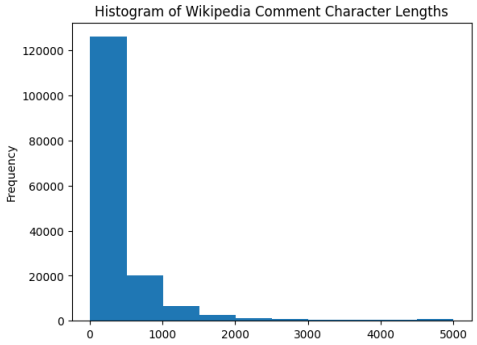

Fig. 1: Histogram of Wikipedia Comment Character Lengths

C. Pre-processing

1) General Pre-Processing: We performed preprocessing tasks on the dataset in two steps. Firstly, it generates additional binary labels to classify comments as toxic or non-toxic based on the existing toxicity labels. The maximum value of the individual toxicity labels is calculated, and if any of them have a value of 1 (indicating toxicity), the new label is assigned a value of 1 as well. Secondly, the comment text undergoes cleaning and transformation. Unnecessary characters and whitespace are removed, leaving only letters and punctuation marks. The text is then converted to lowercase. This preprocessing is applied to both the training and test datasets, resulting in a new column with processed comment text. These steps aim to prepare the dataset for further analysis and modeling by creating binary toxicity labels and ensuring consistency in the comment text.

2) Bi-LSTM Pre-Processing: Next, for the Bi-LSTM implementations, the data is tokenized using a tokenizer with a maximum of 100,000 features. This tokenizer is then fit on the text data from the training set, resulting in sequences that represent the text. These sequences are subsequently padded to a maximum length of 300 to ensure consistent input sizes. To facilitate efficient training, a batch size of 32 is set, and the inputs and labels are converted into tensors and create Dataloaders. Finally, GloVe 840B 300d was used to create an embedding matrix.

3) BERT Pre-Processing: For BERT implementations, we start by obtaining the BERT tokenizer, which is essential for handling out-of-vocabulary words and reducing the vocabulary size. Next, we perform BERT-specific preprocessing, which involves adding special tokens at the start and end of each sentence, padding and truncating sentences to a fixed length, and creating an attention mask to distinguish real tokens from padding tokens. After splitting the data into train and validation sets, we apply the BERT preprocessing function to obtain input IDs and attention masks for both the training and validation data. Finally, we convert the masks to tensors and set the batch size.

D. Model Building

1) Bidirectional LSTM Model for Binary Classification : The model-building process involves constructing a binary classification model using a bidirectional LSTM in Keras. The architecture is designed to effectively capture the sequential information in the input data. The first layer, the embedding layer, maps the input sequence into dense vector representations, which helps to capture the semantic meaning of the words or tokens in the sequence. The bidirectional LSTM layer is then employed to process the embedded sequence, capturing both forward and backward information. This allows the model to have a comprehensive understanding of the context and dependencies within the input sequence. Finally, a dense layer with a Sigmoid activation function is utilized for binary classification, producing a probability score for the positive class.

To train the model, the Adam optimizer from Keras is utilized, which efficiently updates the model parameters based on the gradients of the loss function. The binary cross-entropy loss function is employed, which is well-suited for binary classification tasks. During training, the model's performance is monitored using a validation set, allowing for early stopping if the performance does not improve. The accuracy metric is used to evaluate the model's performance during training, providing a measure of how well the model is able to correctly classify instances. After training, the model is applied to the test data, and predictions are generated. These predictions are initially in continuous values, representing the probability of belonging to the positive class. By applying a threshold of 0.5, the continuous predictions are converted into binary labels, allowing for clear classification into the positive and negative classes.

2) BERT Model for Binary Classification: The BERT implementation takes the pretrained BERT base uncased model as its first "layer". Then, the following layers were applied for binary classification:

- Input Linear Layer: This layer maps the input to the hidden layer with a linear transformation.
- ReLU Activation Layer: This is a hidden layer that introduces non-linearity to the output of the linear layer.
- Output Linear Layer 2: This linear layer takes the output from the hiden layer to make predictions with logits.

Like the bi-LSTM model, the BERT model uses the Adam Optimizer and BCE Loss. This model is also trained in batches of size 32 with the gradients backpropagated and parameters updated to improve the predictions.

3) Bidirectional LSTM for Multi-label Classification: The multi-label model represents a different implementation style from that of the binary model. Here, PyTorch is used and consists of several layers that differ from that of the binary model:

- Embedding layer: It converts input features into dense vectors of fixed size.
- LSTM layer: It processes the embedded input sequence bidirectionally, capturing contextual information.
- Average and Max Pooling layers: They extract relevant information from the LSTM output by taking the average and maximum values respectively.
- Concatenation layer: It concatenates the average and max pooled features.
- Linear layer 1: It maps the concatenated features to a hidden dimension using a linear transformation.
- ReLU activation: It introduces non-linearity to the output of the linear layer.
- Dropout layer: It randomly drops a fraction of the neurons during training to prevent overfitting.
- Linear layer 2: It maps the output of the previous layer to the desired output dimension.
- Sigmoid activation: It applies the sigmoid function to produce probabilities for multilabel classification.

Like the other models, training involves optimizing the model's parameters using the Adam optimizer offered by torch and minimizing the binary cross-entropy loss. The model is trained in batches of size 32, with gradients backpropagated and parameters updated to improve predictions. A separate validation set is used to evaluate the model's performance.

4) BERT Model for Multi-Label Classification: Similar to the binary implementation, the multi-label model takes BERT base uncased as its first "layer" before layering on the following:

- Input Linear Layer: This layer maps the input to the hidden layer with a linear transformation.
- ReLU Activation Layer: This is a hidden layer that introduces non-linearity to the output of the linear layer.
- Output Linear Layer 2: This linear layer takes the output from the hiden layer to make predictions with logits.

Again, optimization is achieved through Adam optimizer, and loss is minimized through binary cross-entropy loss. This model is also trained in batches of size 32 with the gradients backpropagated and parameters updated to improve the predictions. The same evaluation function that was used for

Bi-LSTM multi-label classification was also used to evaluate the BERT multi-label model.

III. Results

1) Bidirectional LSTM Model for Binary Classification : The binary classification Bi-LSTM was trained for two epochs on the training data. After the first epoch, the training loss was 0.1500 with an accuracy of 0.9486. In the second epoch, the loss decreased to 0.1038, and the accuracy improved to 0.9630. The validation results also showed improvement, with a validation loss of 0.1149 and accuracy of 0.9609 after the first epoch, and a loss of 0.1018 and accuracy of 0.9643 after the second epoch. These results demonstrate that the Bi-LSTM model effectively captures patterns and relationships in the binary classification task, showing a decreasing loss and increasing accuracy as it learns from the training data and generalizes well to new examples.

TABLE I: Training Data Results: Bi-LSTM for Binary Classification

Epoch	Loss	Accuracy	F1
1	0.1483	0.9486	0.6271
2	0.1009	0.9633	0.7455

TABLE II: Validation Data Results: Bi-LSTM for Binary Classification

Epoch	Validation Loss	Validation Accuracy	F1
1	0.1102	0.9604	0.7242
2	0.1034	0.9633	0.7619

TABLE III: Test Results: Bi-LSTM for Binary Classification

Test Loss	Test Accuracy	Test F1
0.1894	0.9238	0.63

2) BERT Model for Binary Classification: Unfortunately, despite continued efforts to problem-solve, the team was not able to run the BERT model for binary classification at all. The issue, a memory exception error, likely stems from a combination of the large size of BERT and a persistent memory leak.

3) Bidirectional LSTM for Multi-label Classification: After training for two epochs, the multilabel Bi-LSTM model achieved a weighted F1 score of 0.73 on the training set, indicating its ability to capture patterns and classify multilabel outputs effectively. On the validation set, the model performed slightly better with a score of 0.75, demonstrating its generalization capability to unseen data. However, on the test set, the model's performance dropped to a score of 0.61, suggesting potential overfitting to the training and validation data.

TABLE IV: Weighted F1 Score for BiLSTM Model in Multilabel Classification

Dataset	Weighted F1 Score
Training	0.73
Validation	0.75
Test	0.61

Digging into the specific by-label F1 scores, we see that the disparity between class sizes plagues the model. Classes that have a relatively high incidence, like "toxic", have a much higher F1 score than "threat".

TABLE V: F1 Scores by Label for BiLSTM model in Multilabel Classification

Label Metrics by Dataset	Training	Validation	Test
toxic	0.82	0.85	0.67
severe_toxic	0.05	0.09	0.09
obscene	0.82	0.83	0.69
threat	0.00	0.00	0.00
insult	0.72	0.75	0.60
identity_hate	0.22	0.20	0.22

4) BERT Model for Multi-Label Classification: Here, too, the same issue plagued the team's attempt to run the model. That is, a memory exception issue occurred, and, while hypotheses were tested, no solution was able to be found.

IV. Conclusion

Overall, the learnings from this project were highly informative. On one hand, there are two models - one for binary classification and one for multilabel classification - that can perform reasonably well on their respective tasks. On the other, implementing BERT proved unworkable despite significant time investment from the team.

Looking ahead, further work would be necessary to identify and remedy the apparent memory leak. The implementation of these would allow for the desired comparison of LSTM-based models against transformer-based models.

References

[1] Nagwa El-Makky Mai Ibrahim Marwan Torki. "Imbalanced Toxic Comments Classification using Data Augmentation and Deep Learning". In: *IEEE* (2018).