AUTOMATIC HATE SPEECH DETECTION USING NLP

BERT Implementation

Data Glacier

LISUM 2022-2023

The Hate Speech Detection Using BERT: A Deep Learning Approach

Abstract:

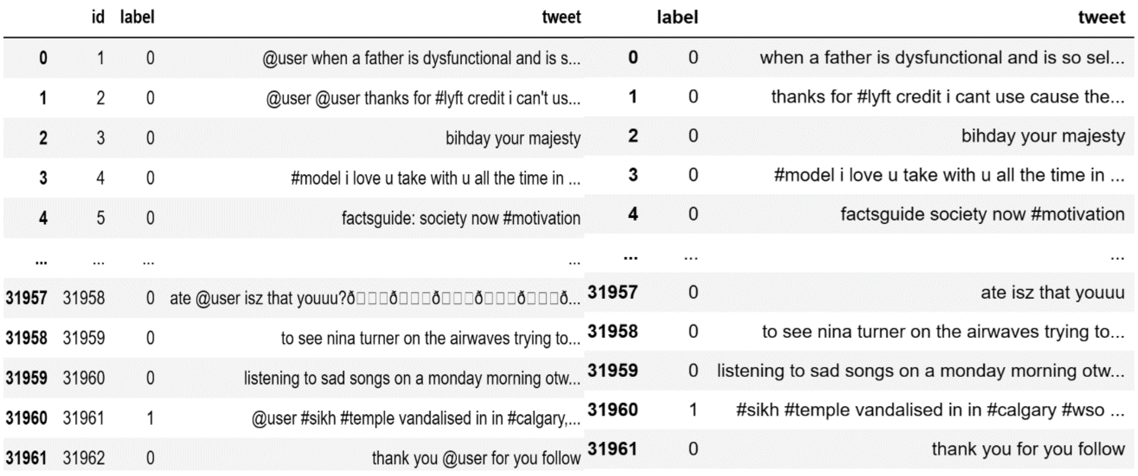
In an age where social media platforms host a vast array of online content, the detection of hate speech has gained significant importance. This report presents a study on the application of the BERT model, a powerful transformer-based neural network architecture, for the task of hate speech detection in Twitter data. The report provides insights into the methodology, data preprocessing, model architecture, training, evaluation, and the results achieved in this project.

A. Introduction:

Online hate speech has become a pervasive issue, causing harm and division within online communities. Detecting such speech automatically is critical in maintaining a safe online environment. This report outlines the research conducted to develop and evaluate a model for hate speech detection. We will explore the data preprocessing, model selection, training, testing, results and limitations.

B. Data Preprocessing:

Data collection for this study involved obtaining Twitter data relevant to the hate speech detection task. The collected data was subjected to a rigorous data cleaning process, including the application of regular expressions to remove unwanted characters, elements and unidentified characters as shown in figures 1 and 2 .



	id	label	tweet
0	1	0	@user when a father is dysfunctional and is s...
1	2	0	@user @user thanks for #lyft credit i can't us...
2	3	0	bihday your majesty
3	4	0	#model i love u take with u all the time in ...
4	5	0	factsguide: society now #motivation
...	...	...	...
31957	31958	0	ate @user isz that youuu?ð□□□ð□□□ð□□□ð□□□ð□□□ð...
31958	31959	0	to see nina turner on the airwaves trying to...
31959	31960	0	listening to sad songs on a monday morning otw...
31960	31961	1	@user #sikh #temple vandalised in in #calgary,...
31961	31962	0	thank you @user for you follow

	label	tweet
0	0	when a father is dysfunctional and is so sel...
1	0	thanks for #lyft credit i cant use cause the...
2	0	bihday your majesty
3	0	#model i love u take with u all the time in ...
4	0	factsguide society now #motivation
...	...	...
31957	0	ate isz that youuu
31958	0	to see nina turner on the airwaves trying to...
31959	0	listening to sad songs on a monday morning otw...
31960	1	#sikh #temple vandalised in in #calgary #wso ...
31961	0	thank you for you follow

Fig 1 & 2: Pre-processed dataset and post-processed dataset

C. Dataset Splitting:

To effectively train and evaluate our model, the dataset was divided into three subsets: a training set, a validation set, and a testing set. The split ratios were chosen to facilitate model training while ensuring an unbiased evaluation. Split ratios were 70%-30% where 70% is for training and 30% is for validation and testing divided equally as shown in figures 3 and 4.

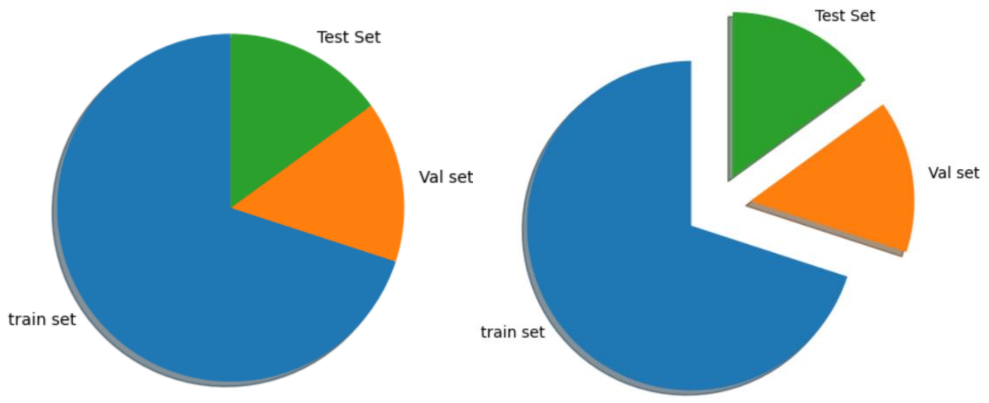


Fig 3 & 4. Visual representation of dataset division.

D. BERT Model and Tokenization:

The model is built upon the BERT (Bidirectional Encoder Representations from Transformers) architecture. BERT is a state-of-the-art transformer-based model renowned for its performance in various natural language processing tasks. The model and its tokenizer were loaded and integrated into the built architecture. Additionally, an analysis of tweet length was conducted using a histogram to understand the data distribution. The tokenization process was applied to convert tweet texts into tensors, making them suitable for model training.

E. Model Architecture:

In a BERT-based architecture, several key components are integrated to facilitate the process of hate speech detection. These components play distinct roles in transforming and processing the input data to make predictions. In this custom BERT architecture, essential components were included such as BERT layers, dropout layers to prevent overfitting, activation functions, and fully connected layers. The purpose of each architectural element is explained below to provide a comprehensive understanding as shown in figure 5.

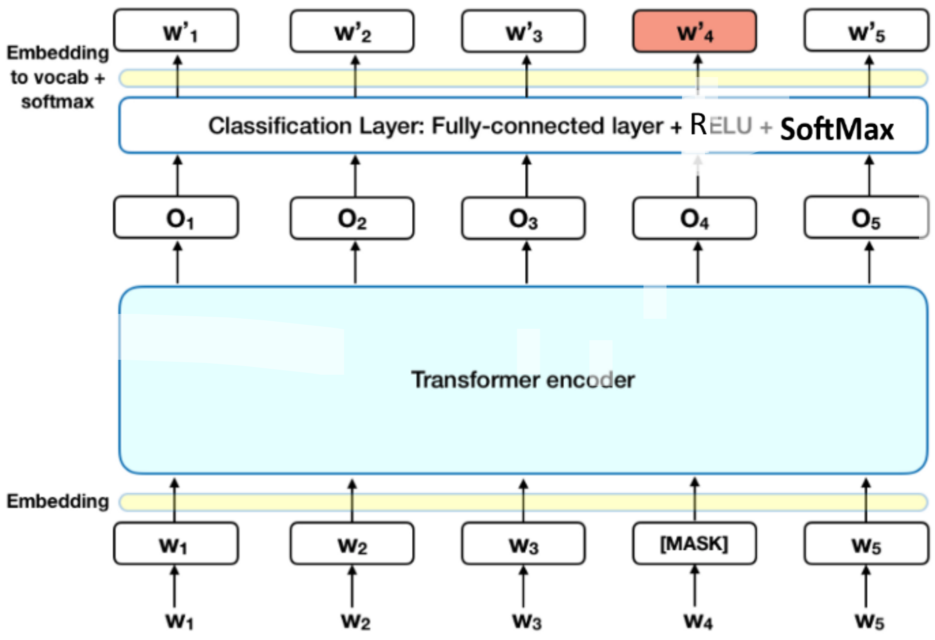

W'1
W'2
W'3
W'4
W'5
Embedding to vocab + softmax
Classification Layer: Fully-connected layer + RELU + SoftMax
O1
O2
O3
O4
O5
Transformer encoder
Embedding
W1
W2
W3
[MASK]
W5
W1
W2
W3
W4
W5

Fig 5. BERT Architecture visualization.

F. BERT Layers:

The core of the model consists of BERT layers, which are based on transformer architecture. BERT, or Bidirectional Encoder Representations from Transformers, is pre-trained on vast amounts of text data and has the ability to capture contextual information from both left and right contexts. These layers provide a deep understanding of the input text's semantics and context.

1. **Dropout Layers:**

   Dropout layers are introduced in the architecture to prevent overfitting. During training, dropout randomly sets a fraction of input units to zero, which helps prevent the model from relying too heavily on specific features or becoming overly complex. By mitigating overfitting, dropout layers enhance the model's generalization capability. They encourage the model to learn more robust features from the data.

2. **ReLU Activation Function:**

   Rectified Linear Unit (ReLU) is a significant activation function in neural networks. It replaces all negative values with zero while keeping positive values as they are. The ReLU activation function introduces non-linearity into the model, enabling it to learn complex patterns in the data. This non-linearity is crucial for the model to capture and represent complex relationships in the input data.

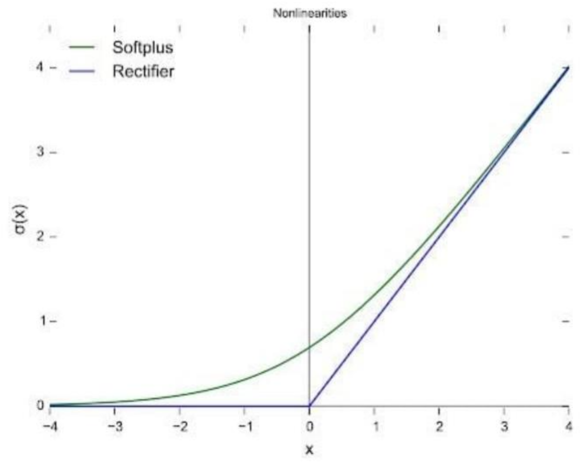


Fig 7. ReLU activation function visualization provided.

3. **Fully Connected Layers:**

   Fully connected layers are traditional neural network layers where each neuron is connected to every neuron in the adjacent layers. In this architecture, there are two fully connected layers. These layers act as intermediate representations that map the output from the BERT layers into a suitable form for classification. They allow the model to adapt to the specific classification task by learning the relationships between features and the target labels.

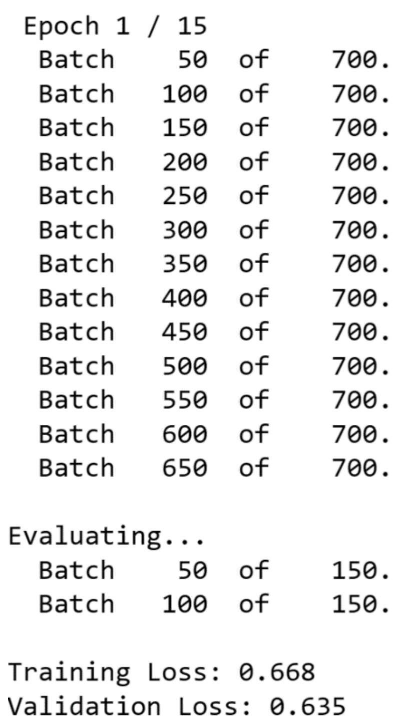
4. **Softmax Activation Function:**

   The softmax activation function is applied to the output of the final fully connected layer. It converts the raw model predictions into probability scores for each class. The class with the highest probability is chosen as the model's prediction. Softmax converts the model's output into a probability distribution, making it easier to interpret and use for classification. It helps in selecting the most likely class for a given input text.

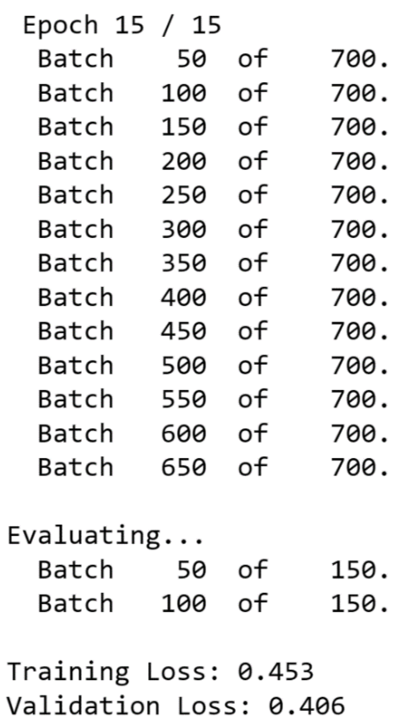
These architectural elements work in synergy to process and analyse the input data. The BERT layers provide deep semantic understanding of the text, while dropout layers mitigate overfitting. ReLU introduces non-linearity, fully connected layers adapt the model to the classification task, and softmax provides a probabilistic classification output. Each element contributes to the model's ability to detect hate speech in the Twitter data by extracting meaningful features, reducing overfitting, and making accurate predictions based on these features.

G. Training and Evaluation:

Model training was executed over a specified number of epochs. During training, a training function that conducted forward and backward passes to compute the gradients, refine the model's weights, and minimize the loss was used. Alongside this, an evaluation function was used to assess the model's performance and calculate the validation loss.



```
 Epoch 1 / 15
  Batch    50  of    700.
  Batch   100  of    700.
  Batch   150  of    700.
  Batch   200  of    700.
  Batch   250  of    700.
  Batch   300  of    700.
  Batch   350  of    700.
  Batch   400  of    700.
  Batch   450  of    700.
  Batch   500  of    700.
  Batch   550  of    700.
  Batch   600  of    700.
  Batch   650  of    700.

Evaluating...
  Batch    50  of    150.
  Batch   100  of    150.

Training Loss: 0.668
Validation Loss: 0.635
```

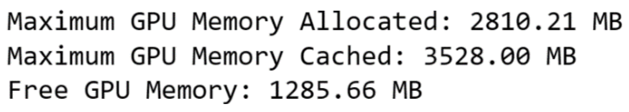


```
 Epoch 15 / 15
  Batch    50  of    700.
  Batch   100  of    700.
  Batch   150  of    700.
  Batch   200  of    700.
  Batch   250  of    700.
  Batch   300  of    700.
  Batch   350  of    700.
  Batch   400  of    700.
  Batch   450  of    700.
  Batch   500  of    700.
  Batch   550  of    700.
  Batch   600  of    700.
  Batch   650  of    700.

Evaluating...
  Batch    50  of    150.
  Batch   100  of    150.

Training Loss: 0.453
Validation Loss: 0.406
```

Fig 8 & 9. Training process of BERT. Difference between first and last epoch.

H. Limitations:

The use of GPU memory was monitored, acknowledging GPU constraints. Due to limited GPU memory, the testing dataset was restricted to a smaller subset for model evaluation. Testing dataset consisted of 4790 tweet that needed at least 1780 MB. However, the available GPU after training was 1285 MB. An attempt to free the GPU allocated spaces was made using torch.cuda.empty_cache(). However, it didn't achieve the requirement.

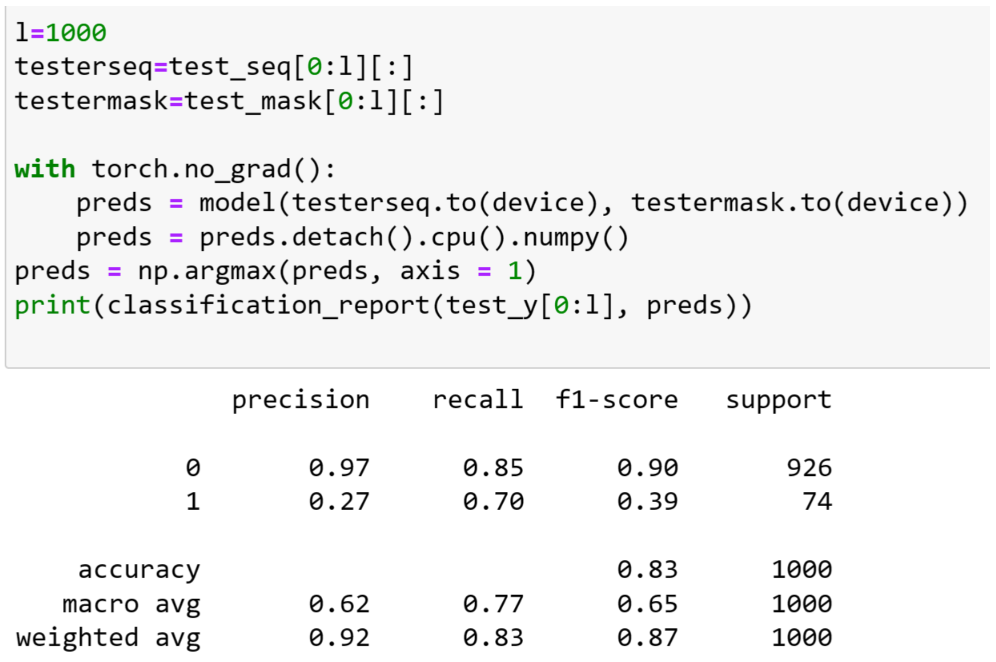


```
Maximum GPU Memory Allocated: 2810.21 MB
Maximum GPU Memory Cached: 3528.00 MB
Free GPU Memory: 1285.66 MB
```

Fig 10. GPU availability.

I. Results and Discussion:

The results of the model performance, including accuracy, precision, recall, and F1-score, are presented. The discussion section offers a thorough analysis of the results, highlighting the model's strengths and limitations, as well as their implications.



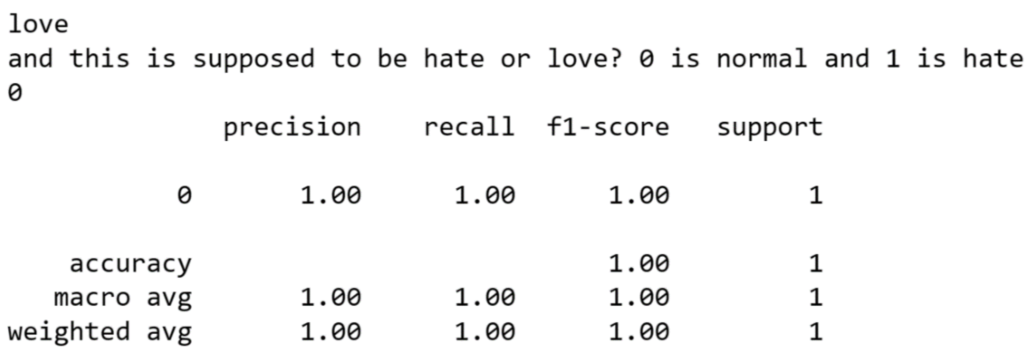
```
l=1000
testerseq=test_seq[0:l][:]
testermask=test_mask[0:l][:]

with torch.no_grad():
    preds = model(testerseq.to(device), testermask.to(device))
    preds = preds.detach().cpu().numpy()
preds = np.argmax(preds, axis = 1)
print(classification_report(test_y[0:l], preds))
```

```
              precision    recall  f1-score   support

           0       0.97      0.85      0.90       926
           1       0.27      0.70      0.39        74

    accuracy                           0.83      1000
   macro avg       0.62      0.77      0.65      1000
weighted avg       0.92      0.83      0.87      1000
```

Fig 11. Testing dataset result.

J. User Input Testing:

To demonstrate the practical application of the model, a simple user interface test was implemented, allowing users to input sentences for classification. The testing methodology involved submitting user inputs to the model for hate speech detection, and the model's predictions were evaluated against performance metrics.



```
love
and this is supposed to be hate or love? 0 is normal and 1 is hate
0
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         1

    accuracy                           1.00         1
   macro avg       1.00      1.00      1.00         1
weighted avg       1.00      1.00      1.00         1
```

Fig 12. User Testing through inputting a word to identify its type.

K. Suggestions for improvement:

- Increase Training Duration (More Epochs): Training a deep learning model like BERT for more epochs can often lead to better performance. It allows the model to refine its parameters and learn from the data for a more extended period. However, this is usually dependent on having adequate computational resources.

- Larger GPU: Upgrading to a more powerful GPU with more VRAM can enable you to work with larger batch sizes and more extensive datasets. This can speed up training and allow for more complex model architectures.

- Diverse Dataset: A diverse and representative dataset is crucial. To improve the model's performance and generalization, you can consider collecting or sourcing a more extensive and diverse dataset with a broader range of hate speech types. This can help the model learn a wider variety of hate speech patterns.

- Data alterations (word variations): performing word alterations in the dataset by creating variations of existing samples (e.g., through synonym replacement, paraphrasing, or introducing typos) can help the model become more robust and generalize better. Data augmentation techniques can help mitigate issues stemming from dataset size limitations.

- Advanced Preprocessing and Cleaning: Enhancing the data cleaning and preprocessing steps. For instance, it is believed that exploring advanced techniques for handling things like misspellings, slang, or noisy text data may yield better results of data cleaning. Advanced natural language processing (NLP) techniques, like spell-checkers and lemmatization, can further improve text quality.

- Ensemble Models: it is possible to consider creating an ensemble of models. Combining the predictions of multiple models, each with different architectures or hyperparameters, often leads to improved performance and model robustness.

- Model Tuning: Fine-tuning the model is an essential aspect of improving performance. Experiment with different hyperparameters, such as learning rates, batch sizes, and dropout rates. it is also possible to consider tuning BERT-specific hyperparameters.

L. Conclusion:

In conclusion, this report summarizes the findings and contributions of the study. It underscores the potential for hate speech detection using deep learning models and offers suggestions for future research and enhancements.