Data Glacier

Your Deep Learning Partner

NLP Project: Hate speech Detection

12/30/2024

Data Glacier

Your Deep Learning Partner

Team Members

	Name	Email
1	Keilor Fallas Prado	kfallasprado@gmail.com
2	Ky Quang Dang	Keith.dang1610@gmail.com

Agenda

Executive Summary

Problem Statement

Approach

Results and Evaluation

Recommendations

Data Glacier

Your Deep Learning Partner

Executive Summary

Problem Statement:

Detecting and addressing hate speech is a pressing concern to maintain a healthy online ecosystem, protect users from harm, and ensure regulatory compliance.

Business understanding:

Developing a hate speech detection model isn't just a technical challenge but also an effort to address real-world problems.

Deliveries:

The NLP model can detect hate speech and non-hate speech.

The report

Executive Summary

Approach:

- **Data:** train shape (31962, 3) test shape(17197, 2), features (ids, tweet, labels)
- **Methodology:** Transformer model (BERT, distilBERT)
- **Tools and Techniques:** Mention any NLP techniques, libraries, or frameworks used (e.g., DistilBERT, emoji library, tokenization).

Model training and Evaluation:

- **Model Selection:** Using **DistilBERT (lighter than BERT)** for efficient resources consumption.
- **Training Process:** Trained model can achieve the high accuracy over 90% and loss is decreased after every epoch

Problem Statement

Hate speech is a form of communication, whether verbal, written, or behavioral, that attacks or discriminates against an individual or group based on their inherent characteristics, such as religion, ethnicity, nationality, race, gender, or other identity factors. The emergence of hate speech on social media platforms like Twitter poses significant challenges, including creating a toxic environment for users and impacting the platform's reputation.

Detecting and addressing hate speech is a pressing concern to maintain a healthy online ecosystem, protect users from harm, and ensure regulatory compliance.

Data Glacier

Approach

Data overview: Dataset consists of two subsets: train and test.

- Train dataset: three features, including labels
- Test dataset: two columns, without labels

```
[5]  df_train.shape
...  (31962, 3)
```

```
[10] df_test.shape
...  (17197, 2)
```

Methodology:

- Traditional neural learning such as: LTSM, GRU…
- Transformer model, particularly pre-trained DistilBERT model.

Tools and Techniques:

- Using deep learning to train and fine-tune the model on the training dataset.
- Running model on Kaggle environment to leverage resources, high performance on training model, to save time every run time
- Libraries: DistilBERT, emoji, pandas, numpy, matplotlib, re..etc.

Data Preprocessing

Data cleansing:

- Remove mentions (e.g @user,..)
- Remove hashtags (e.g #)
- Remove special charaters

- Remove URLs
- Remove Punctuation & Numbers
- Convert to Lowercase: in this case of using BERT model, this step is unnecessary.

```
def clean_text(text):
    text = re.sub(r"http\S+", "", text) # remove URLs
    text = re.sub(r"@\w+", "", text) #remove mentions
    text = re.sub(r"#\w+", "", text) #remove hashtags
    #text = re.sub(r"[^\w\s]", "", text) #remove special characters
    return text.lower().strip()
[18]
```

Data Preprocessing

Data transforming:

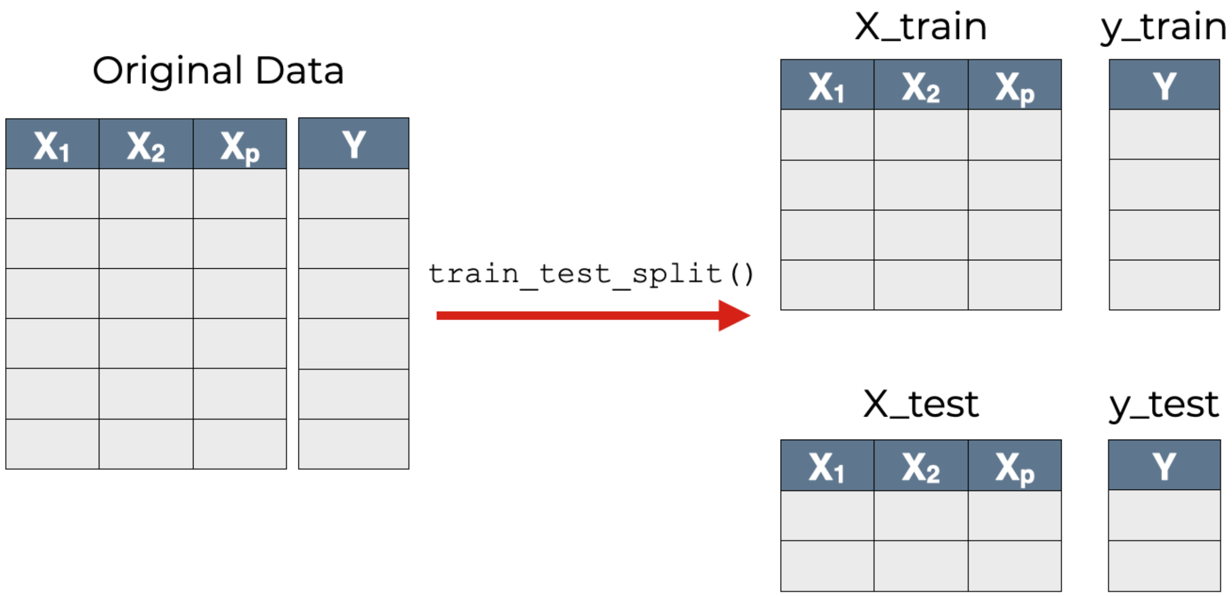
- Decode text for handling the characters’s error
- emoji-to-text conversion: **Retaining emojis** in tweets enhances the ability to accurately predict labels by preserving contextual meaning.

```
def decode_text(text):
    try:
        #Handling the characters's error
        return text.encode('latin1').decode('utf-8', errors='ignore')
    except UnicodeDecodeError:
        return text
[22]
```

Data Preprocessing

Split data:

- Split train dataset into two subsets train and test (with labels) consist of: X_train, y_train, X_test, y_test.
- Train subset will be used for training model and the other one for evaluation.



Data Preprocessing

Tokenization:

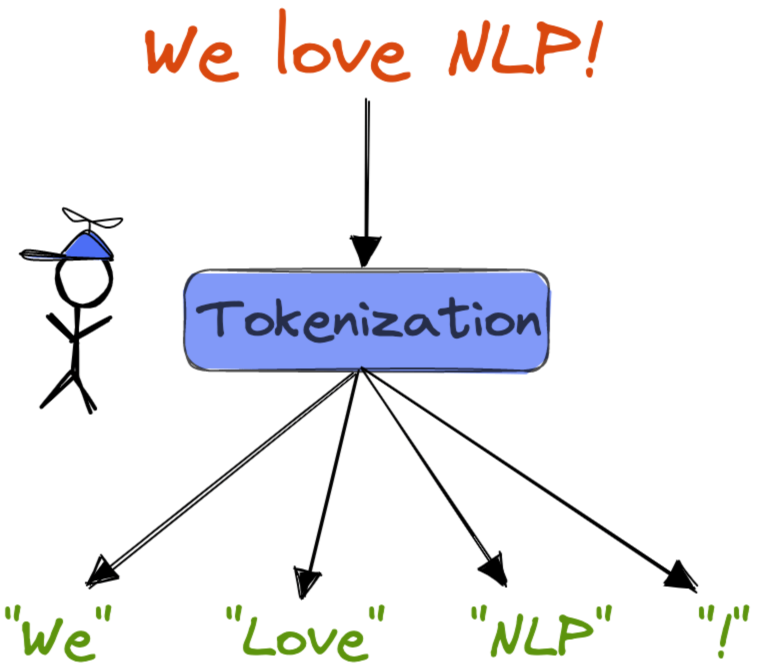
- Using DistilBertTokenizer of tranformers library to tokenize tweet texts.
- Check the max_length of tokenized texts by using:

```
lengths = [len(tokenizer.tokenize(tweet)) for tweet in new_train_df['deemoji_tweet']]

print(f“the max length: {max(lengths)}")

print(f“average length: {sum(lengths) / len(lengths)}")
```

- Using appropriate max_length to balance both efficiency and performance.

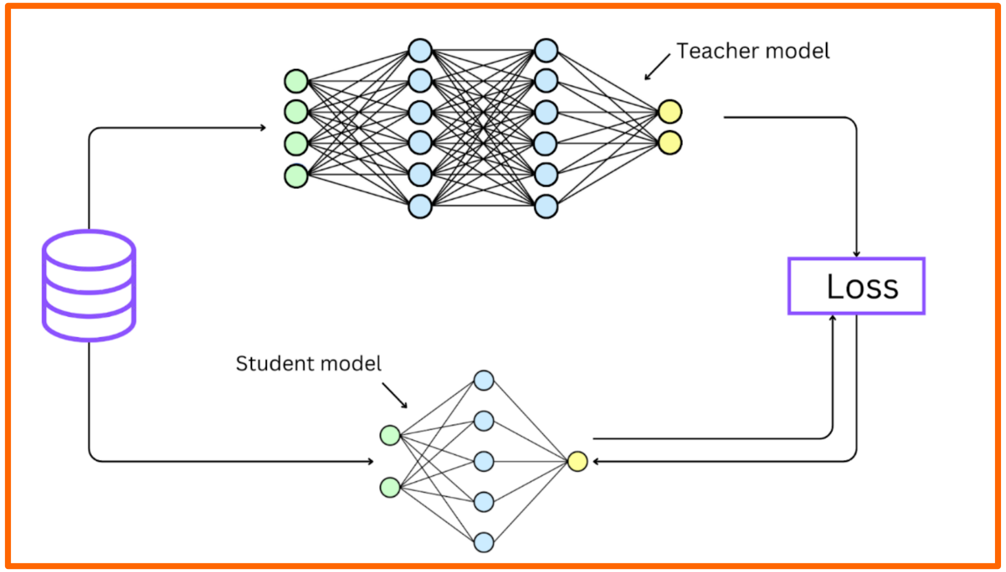


Data Glacier

Model Training and Evaluation

Model training:

- After tokenizing texts, we used these features of distilBERT model: DataLoader, TensorDataset, AdamW, epoch… to set up model training.
- The loss received in each epoch tends to decrease, which proves the model is improving its performance, the model's predictions are becoming more accurate




```
Epoch 1/5, Loss: 0.1662
Epoch 2/5, Loss: 0.0881
Epoch 3/5, Loss: 0.0410
Epoch 4/5, Loss: 0.0231
Epoch 5/5, Loss: 0.0172
```

Model Training and Evaluation

Model Evaluation:

- The overall accuracy of predictions on test subset and received the result of 87% along side with recall and f1-score.

Classification Report:

	precision	recall	f1-score	support
No Hate	0.93	0.93	0.93	5916
Hate	0.08	0.07	0.07	477
accuracy			0.87	6393
macro avg	0.50	0.50	0.50	6393
weighted avg	0.86	0.87	0.87	6393

- The accuracy of detect hate speech is quite low due to the imbalanced classes of two labels, even though we added the improve method

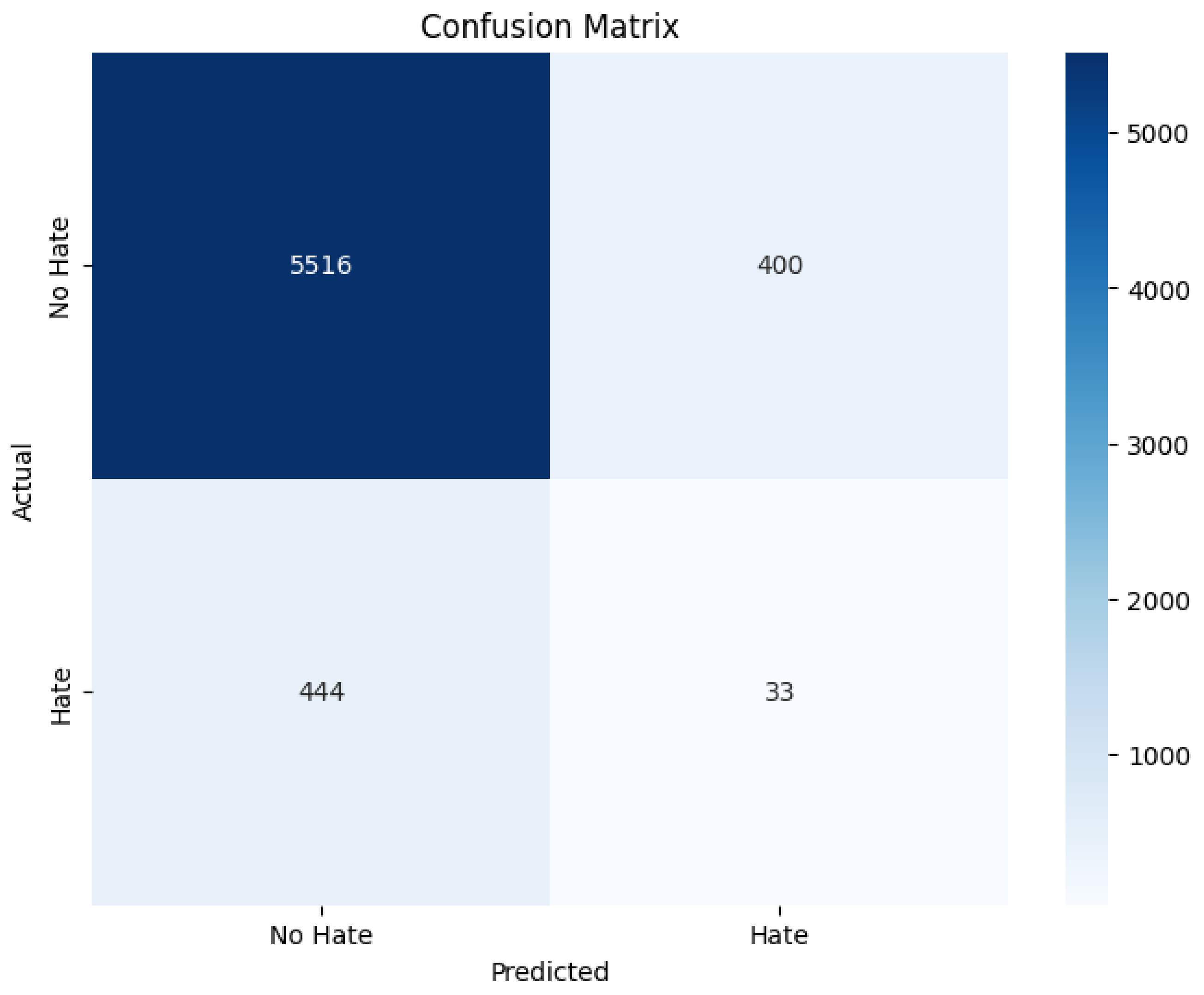
```
# Class Weighting to Handle Imbalance
class_weights = compute_class_weight('balanced', classes=np.unique(labels.numpy()), y=labels.numpy())
class_weights = torch.tensor(class_weights, dtype=torch.float).to(device)
criterion = torch.nn.CrossEntropyLoss(weight=class_weights)
optimizer = torch.optim.AdamW(model.parameters(), lr=5e-5)
```

Data Glacier

Model Training and Evaluation

Model Evaluation:

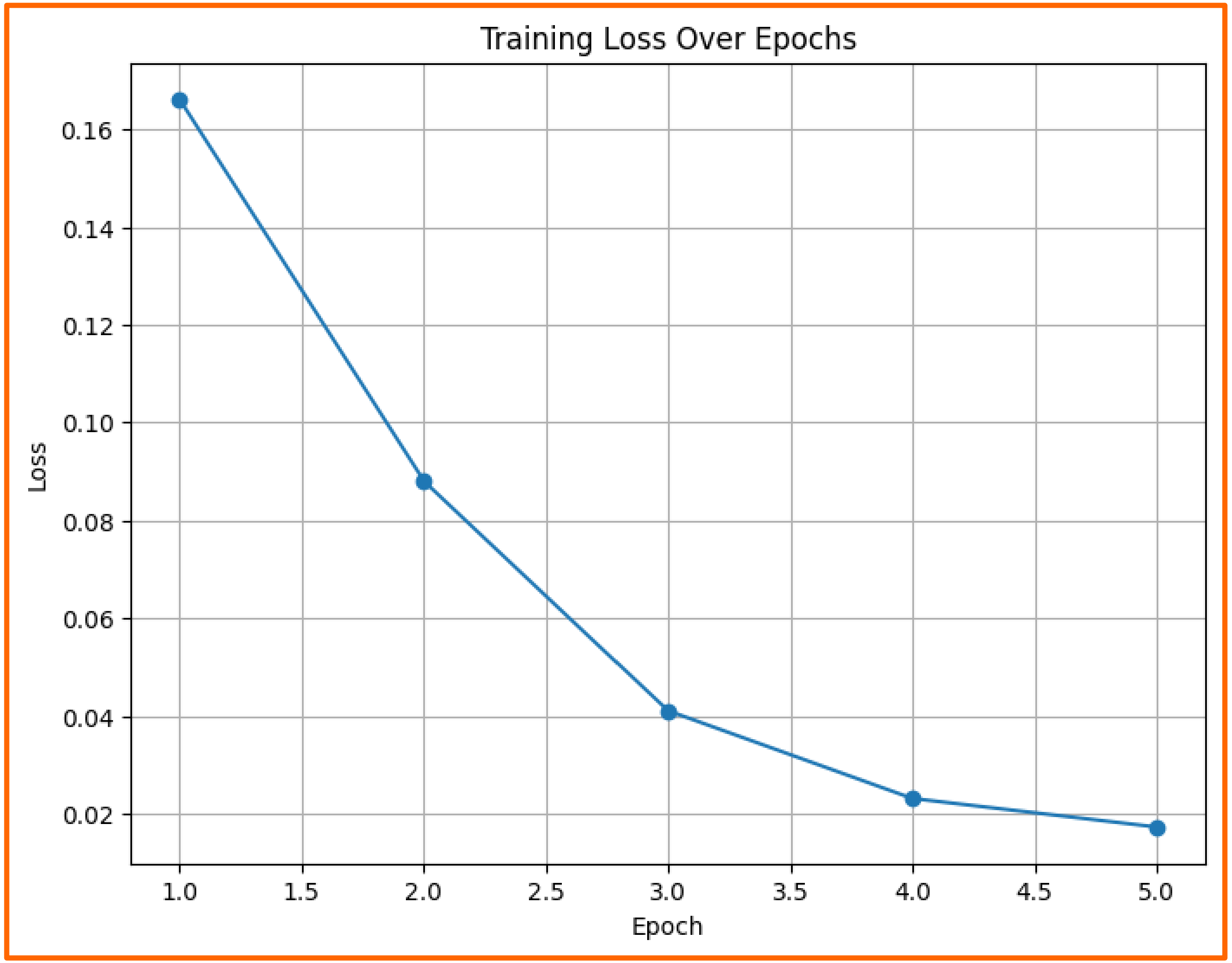
- This confusion matrix shows the big difference between True Positives (True non-hate) and True Negatives (True-hate)



Data Glacier

Model Training and Evaluation

Training Loss:



Data Glacier

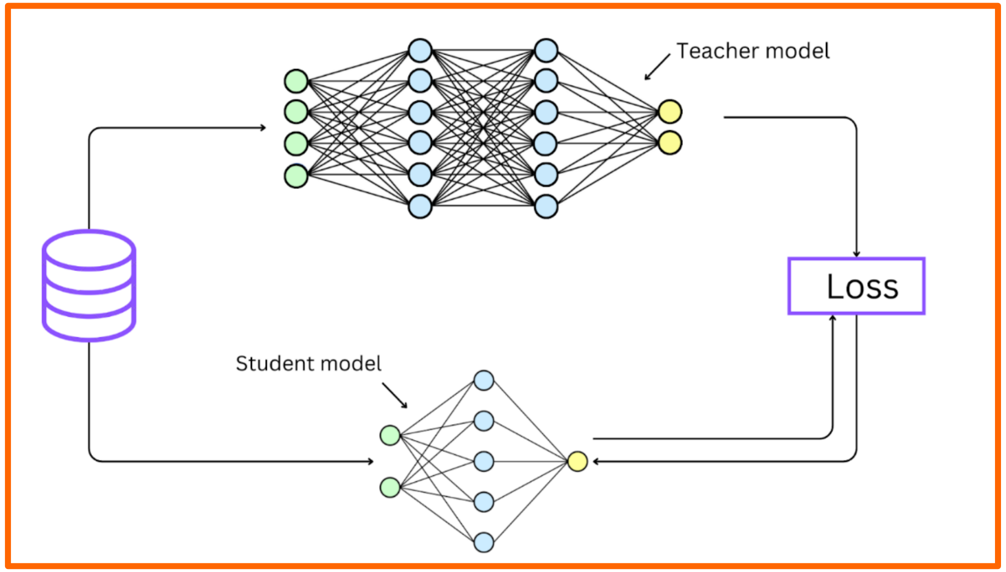
Model Improving and Reevaluation

Model training:

- We adjusted a several code lines to rerun the model
- The loss received in each epoch tends to decrease, which proves the model is improving its performance, the model’s predictions are becoming more accurate
- New notebook file here:

  Notebook_Link
- Compressed model link:

  NLP_final_model




```
...    Epoch 1/5, Loss: 0.3876
       Epoch 2/5, Loss: 0.2021
       Epoch 3/5, Loss: 0.1023
       Epoch 4/5, Loss: 0.0671
       Epoch 5/5, Loss: 0.0575
```

Model Improving and Reevaluation

Model Reevaluation:

- The overall accuracy of predictions on the test subset from train dataset (x_test, y_test) improved from 87% (with the previous model) to 94% (with the rerun model), along with increases in recall and F1-score

Classification Report:

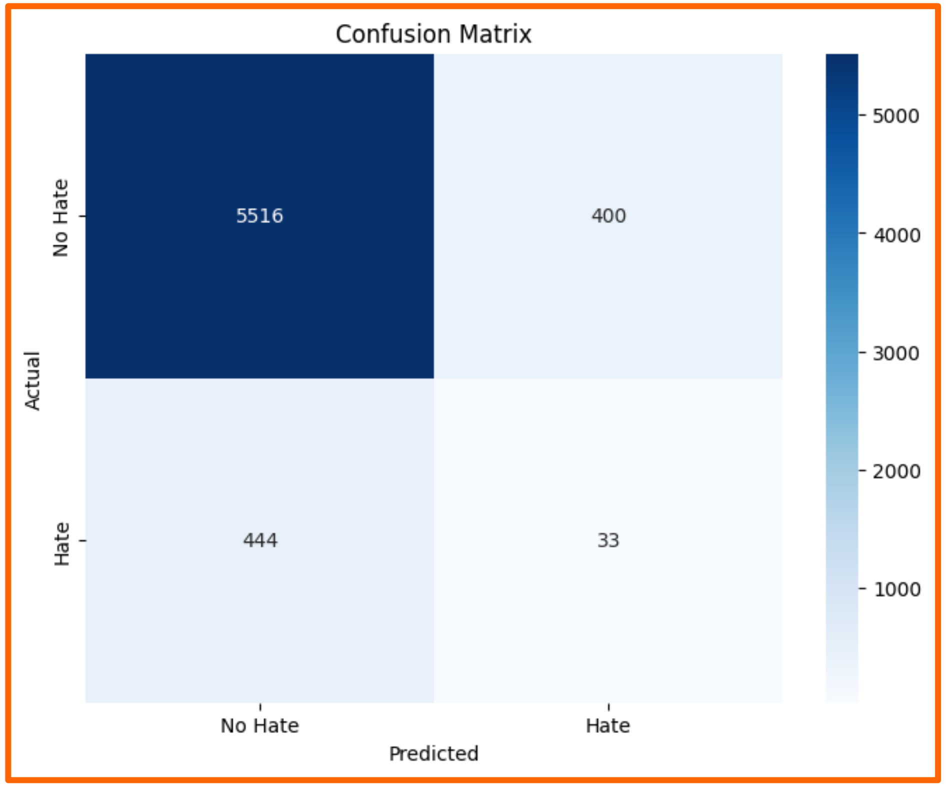
	precision	recall	f1-score	support
No Hate	0.98	0.96	0.97	5945
Hate	0.59	0.67	0.63	448
accuracy			0.94	6393
macro avg	0.78	0.82	0.80	6393
weighted avg	0.95	0.94	0.95	6393

- The accuracy of detect hate speech significantly enhanced from 7% (with the previous model) to 59%.

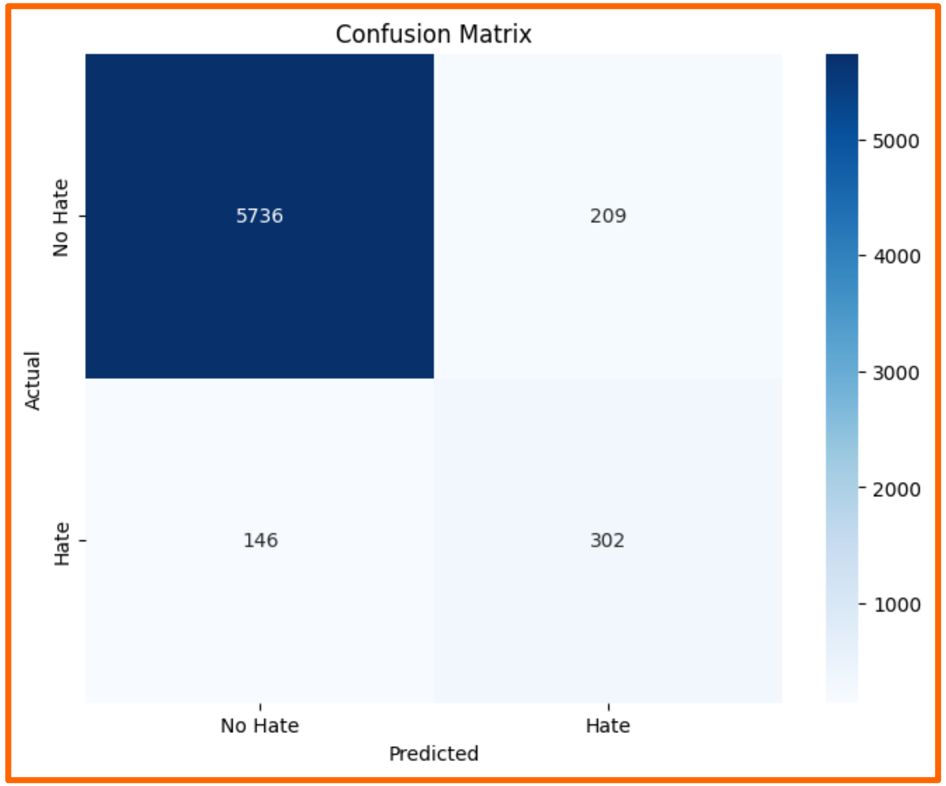
Model Training and Evaluation

Despite the difference in the two train-test splits, where the combinations of data were different, there is a noticeable improvement in accuracy.

Before

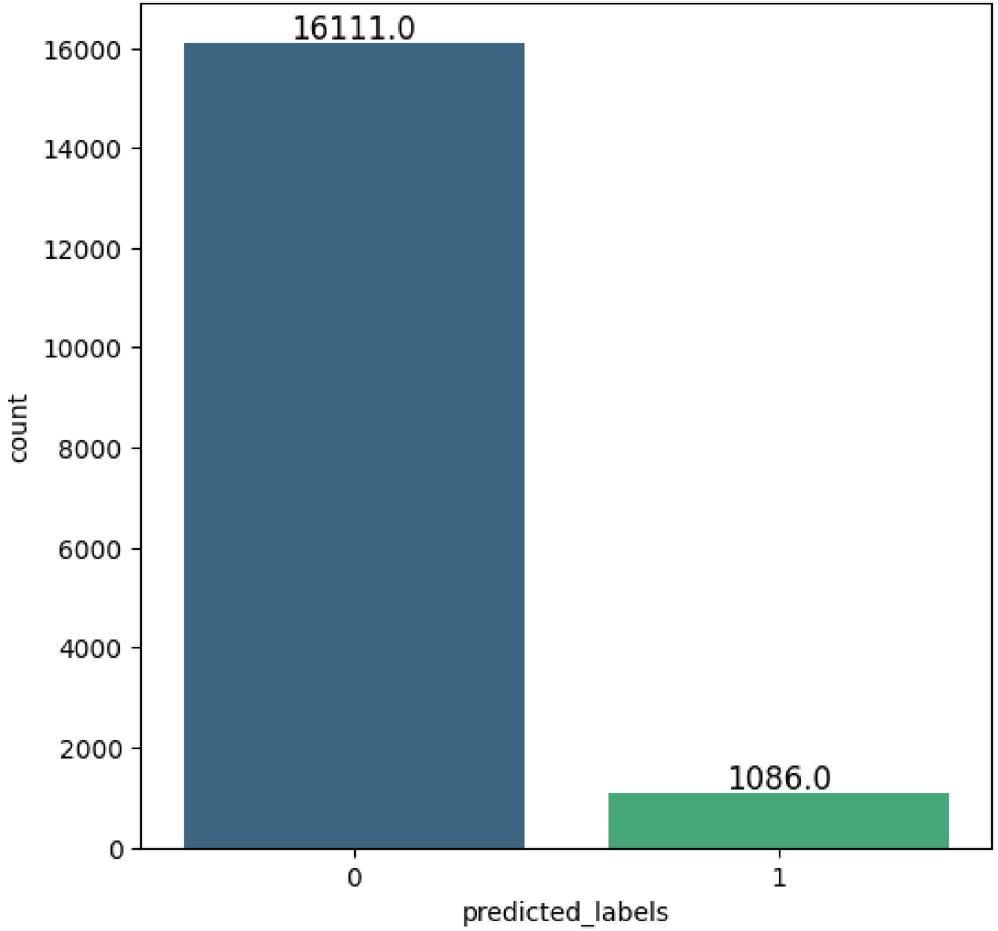


After

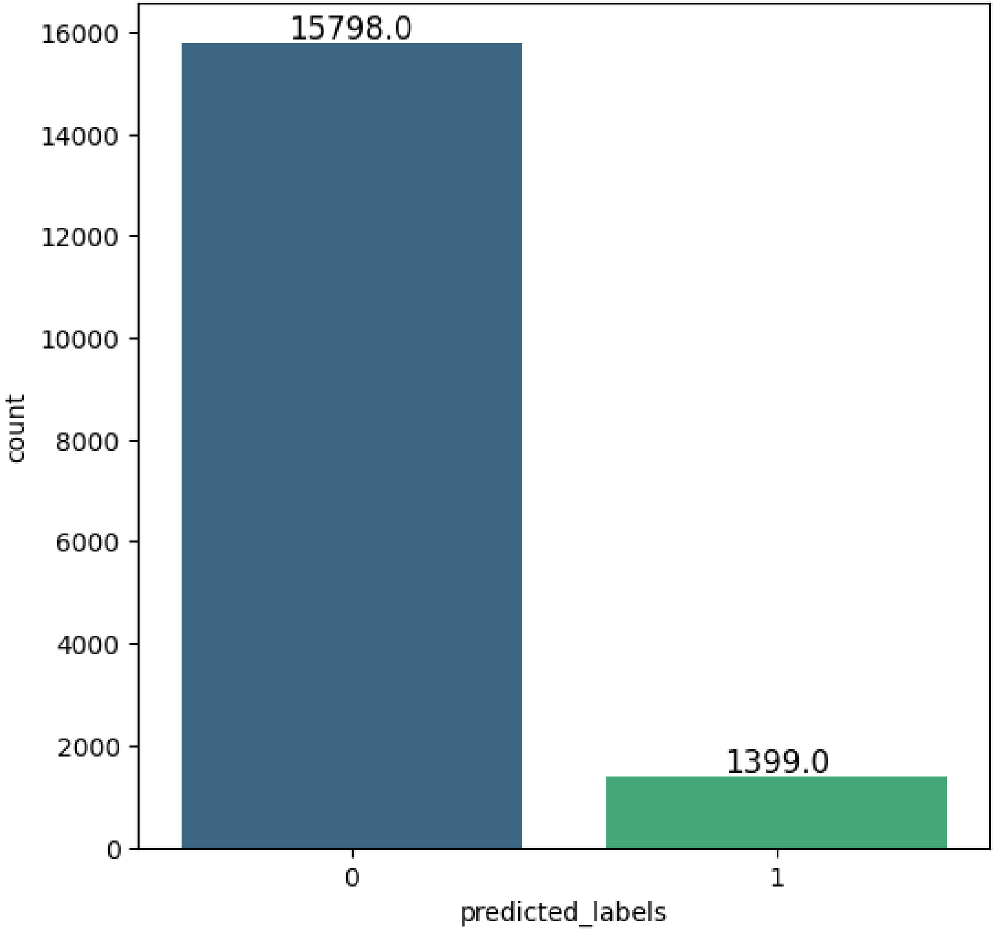


Test dataset Prediction

Before



After



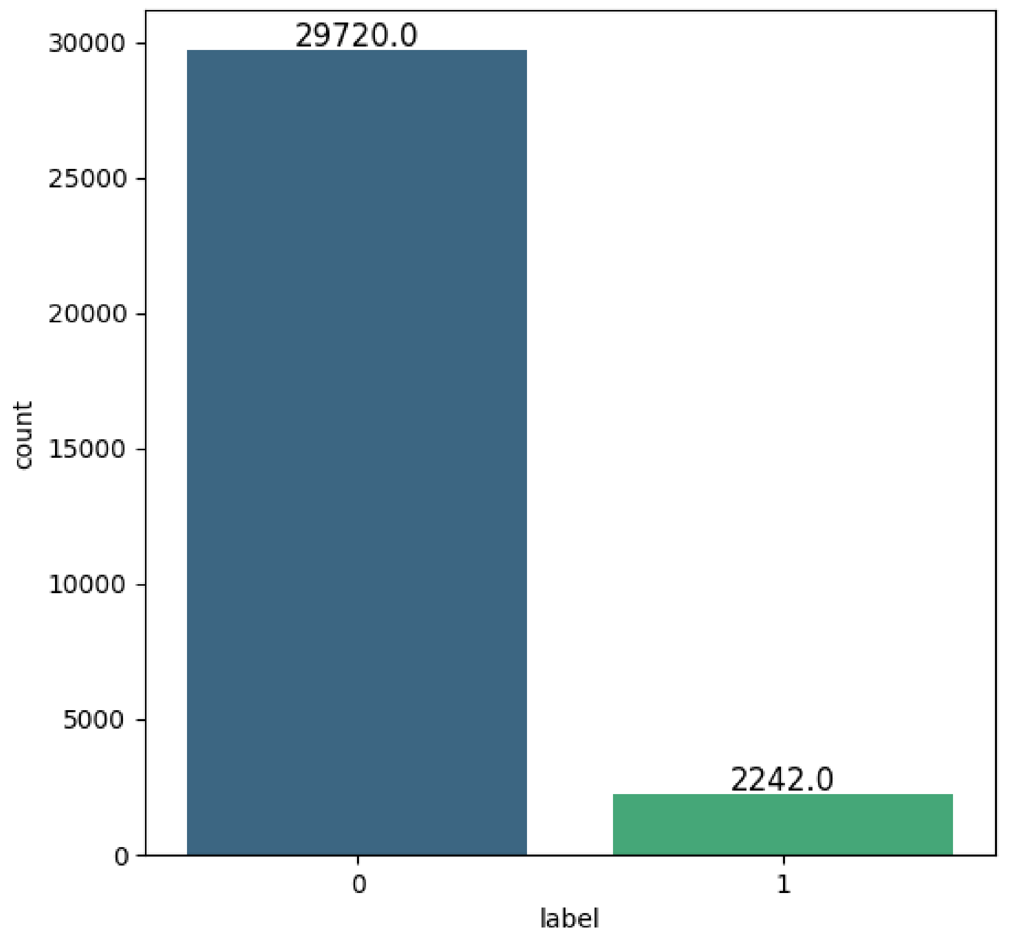
Data Glacier

Challenges and Recommendations

Challenges:

- The dataset has a significant imbalance between two classes
- This issue has a significant impact to the predictions and accuracy, especially in detecting hate-speech
- Inappropriate approach will not improve this problem.

Input Data (Train Subset: including X_train and X_test)



Data Glacier

Challenges and Recommendations

Recommendations:

- Although the accuracy of non-speech detection is considerably high and the improvement of hate detection's accuracy, but the main target is the hate speech detection. The model should be improved in several times to achieve the desired performance.
- The approach to address the imbalanced classes should be implemented before the train-test split.
- Additionally, other pre-trained models from platforms like Hugging Face, such as emojiBERT or emoBERT, which require more energy and resources than we can allocate, may be better suited for the context due to their specialized features.

Thank You

Data Glacier

Your Deep Learning Partner