Task 1.3 - Cellula Technologies NLP Internship
Toxic Topic Classification using DistilBERT (fine-tuned with LoRA)
Task Report

Abdelrahman Elaraby

September 2025

Contents

1 A short brief

There is no Exploratory Data Analysis (EDA) discussed in this report since I already did that in the LSTM one. Here I started working with the new dataset that contains the new generated rows. In Table 1 the number of samples per each toxic category is shown.

Toxic Category	**Count**
Safe	879
Violent Crimes	691
Unknown S-Type	284
Sex-Related Crimes	252
Child Sexual Exploitation	252
Suicide & Self-Harm	247
Elections	229
Non-Violent Crimes	205
unsafe	141

Table 1: Distribution of toxic categories in the dataset.

I used the Hugging-face transformers library for building the DistilBERT model. and I used the PEFT library to do the LoRA fine-tuning. You can find the model architecture in the last section of this report.

2 Learning Curves and Testing

Using LoRA you can witness the enormous reduction in the number of trainable parameters as seen below:
Trainable params: 744,969 **All params:** 67,705,362 **Trainable %:** 1.1003
As you can see it is only 1% of the total number of the BERT parameters. I fine-tuned this model using a very small learning rate $LR = 2 \times 10^{-5}$, and i trained it for 20 epochs with enabling the "save best model weights" feature that track the validation loss for knowing the best model. The learning curve is shown in Figure 1.

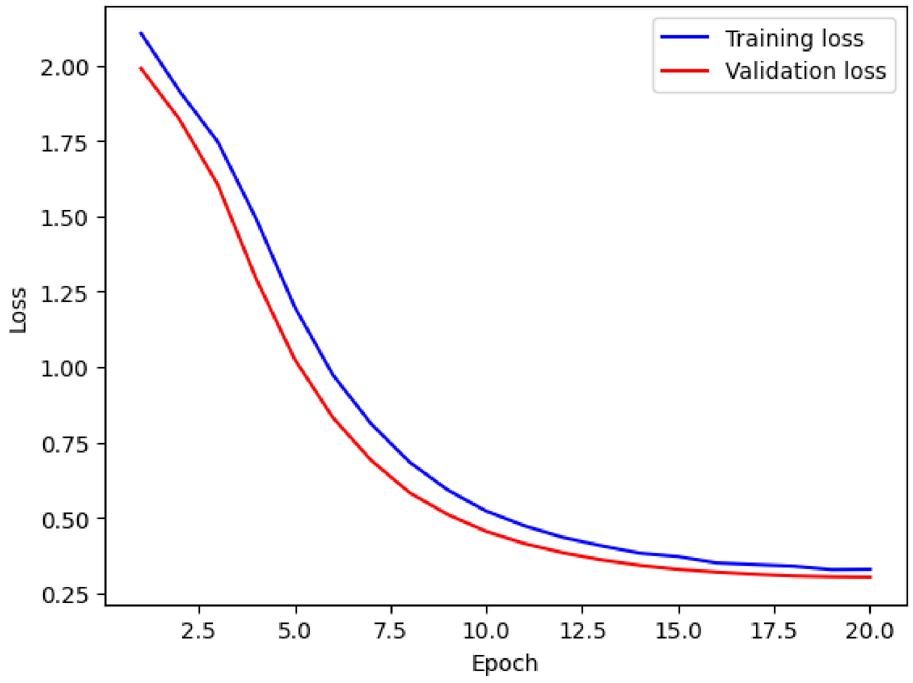

Figure 1: Learning curve of the model.

Then after training, I tested the model on the Test dataset, and printed the classification report which is shown in Table 2. Also the confusion matrix is shown in Figure 2

Category	**Precision**	**Recall**	**F1-score**	**Support**
Child Sexual Exploitation	1.00	1.00	1.00	51
Elections	1.00	1.00	1.00	45
Non-Violent Crimes	1.00	0.98	0.99	41
Safe	0.84	0.98	0.91	176
Sex-Related Crimes	0.98	0.98	0.98	51
Suicide & Self-Harm	0.91	0.98	0.94	49
Unknown S-Type	0.93	0.44	0.60	57
Violent Crimes	0.99	0.99	0.99	138
unsafe	1.00	0.93	0.96	28
Accuracy			0.93	636
Macro avg	0.96	0.92	0.93	636
Weighted avg	0.94	0.93	0.93	636

Table 2: Classification report showing precision, recall, F1-score, and support for each category.

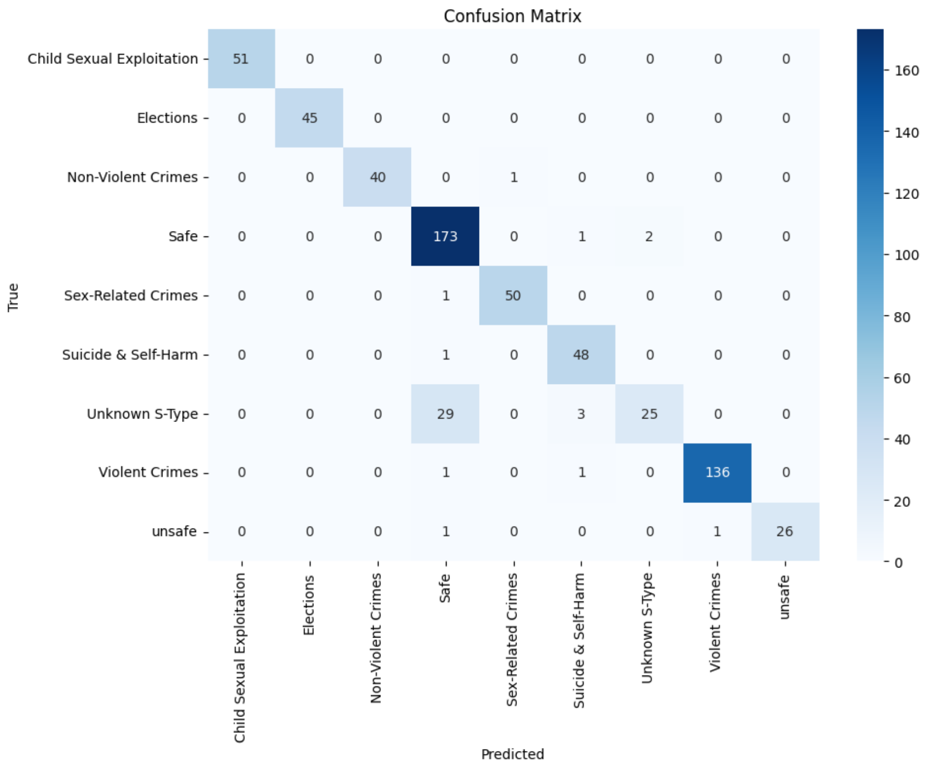

Figure 2: Confusion matrix of classification results.

3 Comments

The fine-tuned DistilBERT model has comparable accuracy to the LSTM results shown before; however it is a bit worse than the LSTM. A possible reason for that is that the transformer models - even the distilled small ones - are data hungry needing lots of data points to function well, and the dataset we have does not support that.

4 Full Model Architecture

```
PeftModelForSequenceClassification(
  (base_model): LoraModel(
    (model): DistilBertForSequenceClassification(
      (distilbert): DistilBertModel(
        (embeddings): Embeddings(
          (word_embeddings): Embedding(30522, 768, padding_idx=0)
          (position_embeddings): Embedding(512, 768)
          (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
          (dropout): Dropout(p=0.1, inplace=False)
        )
        (transformer): Transformer(
          (layer): ModuleList(
            (0-5): 6 x TransformerBlock(
              (attention): DistilBertSdpaAttention(
                (dropout): Dropout(p=0.1, inplace=False)
                (q_lin): lora.Linear(
                  (base_layer): Linear(in_features=768, out_features=768, bias=True)
                  (lora_dropout): ModuleDict(
                    (default): Dropout(p=0.1, inplace=False)
                  )
                  (lora_A): ModuleDict(
                    (default): Linear(in_features=768, out_features=8, bias=False)
                  )
                  (lora_B): ModuleDict(
                    (default): Linear(in_features=8, out_features=768, bias=False)
                  )
                  (lora_embedding_A): ParameterDict()
                  (lora_embedding_B): ParameterDict()
                  (lora_magnitude_vector): ModuleDict()
                )
                (k_lin): Linear(in_features=768, out_features=768, bias=True)
                (v_lin): lora.Linear(
                  (base_layer): Linear(in_features=768, out_features=768, bias=True)
                  (lora_dropout): ModuleDict(
                    (default): Dropout(p=0.1, inplace=False)
                  )
                  (lora_A): ModuleDict(
                    (default): Linear(in_features=768, out_features=8, bias=False)
                  )
                  (lora_B): ModuleDict(
                    (default): Linear(in_features=8, out_features=768, bias=False)
                  )
                  (lora_embedding_A): ParameterDict()
                  (lora_embedding_B): ParameterDict()
                  (lora_magnitude_vector): ModuleDict()
                )
                (out_lin): Linear(in_features=768, out_features=768, bias=True)
              )
              (sa_layer_norm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
              (ffn): FFN(
                (dropout): Dropout(p=0.1, inplace=False)
                (lin1): Linear(in_features=768, out_features=3072, bias=True)
                (lin2): Linear(in_features=3072, out_features=768, bias=True)
                (activation): GELUActivation()
              )
              (output_layer_norm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
            )
          )
        )
      )
      (pre_classifier): ModulesToSaveWrapper(
        (original_module): Linear(in_features=768, out_features=768, bias=True)
        (modules_to_save): ModuleDict(
          (default): Linear(in_features=768, out_features=768, bias=True)
        )
      )
      (classifier): ModulesToSaveWrapper(
        (original_module): Linear(in_features=768, out_features=9, bias=True)
        (modules_to_save): ModuleDict(
          (default): Linear(in_features=768, out_features=9, bias=True)
        )
      )
      (dropout): Dropout(p=0.2, inplace=False)
    )
  )
)
```