Execution Report: Text Classification Pipeline

Vanshita Gupta S

Problem Overview

Fine-tune a transformer-based model (DistilBERT) on the IMDB sentiment classification task, using Hugging Face's ecosystem and evaluating model performance on standard NLP metrics.

Dataset

- **Name**: IMDB Movie Review Dataset
- **Size**: 10,000 reviews (balanced positive and negative)
- **Source**: Hugging Face or manually uploaded in Colab

	A	B	C	D
1	review	sentiment		
2	One of the	positive		
3	A wonderf	positive		
4	I thought t	positive		
5	Basically tl	negative		
6	Petter Mat	positive		
7	Probably n	positive		
8	I sure wou	positive		
9	This show	negative		
10	Encourage	negative		
11	If you like	positive		
12	Phil the Ali	negative		
13	I saw this r	negative		
14	So im not a	negative		
15	The cast pl	negative		

Preprocessing Steps

- Converted CSV to Pandas
- Renamed columns: review → text, sentiment → label
- Mapped sentiment values to binary:
  - "positive" → 1
  - "negative" → 0

- Converted DataFrame to Hugging Face Dataset
- Tokenized with DistilBertTokenizerFast using padding and truncation

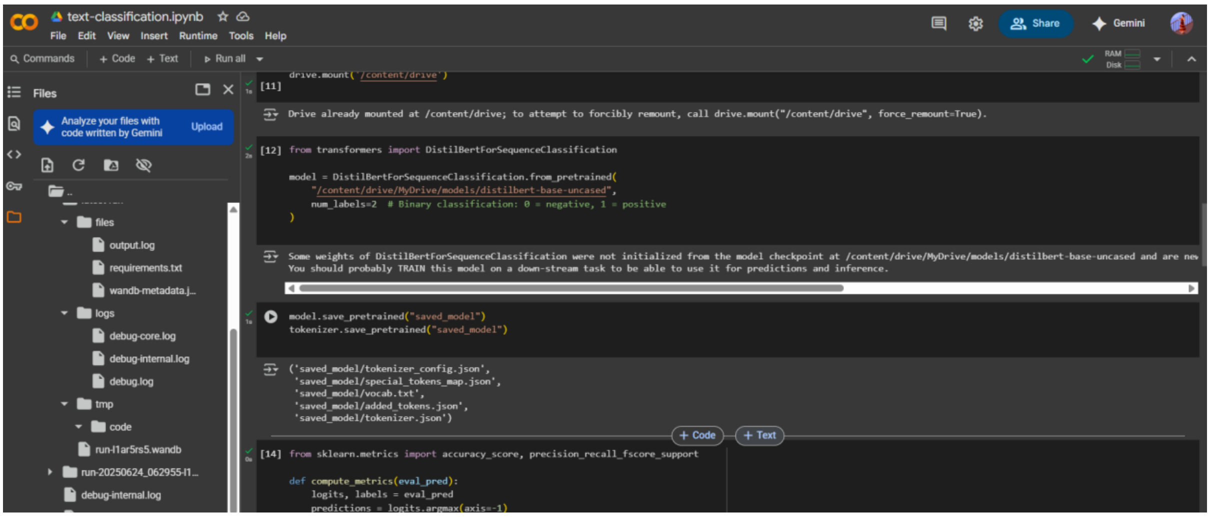
	text	label
0	One of the other reviewers has mentioned that ...	1
1	A wonderful little production. The...	1
2	I thought this was a wonderful way to spend ti...	1
3	Basically there's a family where a little boy ...	0
4	Petter Mattei's "Love in the Time of Money" is...	1

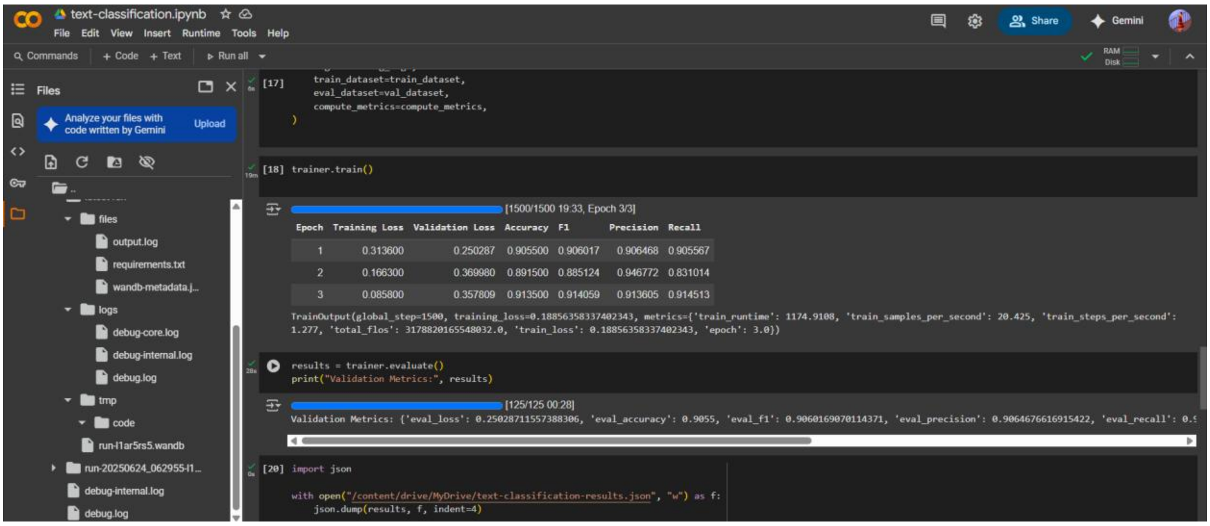
Model Setup

- **Base Model**: distilbert-base-uncased
- **Head**: Classification layer with num_labels=2
- Loaded using DistilBertForSequenceClassification
- Training managed using Trainer from transformers

TrainingArguments:

```
TrainingArguments(
    output_dir="./models/checkpoints",
    learning_rate=2e-5,
    per_device_train_batch_size=16,
    per_device_eval_batch_size=16,
    num_train_epochs=3,
    weight_decay=0.01,
    evaluation_strategy="epoch",
    save_strategy="epoch",
    logging_dir="./logs",
    load_best_model_at_end=True,
)
```

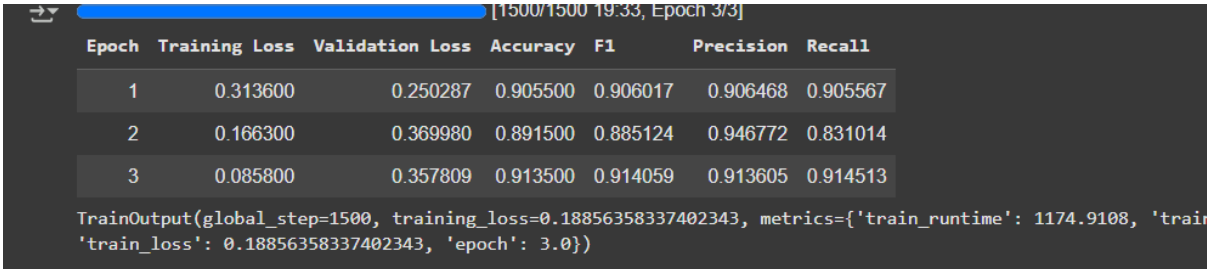




Evaluation Results

Used compute_metrics() function with:

- Accuracy
- F1 Score
- Precision
- Recall



[1500/1500 19:33, Epoch 3/3]

Epoch	Training Loss	Validation Loss	Accuracy	F1	Precision	Recall
1	0.313600	0.250287	0.905500	0.906017	0.906468	0.905567
2	0.166300	0.369980	0.891500	0.885124	0.946772	0.831014
3	0.085800	0.357809	0.913500	0.914059	0.913605	0.914513

TrainOutput(global_step=1500, training_loss=0.18856358337402343, metrics={'train_runtime': 1174.9108, 'trai
'train_loss': 0.18856358337402343, 'epoch': 3.0})

```
results = trainer.evaluate()
print("Validation Metrics:", results)
```

```
[125/125 00:28]
Validation Metrics: {'eval_loss': 0.25028711557388306, 'eval_accuracy': 0.9055, 'eval_f1': 0.9060169070114371, 'eval_precision': 0.9064676616915422, 'eval_recall': 0.9055666003976143, 'eval_runtime': 28.723
```

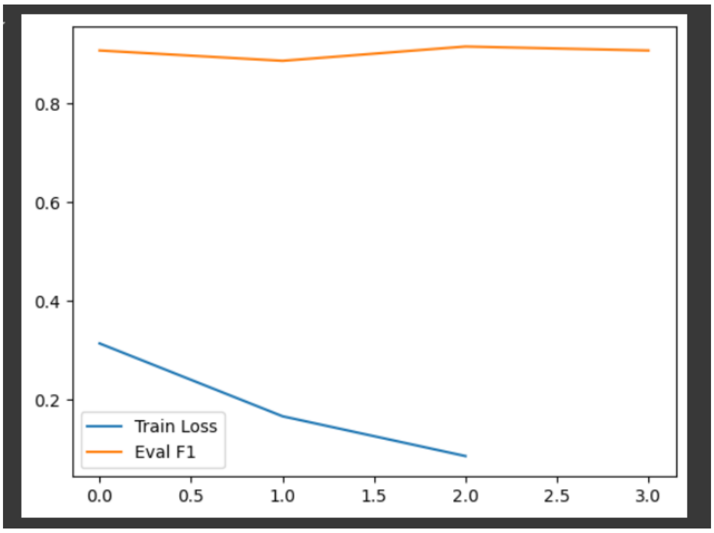
Training Loss vs. Evaluation F1 Score

This line graph visualizes the training dynamics over 3 epochs:

- The **blue line** represents the **training loss**, which steadily decreases from ~0.30 to below 0.10, indicating that the model is effectively learning the task.
- The **orange line** shows the **evaluation F1 score**, which remains consistently high (~0.90+) across all epochs with a slight peak at epoch 2, reflecting stable generalization performance on the validation set.

Insights:

- The divergence between training loss and evaluation F1 is minimal, indicating no overfitting.
- A consistently high F1 score suggests the model is balanced in handling both precision and recall on the binary classification task.



Predication for new text

```
[22] from transformers import pipeline

     # Load the classifier
     classifier = pipeline("text-classification", model=model, tokenizer=tokenizer)

     # Predict new texts
     classifier(["This movie was amazing!", "Worst film ever."])
```

```
Device set to use cuda:0
[{'label': 'LABEL_1', 'score': 0.9881706237792969},
 {'label': 'LABEL_0', 'score': 0.9758895635604858}]
```

Model Outputs

- Saved checkpoints at: checkpoint-500, checkpoint-1000, checkpoint-1500
- Final model saved in: models/trained_model/
- Tokenizer saved in: models/tokenizer/
- WandB run folder: wandb/run-20240624_xxxxxx/

Key Learnings

- Trainer abstracts most training complexities with minimal code
- Tokenizer and Dataset compatibility must be carefully managed
- Intermediate checkpoints are useful for recovery or comparative evaluation
- Logging via **wandb** enables real-time experiment tracking

---

Bonus Scope: Multilingual Extension

To enable multilingual support:

- Use a multilingual base model like xlm-roberta-base
- Change tokenizer accordingly
- Use a multilingual dataset (e.g., Amazon or Twitter multilingual sentiment)

Final Colab notebook, model outputs, metrics, and training logs are organized and documented in the repository.

All screenshots mentioned above have been captured and attached in the report folder or embedded in the markdown where supported.