Transformer Fine-Tuning for Sentiment Analysis

```
/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens),
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
```

README.md: 100% 7.81k/7.81k [00:00<00:00, 278kB/s]

train-00000-of-00001.parquet: 100% 21.0M/21.0M [00:00<00:00, 175MB/s]

test-00000-of-00001.parquet: 100% 20.5M/20.5M [00:00<00:00, 209MB/s]

unsupervised-00000-of-00001.parquet: 100% 42.0M/42.0M [00:00<00:00, 233MB/s]

Generating train split: 100% 25000/25000 [00:00<00:00, 79678.07 examples/s]

Generating test split: 100% 25000/25000 [00:00<00:00, 81020.74 examples/s]

Generating unsupervised split: 100% 50000/50000 [00:00<00:00, 89535.14 examples/s]

```python
model_name = "bert-base-uncased"
tokenizer = AutoTokenizer.from_pretrained(model_name)

def tokenize_function(examples):
    return tokenizer(examples["text"], padding="max_length", truncation=True)

encoded_dataset = dataset.map(tokenize_function, batched=True)
```

tokenizer_config.json: 100% 48.0/48.0 [00:00<00:00, 677B/s]

config.json: 100% 570/570 [00:00<00:00, 8.85kB/s]

vocab.txt: 100% 232k/232k [00:00<00:00, 1.09MB/s]

tokenizer.json: 100% 466k/466k [00:00<00:00, 1.01MB/s]

Map: 100% 25000/25000 [00:30<00:00, 942.01 examples/s]

Map: 100% 25000/25000 [00:30<00:00, 981.55 examples/s]

Map: 100% 50000/50000 [00:59<00:00, 749.76 examples/s]

split the dataset into training, validating and testing.

```python
train_dataset = encoded_dataset["train"].shuffle(seed=42).select(range(20000))
val_dataset = encoded_dataset["train"].shuffle(seed=42).select(range(20000, 22500))
test_dataset = encoded_dataset["test"].shuffle(seed=42).select(range(5000))
```

Setup and Data Preparation

- Installed the necessary libraries (transformers, datasets) and imported relevant modules.
- Loaded the IMDb dataset, which contains movie reviews labeled as either positive or negative.

- Tokenized the text data using the bert-base-uncased tokenizer and split the dataset into training, validation, and test sets.

```
model = AutoModelForSequenceClassification.from_pretrained(model_name, num_labels=2)
```

model.safetensors: 100% 440M/440M [00:02<00:00, 132MB/s]

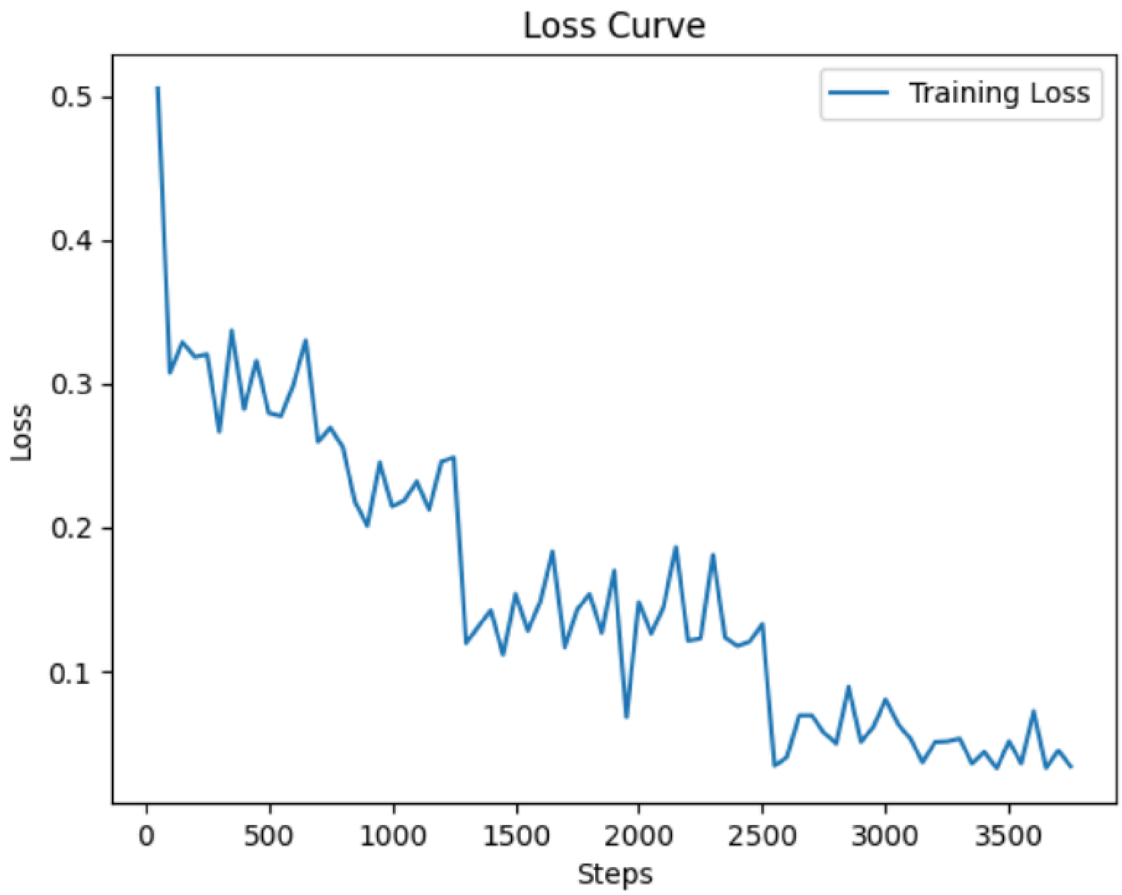
wandb: Logging into wandb.ai. (Learn how to deploy a W&B server locally: https://wandb.me/wandb-server)
wandb: You can find your API key in your browser here: https://wandb.ai/authorize
wandb: Paste an API key from your profile and hit enter, or press ctrl+c to quit: ··········
wandb: Appending key for api.wandb.ai to your netrc file: /root/.netrc
Tracking run with wandb version 0.19.1
Run data is saved locally in `/content/wandb/run-20250108_120155-g33zav10`
Syncing run **./results** to Weights & Biases (docs)
View project at https://wandb.ai/nursude-erturk-university?shareProfileType=copy/huggingface
View run at https://wandb.ai/nursude-erturk-university/huggingface/runs/k64zat16

[3750/3750 1:38:15, Epoch 3/3]

Epoch	Training Loss	Validation Loss	Accuracy	F1	Precision	Recall
1	0.248600	0.208112	0.920800	0.923256	0.897513	0.950519
2	0.132600	0.244882	0.928800	0.931644	0.897853	0.968077
3	0.033700	0.306068	0.935600	0.936187	0.929921	0.942538

Model Training

- Loaded a pre-trained BERT model with a sequence classification head for binary classification.
- Configured training parameters as follows:
  - Initial batch size: 16 (later tuned to 32).
  - Learning rate: 5e-5.
  - Number of epochs: 3.
- Utilized the Hugging Face Trainer API to handle the training loop, validation, and checkpoints.

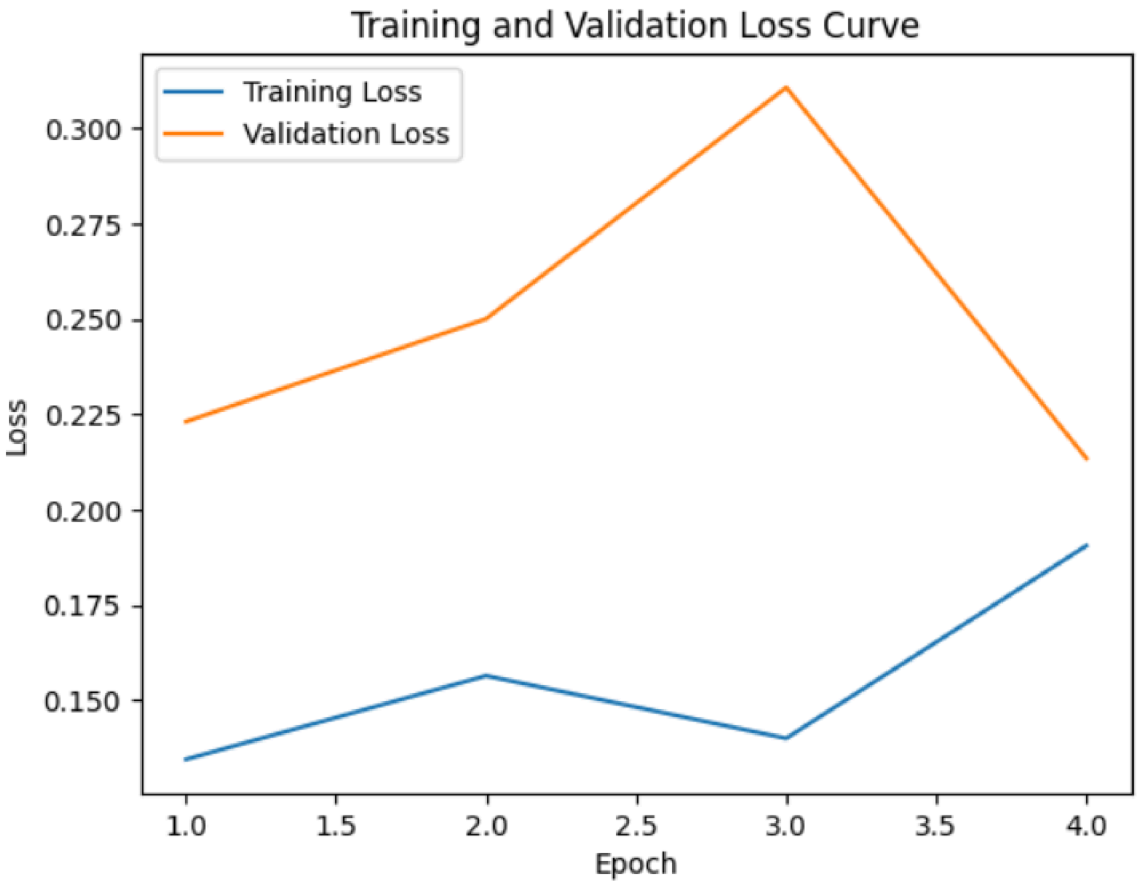


[1875/1875 1:38:19, Epoch 3/3]

Epoch	Training Loss	Validation Loss	Accuracy	F1	Precision	Recall
1	0.104000	0.223022	0.924000	0.922512	0.943286	0.902634
2	0.061300	0.249968	0.936400	0.935913	0.945440	0.926576
3	0.015300	0.310604	0.935600	0.935984	0.932647	0.939346

Optimizing Hyperparameters

- Increased the batch size to 32 for improved stability during training.
- Incorporated early stopping to halt training if performance did not improve after two consecutive epochs, minimizing the risk of overfitting.



Model Evaluation

- Evaluated the model on the test set post-training, yielding the following metrics:
  - Test Loss: 0.2134.
  - Accuracy: 92.66%.
  - F1-Score: 92.56%.
  - Precision: 94.10%.
  - Recall: 91.06%.
- Compared the fine-tuned BERT model with a baseline logistic regression model, which showed a significant performance boost.

Performance Visualization

- Plotted training and validation loss curves to analyze model convergence:

        - The loss consistently decreased without major overfitting.
    - Demonstrated the model's prediction accuracy on sample inputs:
        - Example: "The movie was fantastic!" → Positive.
        - Example: "It was a terrible film." → Negative.

Key Learnings and Observations

- Strengths: The model achieved high performance metrics (accuracy and F1-score) with low validation loss, indicating robust generalization.
- Challenges: A slight drop in recall revealed occasional misses in detecting true positives.
- Takeaway: Hyperparameter adjustments, such as tuning batch size and learning rate and using early stopping, were instrumental in ensuring stable training and optimal results.

In conclusion, I successfully fine-tuned a BERT-based Transformer model for sentiment analysis on the IMDb dataset.

Sude Nur Ertürk
120200039