Comparative Analysis of BERT and FinBERT for Financial Sentiment Classification

Sajidur Rahman Sajid
CSE
American International University-Bangladesh
Dhaka,Bangladesh
22-49076-3@student.aiub.edu

Badhan Ghosh
CSE
American International University-Bangladesh
Dhaka,Bangladesh
22-48395-3@student.aiub.edu

Sidratul Muntaha
CSE
American International University-Bangladesh
Dhaka,Bangladesh
22-48557-3@student.aiub.edu

***Abstract*— This project focuses on financial sentiment analysis using two transformer-based language models, BERT and FinBERT. The primary objective is to examine how domain-specific pretraining affects sentiment classification accuracy in financial texts. Financial sentiment analysis is an essential task in natural language processing (NLP) because investor opinions, news articles, and market reports directly influence financial decisions and stock performance.**

To solve this problem, both BERT and FinBERT were fine-tuned and evaluated on Kaggle datasets that are similar to the Financial PhraseBank and FiQA corpora. The datasets contain financial statements labeled into three sentiment categories: positive, neutral, and negative. The models were trained under identical configurations — epochs = 3, learning rate = 2e-5, and max_length = 128 — to ensure a fair comparison.

The experimental results demonstrate that FinBERT consistently achieved higher accuracy and macro-F1 scores than BERT in both validation and test sets. This improvement confirms that domain-specific pretraining significantly enhances model performance in understanding complex financial language and context. Consequently, FinBERT proves to be a more robust and effective model for financial sentiment classification compared to general-purpose BERT.

Keywords*— *financial sentiment analysis, BERT, FinBERT, natural language processing, transformer models

I. Introduction

Financial sentiment analysis has become an essential research area in Natural Language Processing (NLP) because opinions expressed in financial news, reports, and analyst statements have a direct impact on market trends, investor confidence, and trading behavior. The accurate identification of sentiment—positive, neutral, or negative—within such text is critical for making informed financial and business decisions. Traditional machine-learning models depend heavily on handcrafted features and simple text representations, which often fail to capture the complex linguistic structures and domain-specific vocabulary used in financial contexts. To overcome these limitations, transformer-based models such as BERT (Bidirectional Encoder Representations from Transformers) have introduced a deep contextual understanding of language through attention mechanisms that process words bidirectionally.

However, general BERT models are pre-trained on large general-domain corpora like Wikipedia and BookCorpus, which limits their ability to interpret financial terminology and nuanced tone. In contrast, FinBERT is a domain-specific adaptation of BERT that has been further pre-trained on large-scale financial texts such as company reports and financial news, allowing it to better understand the subtleties of financial sentiment.

This project focuses on a comparative study between BERT and FinBERT for financial sentiment classification. Both models were fine-tuned and evaluated on Kaggle datasets that resemble the Financial PhraseBank and FiQA datasets, covering three sentiment categories: positive, neutral, and negative. The primary goal is to analyze the effect of domain-specific pretraining on model performance. Through this comparison, the project aims to demonstrate how FinBERT's specialized training significantly improves accuracy and overall understanding in financial sentiment analysis tasks.

II. Implementation

This section presents the detailed implementation of the system, including the experimental environment, dataset preprocessing, and fine-tuning procedures for both BERT and FinBERT models. The goal was to ensure a fair comparison between a general-domain model (BERT) and a domain-specific model (FinBERT) under identical configurations.

A. Programming Environment Programming Environment

All experiments were conducted using Google Colab, a cloud-based Python environment with GPU acceleration.

The configuration details are as follows:

- Operating platform: Google Colab (Linux-based cloud GPU)
- Programming language: Python 3.10
- GPU: NVIDIA Tesla T4 (16 GB VRAM)
- Deep learning framework: PyTorch (v2.3.0)
- Libraries used:
    - Transformers (v4.41) – model loading, tokenization, fine-tuning
    - Datasets (v2.20) – dataset management and splitting

- Scikit-learn (v1.5) – evaluation metrics, confusion matrix
- Accelerate, Pandas, Numpy, Matplotlib – computation and visualization

A fixed random seed (42) was used across all runs to maintain reproducibility.

B. Dataset and Preprocessing

Two Kaggle-based datasets were used to replicate the Financial PhraseBank and FiQA domains:

1. Financial PhraseBank-like dataset (all-data.csv) – financial news statements labeled as *positive*, *neutral*, or *negative*.
2. FPB + FiQA mixed dataset (data.csv) – a broader dataset containing market sentiment and financial reports.

Pre-processing Steps:

- Removed missing entries and duplicates.
- Normalized text to lowercase and trimmed whitespaces.
- Mapped labels to numeric classes: negative = 0, neutral = 1, positive = 2.
- Renamed text fields consistently across datasets.

Splitting & Tokenization:

- Data split into 80% training, 10% validation, and 10% test (stratified sampling).
- Tokenization used the respective model's tokenizer with a maximum sequence length of 128 tokens.
- Dynamic padding was applied using DataCollatorWithPadding.

C. Training Configuration

Both BERT and FinBERT were fine-tuned under identical hyperparameters to ensure a controlled and unbiased comparison.
The complete set of hyperparameters used for both models is summarized in **Table I**.
Models were trained using the Hugging Face Trainer API, which automated training loops, logging, and evaluation.
Checkpoints were saved at the end of each epoch, and the best-performing model was selected based on validation accuracy.
Metrics such as Accuracy and Macro-F1 score were computed for both validation and test sets.

TABLE I. TRAINING CONFIGURATION AND PARAMETERS

Parameter	Value
Learning Rate	2×10^{-5}
Epochs	3
Batch Size	16
Optimizer	AdamW
Loss Function	Cross-Entropy
Max Sequence Length	128
Evaluation Strategy	Per Epoch
Metric for Best Model	Validation Accuracy
Metrics Used	Accuracy, Macro-F1

D. BERT Implementation

The BERT-base-uncased model was selected as the baseline. It is a 12-layer Transformer encoder model with 768 hidden units and 12 attention heads, totaling around 110 million parameters.
The WordPiece tokenizer was used for tokenization, converting each sentence into subword tokens.

Implementation Steps:

1. Loaded the model using AutoModelForSequenceClassification with num_labels=3.
2. Tokenized data with AutoTokenizer (BERT tokenizer).
3. Trained on both datasets for three epochs using the configuration from Table I.
4. Evaluated using validation and test sets to calculate Accuracy and Macro-F1.

Purpose:
The BERT model serves as a general-domain benchmark, representing models not specialized for financial text.

E. FinBERT Implementation

The FinBERT model (checkpoint: ProsusAI/finbert) was fine-tuned as the second model.
FinBERT uses the same architecture as BERT but is pre-trained on financial news, reports, and earnings calls, making it better suited for domain-specific sentiment detection.

Model Details:

- Architecture: 12-layer Transformer (110M parameters)
- Tokenizer: Domain-specific WordPiece vocabulary
- Objective: 3-class sentiment classification (*positive, neutral, negative*)

Implementation Steps:

1. Loaded ProsusAI/finbert using the Transformers library.

2. Tokenized text using FinBERT's tokenizer to preserve financial context.
3. Fine-tuned using the same hyperparameters and Trainer setup as BERT.
4. Monitored training loss and validation metrics across epochs.

Purpose:
FinBERT tests the effectiveness of domain-specific pretraining, highlighting how specialization improves contextual understanding in financial text sentiment classification.

F. *Workflow Overview*

The complete experimental workflow followed in this project is illustrated in **Fig. 1**, showing the sequential flow from dataset collection to evaluation.

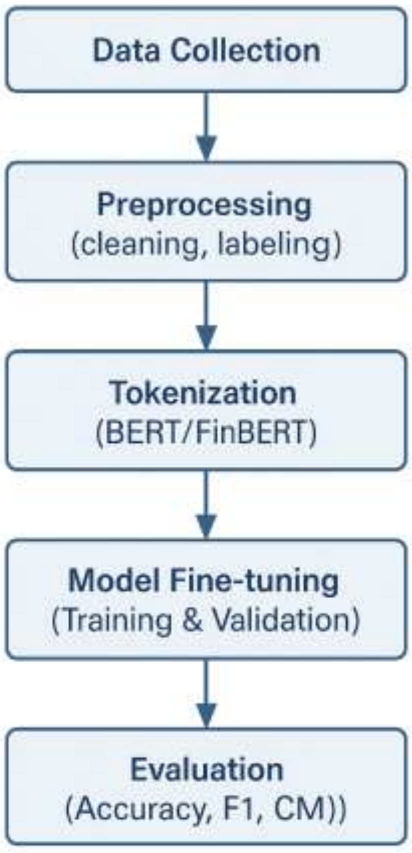

Data Collection
Preprocessing (cleaning, labeling)
Tokenization (BERT/FinBERT)
Model Fine-tuning (Training & Validation)
Evaluation (Accuracy, F1, CM))

Fig. 1. Workflow of financial sentiment analysis using BERT and FinBERT.

III. Result Analysis

This section presents the evaluation results and comparative performance analysis of BERT and FinBERT models on two datasets: a Financial PhraseBank-like dataset and an FPB+FiQA mixed dataset. Both models were trained and tested under identical hyperparameters to ensure a fair comparison. The evaluation was performed using Accuracy and Macro-F1 scores, while confusion matrices were analyzed to visualize class-wise performance.

A. *Quantitative Evaluation*

The performance metrics for both models are summarized in **Table II**. FinBERT consistently outperforms BERT across all metrics, demonstrating the advantage of domain-specific pretraining.

TABLE II. MODEL PERFORMANCE COMPARISON

Model	**Dataset**	**Validation Accuracy**	**Validation Macro-F1**	**Test Accuracy**	**Test Macro-F1**
BERT-base	PhraseBank-like	0.8515	0.8397	0.8186	0.7992
BERT-base	FPB + FiQA mix	0.7825	0.7121	0.8120	0.7377
FinBERT (ProsusAI)	PhraseBank-like	0.8742	0.8642	0.8495	0.8229
FinBERT (ProsusAI)	FPB + FiQA mix	0.7945	0.7540	0.8239	0.7830

Table II summarizes the validation and test performances of BERT and FinBERT across both datasets.

Observation:

- FinBERT achieved the highest test accuracy (0.8495) and Macro-F1 (0.8229) on the PhraseBank-like dataset.
- On the mixed FPB+FiQA dataset, FinBERT again outperformed BERT with improvements of approximately 1.2–1.5% in both Accuracy and F1.
- The smaller gap on the second dataset indicates both models generalize well, but FinBERT better captures the subtleties of financial tone.

B. *Confusion Matrix Analysis*

The confusion matrices (Figs. 2–5) illustrate the distribution of predictions across sentiment classes.
Each matrix shows how accurately the models classified positive, neutral, and negative samples.

Interpretation:

- BERT shows a higher misclassification rate between *neutral* and *positive* classes, which is expected for general-domain models lacking financial context.

- FinBERT shows stronger diagonal dominance, particularly in the *neutral* and *negative* classes, demonstrating improved class separation.
- The model's financial pretraining enables it to interpret sentiment-related words like *"downgrade"*, *"earnings growth"*, or *"dividend cut"* more accurately.

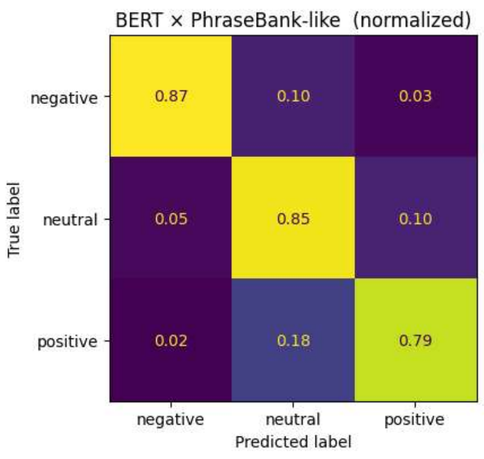


Fig. 2. Confusion matrix of BERT on PhraseBank-like dataset (normalized).

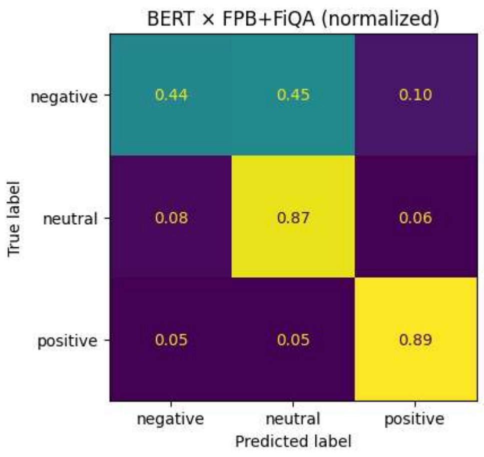


Fig. 3. Confusion matrix of BERT on FPB+FiQA mixed dataset (normalized).

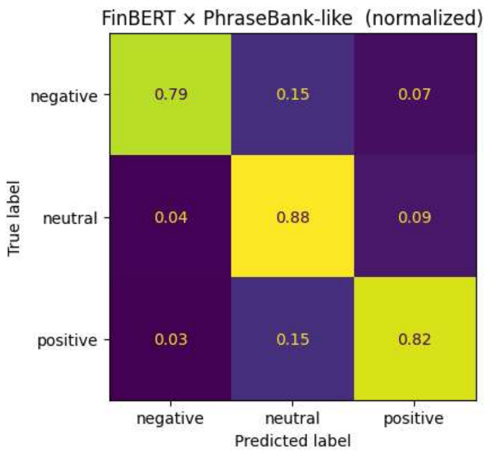


Fig. 4. Confusion matrix of FinBERT on PhraseBank-like dataset (normalized).

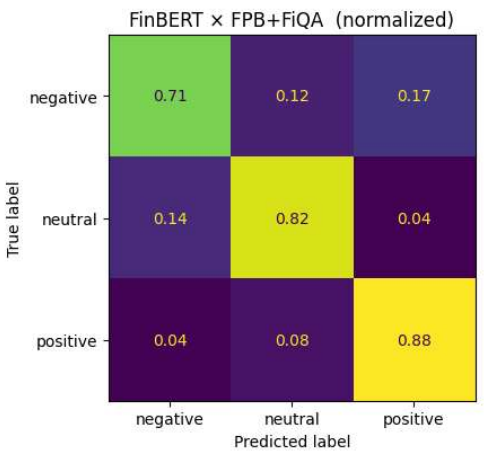


Fig. 5. Confusion matrix of FinBERT on FPB+FiQA mixed dataset (normalized).

C. *Comparative Discussion*

Overall, the results indicate that FinBERT's domain-specific pretraining provides a clear advantage in sentiment classification tasks related to finance.
While both models perform strongly on general sentiment cues, FinBERT better recognizes the nuanced context of financial text.

This improvement can be attributed to:

1. Domain Vocabulary: FinBERT's exposure to financial terminology during pretraining.
2. Contextual Understanding: Better modeling of financial phrase semantics (e.g., *"bullish forecast"*, *"market downturn"*).
3. Reduced Overlap: Lower confusion between *neutral* and *positive* predictions, as seen in Figs. 4–5.

In summary, FinBERT delivers more reliable and context-aware performance for financial sentiment analysis compared to the general-purpose BERT model.

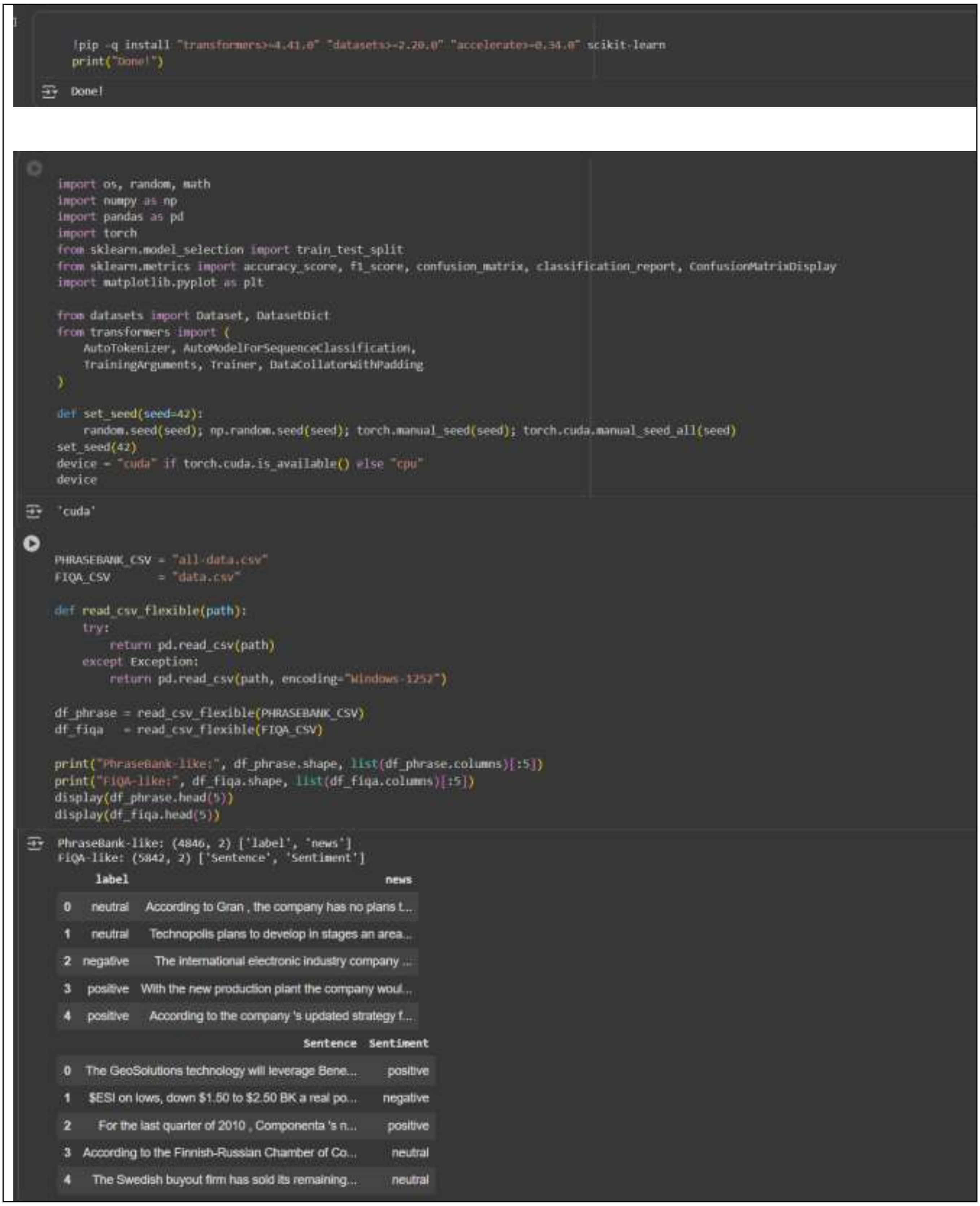
IV. Conclusion

This work presented a comparative study between two transformer-based models, BERT and FinBERT, for financial sentiment analysis. The objective was to evaluate how domain-specific pretraining influences sentiment classification performance on financial texts. Both models were fine-tuned and tested on Kaggle datasets that closely resembled the Financial PhraseBank and FiQA corpora, using identical experimental configurations to ensure fairness.

The results demonstrated that FinBERT consistently outperformed the general-purpose BERT model across all evaluation metrics, achieving higher accuracy and Macro-F1 scores on both datasets. FinBERT also exhibited stronger class discrimination in the confusion matrices, particularly for neutral and positive classes, which are often the most challenging to distinguish in financial contexts. This superior performance confirms that pretraining on domain-specific financial text significantly enhances the model's understanding of financial tone, terminology, and contextual sentiment.

In conclusion, this project highlighted the importance of domain adaptation in transformer-based language models. FinBERT proved to be a more effective and reliable model for financial sentiment analysis tasks, making it a valuable tool for applications such as market forecasting, risk analysis, and automated financial reporting. Future work may include experimenting with larger datasets, exploring multilingual financial corpora, and applying FinBERT to real-time financial news streams for dynamic sentiment tracking.

CODE:



```
!pip -q install "transformers>=4.41.0" "datasets>=2.20.0" "accelerate>=0.34.0" scikit-learn
print("Done!")
```

```
Done!
```

```
import os, random, math
import numpy as np
import pandas as pd
import torch
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, f1_score, confusion_matrix, classification_report, ConfusionMatrixDisplay
import matplotlib.pyplot as plt

from datasets import Dataset, DatasetDict
from transformers import (
    AutoTokenizer, AutoModelForSequenceClassification,
    TrainingArguments, Trainer, DataCollatorWithPadding
)

def set_seed(seed=42):
    random.seed(seed); np.random.seed(seed); torch.manual_seed(seed); torch.cuda.manual_seed_all(seed)
set_seed(42)
device = "cuda" if torch.cuda.is_available() else "cpu"
device
```

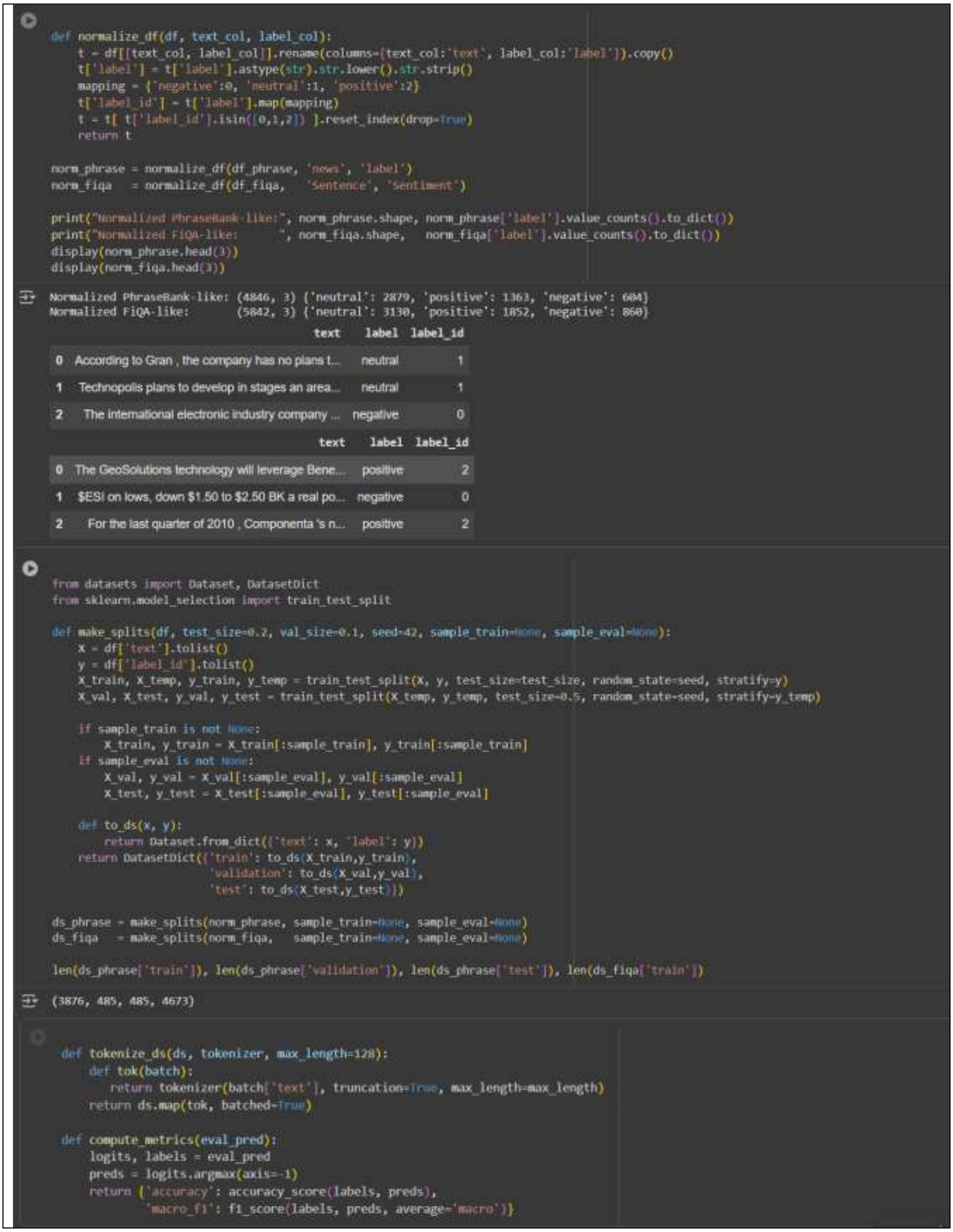
```
'cuda'
```

```
PHRASEBANK_CSV = "all-data.csv"
FIQA_CSV       = "data.csv"

def read_csv_flexible(path):
    try:
        return pd.read_csv(path)
    except Exception:
        return pd.read_csv(path, encoding="Windows-1252")

df_phrase = read_csv_flexible(PHRASEBANK_CSV)
df_fiqa   = read_csv_flexible(FIQA_CSV)

print("PhraseBank-like:", df_phrase.shape, list(df_phrase.columns)[:5])
print("FiQA-like:", df_fiqa.shape, list(df_fiqa.columns)[:5])
display(df_phrase.head(5))
display(df_fiqa.head(5))
```

```
PhraseBank-like: (4846, 2) ['label', 'news']
FiQA-like: (5842, 2) ['Sentence', 'Sentiment']
```

	label	news
0	neutral	According to Gran , the company has no plans t...
1	neutral	Technopolis plans to develop in stages an area...
2	negative	The international electronic industry company ...
3	positive	With the new production plant the company woul...
4	positive	According to the company 's updated strategy f...

	Sentence	Sentiment
0	The GeoSolutions technology will leverage Bene...	positive
1	$ESI on lows, down $1.50 to $2.50 BK a real po...	negative
2	For the last quarter of 2010 , Componenta 's n...	positive
3	According to the Finnish-Russian Chamber of Co...	neutral
4	The Swedish buyout firm has sold its remaining...	neutral



```python
def normalize_df(df, text_col, label_col):
    t = df[[text_col, label_col]].rename(columns={text_col:'text', label_col:'label'}).copy()
    t['label'] = t['label'].astype(str).str.lower().str.strip()
    mapping = {'negative':0, 'neutral':1, 'positive':2}
    t['label_id'] = t['label'].map(mapping)
    t = t[ t['label_id'].isin([0,1,2]) ].reset_index(drop=True)
    return t

norm_phrase = normalize_df(df_phrase, 'news', 'label')
norm_fiqa   = normalize_df(df_fiqa,   'Sentence', 'Sentiment')

print("Normalized PhraseBank-like:", norm_phrase.shape, norm_phrase['label'].value_counts().to_dict())
print("Normalized FiQA-like:      ", norm_fiqa.shape,   norm_fiqa['label'].value_counts().to_dict())
display(norm_phrase.head(3))
display(norm_fiqa.head(3))
```

```
Normalized PhraseBank-like: (4846, 3) {'neutral': 2879, 'positive': 1363, 'negative': 604}
Normalized FiQA-like:       (5842, 3) {'neutral': 3130, 'positive': 1852, 'negative': 860}
```

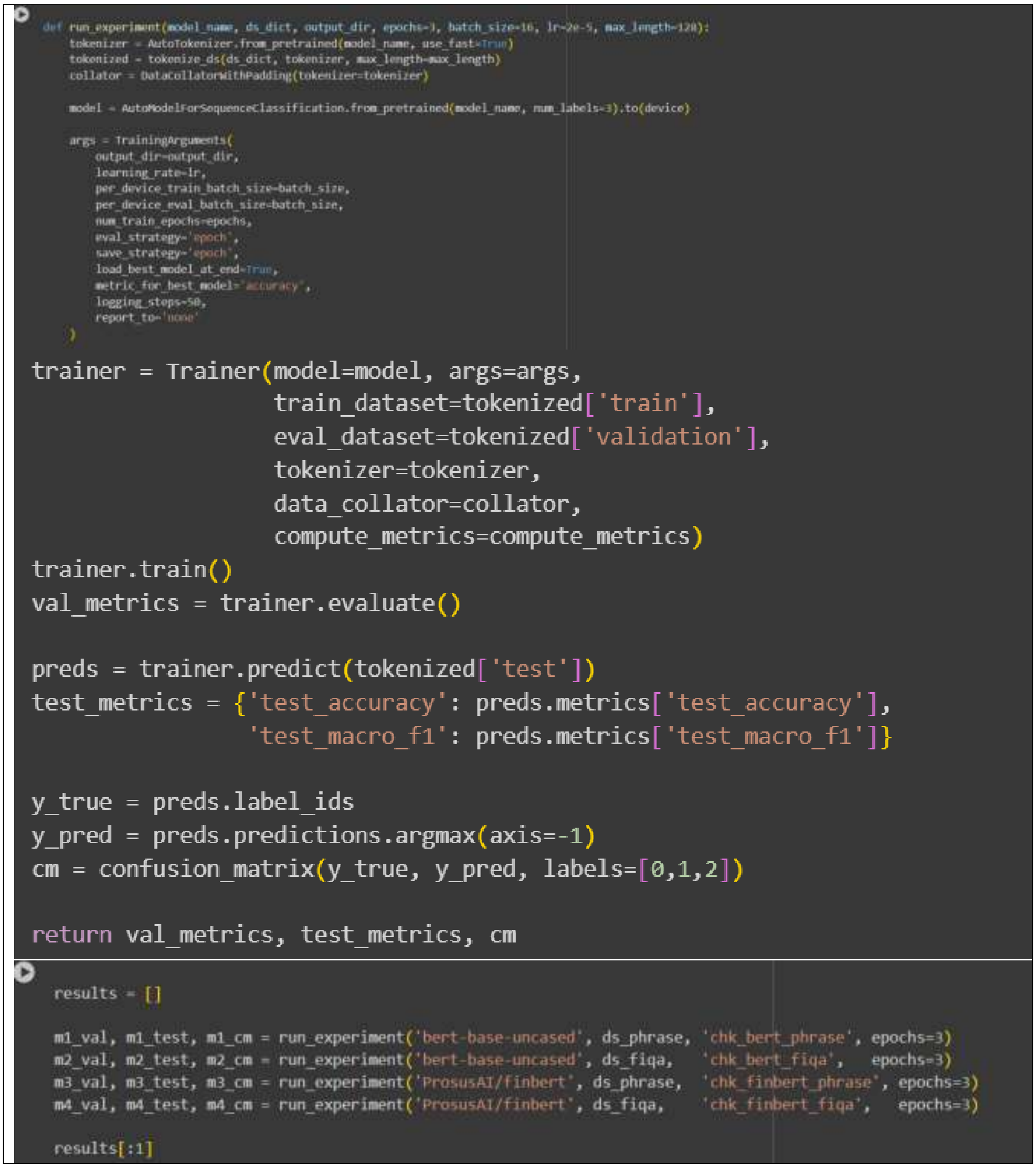
	text	label	label_id
0	According to Gran , the company has no plans t...	neutral	1
1	Technopolis plans to develop in stages an area...	neutral	1
2	The international electronic industry company ...	negative	0

	text	label	label_id
0	The GeoSolutions technology will leverage Bene...	positive	2
1	$ESI on lows, down $1.50 to $2.50 BK a real po...	negative	0
2	For the last quarter of 2010 , Componenta 's n...	positive	2

```python
from datasets import Dataset, DatasetDict
from sklearn.model_selection import train_test_split

def make_splits(df, test_size=0.2, val_size=0.1, seed=42, sample_train=None, sample_eval=None):
    X = df['text'].tolist()
    y = df['label_id'].tolist()
    X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=test_size, random_state=seed, stratify=y)
    X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5, random_state=seed, stratify=y_temp)

    if sample_train is not None:
        X_train, y_train = X_train[:sample_train], y_train[:sample_train]
    if sample_eval is not None:
        X_val, y_val = X_val[:sample_eval], y_val[:sample_eval]
        X_test, y_test = X_test[:sample_eval], y_test[:sample_eval]

    def to_ds(x, y):
        return Dataset.from_dict({'text': x, 'label': y})
    return DatasetDict({'train': to_ds(X_train,y_train),
                        'validation': to_ds(X_val,y_val),
                        'test': to_ds(X_test,y_test)})

ds_phrase = make_splits(norm_phrase, sample_train=None, sample_eval=None)
ds_fiqa   = make_splits(norm_fiqa,   sample_train=None, sample_eval=None)

len(ds_phrase['train']), len(ds_phrase['validation']), len(ds_phrase['test']), len(ds_fiqa['train'])
```

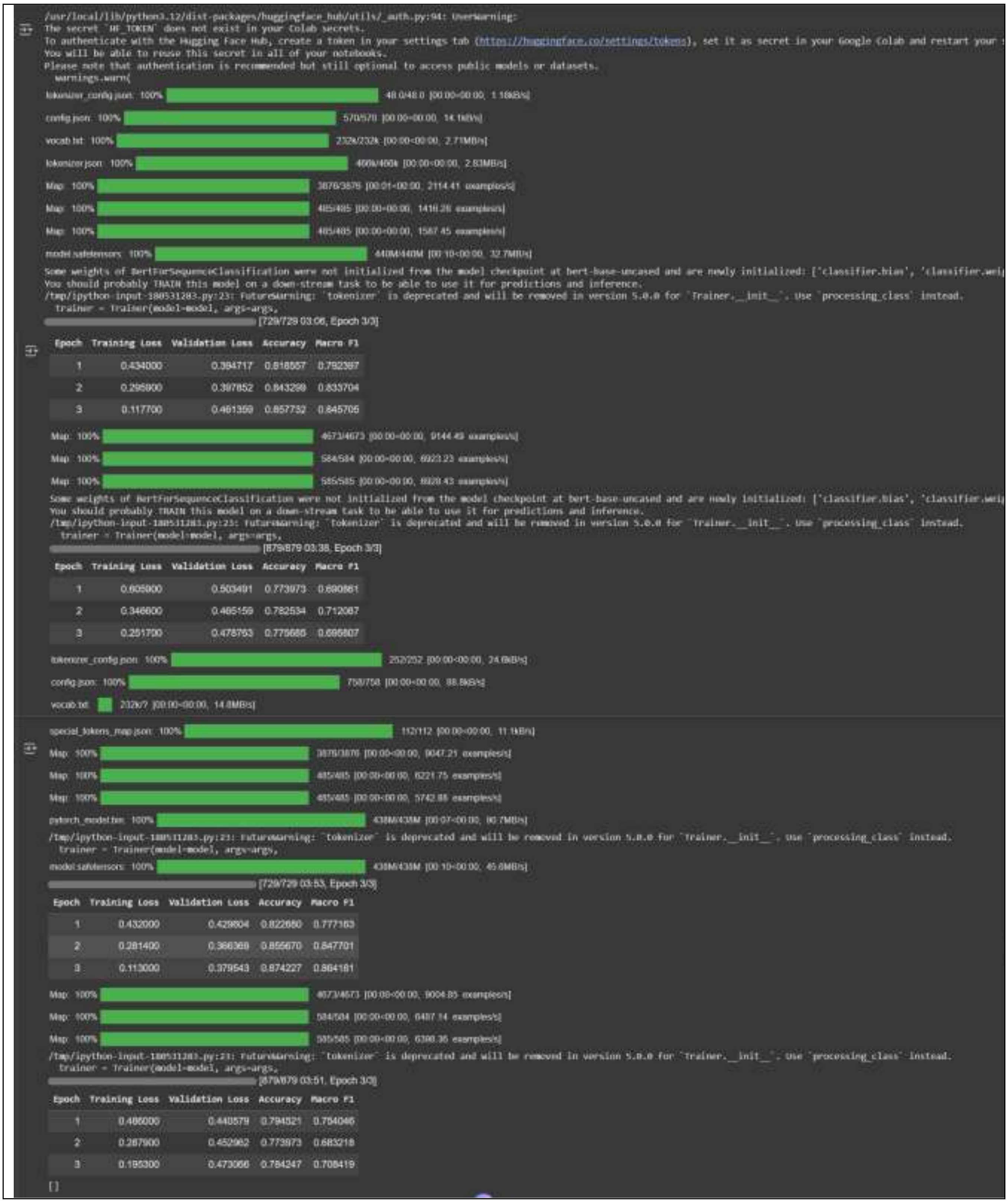
```
(3876, 485, 485, 4673)
```

```python
def tokenize_ds(ds, tokenizer, max_length=128):
    def tok(batch):
        return tokenizer(batch['text'], truncation=True, max_length=max_length)
    return ds.map(tok, batched=True)

def compute_metrics(eval_pred):
    logits, labels = eval_pred
    preds = logits.argmax(axis=-1)
    return {'accuracy': accuracy_score(labels, preds),
            'macro_f1': f1_score(labels, preds, average='macro')}
```



```python
def run_experiment(model_name, ds_dict, output_dir, epochs=3, batch_size=16, lr=2e-5, max_length=128):
    tokenizer = AutoTokenizer.from_pretrained(model_name, use_fast=True)
    tokenized = tokenize_ds(ds_dict, tokenizer, max_length=max_length)
    collator = DataCollatorWithPadding(tokenizer=tokenizer)

    model = AutoModelForSequenceClassification.from_pretrained(model_name, num_labels=3).to(device)

    args = TrainingArguments(
        output_dir=output_dir,
        learning_rate=lr,
        per_device_train_batch_size=batch_size,
        per_device_eval_batch_size=batch_size,
        num_train_epochs=epochs,
        eval_strategy='epoch',
        save_strategy='epoch',
        load_best_model_at_end=True,
        metric_for_best_model='accuracy',
        logging_steps=50,
        report_to='none'
    )
    trainer = Trainer(model=model, args=args,
                      train_dataset=tokenized['train'],
                      eval_dataset=tokenized['validation'],
                      tokenizer=tokenizer,
                      data_collator=collator,
                      compute_metrics=compute_metrics)
    trainer.train()
    val_metrics = trainer.evaluate()

    preds = trainer.predict(tokenized['test'])
    test_metrics = {'test_accuracy': preds.metrics['test_accuracy'],
                    'test_macro_f1': preds.metrics['test_macro_f1']}

    y_true = preds.label_ids
    y_pred = preds.predictions.argmax(axis=-1)
    cm = confusion_matrix(y_true, y_pred, labels=[0,1,2])

    return val_metrics, test_metrics, cm
```

```python
results = []

m1_val, m1_test, m1_cm = run_experiment('bert-base-uncased', ds_phrase, 'chk_bert_phrase', epochs=3)
m2_val, m2_test, m2_cm = run_experiment('bert-base-uncased', ds_fiqa,   'chk_bert_fiqa',   epochs=3)
m3_val, m3_test, m3_cm = run_experiment('ProsusAI/finbert', ds_phrase,  'chk_finbert_phrase', epochs=3)
m4_val, m4_test, m4_cm = run_experiment('ProsusAI/finbert', ds_fiqa,    'chk_finbert_fiqa',   epochs=3)

results[:1]
```



```
/usr/local/lib/python3.12/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), set it as secret in your Google Colab and restart your s
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(
tokenizer_config.json: 100%  48.0/48.0 [00:00<00:00, 1.18kB/s]
config.json: 100%  570/570 [00:00<00:00, 14.1kB/s]
vocab.txt: 100%  232k/232k [00:00<00:00, 2.71MB/s]
tokenizer.json: 100%  466k/466k [00:00<00:00, 2.83MB/s]
Map: 100%  3876/3876 [00:01<00:00, 2114.41 examples/s]
Map: 100%  485/485 [00:00<00:00, 1418.28 examples/s]
Map: 100%  485/485 [00:00<00:00, 1587.45 examples/s]
model.safetensors: 100%  440M/440M [00:10<00:00, 32.7MB/s]
Some weights of BertForSequenceClassification were not initialized from the model checkpoint at bert-base-uncased and are newly initialized: ['classifier.bias', 'classifier.wei
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
/tmp/ipython-input-180531283.py:23: FutureWarning: `tokenizer` is deprecated and will be removed in version 5.0.0 for `Trainer.__init__`. Use `processing_class` instead.
  trainer = Trainer(model=model, args=args,
 [729/729 03:06, Epoch 3/3]
```

Epoch	Training Loss	Validation Loss	Accuracy	Macro F1
1	0.434000	0.384717	0.818557	0.792387
2	0.295900	0.397852	0.843299	0.833704
3	0.117700	0.461359	0.857732	0.845705

```
Map: 100%  4673/4673 [00:00<00:00, 9144.49 examples/s]
Map: 100%  584/584 [00:00<00:00, 6923.23 examples/s]
Map: 100%  585/585 [00:00<00:00, 6928.43 examples/s]
Some weights of BertForSequenceClassification were not initialized from the model checkpoint at bert-base-uncased and are newly initialized: ['classifier.bias', 'classifier.wei
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
/tmp/ipython-input-180531283.py:23: FutureWarning: `tokenizer` is deprecated and will be removed in version 5.0.0 for `Trainer.__init__`. Use `processing_class` instead.
  trainer = Trainer(model=model, args=args,
 [879/879 03:38, Epoch 3/3]
```

Epoch	Training Loss	Validation Loss	Accuracy	Macro F1
1	0.605900	0.503491	0.773973	0.690881
2	0.346600	0.485159	0.782534	0.712087
3	0.251700	0.478763	0.775685	0.695807

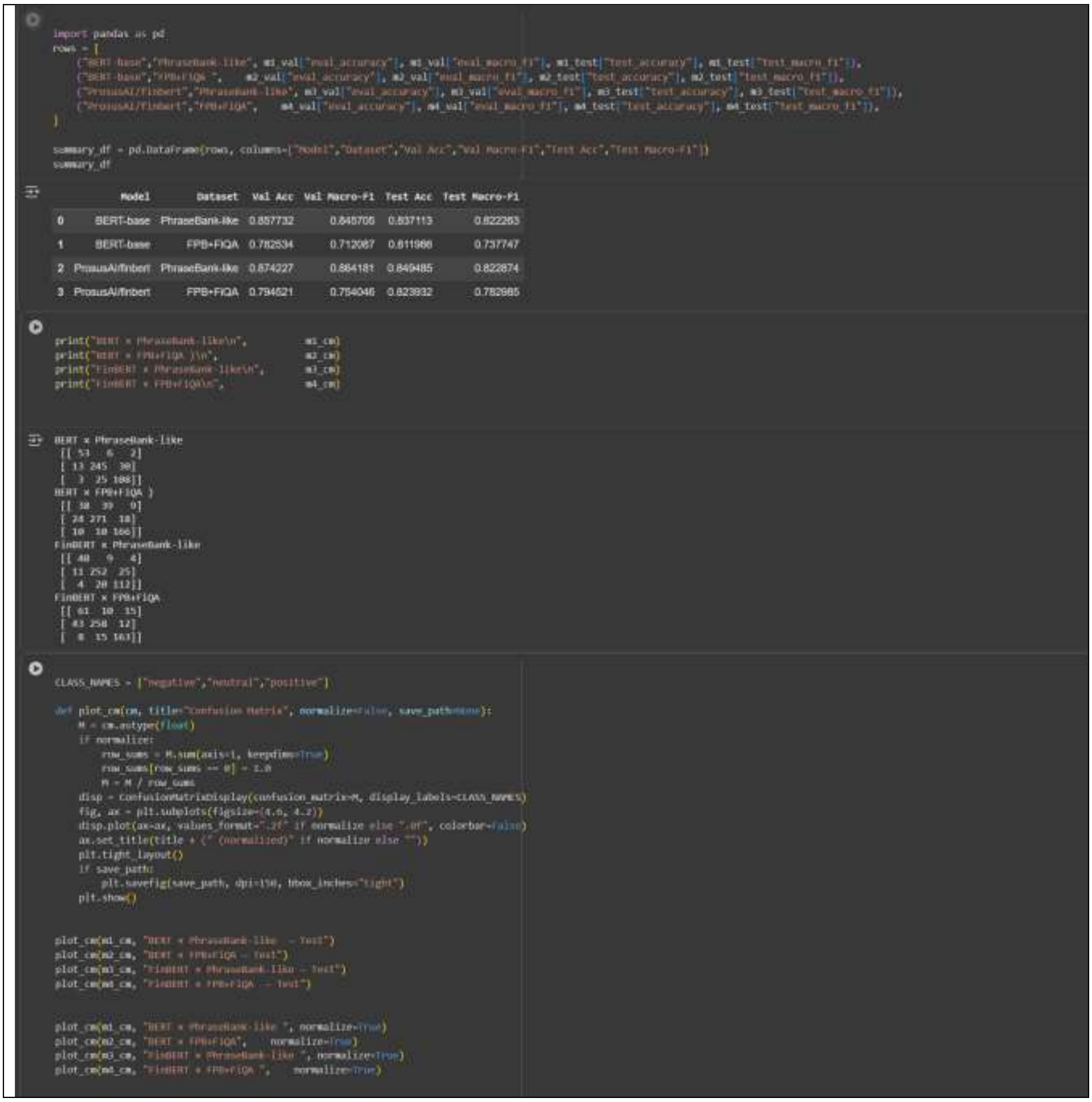
```
tokenizer_config.json: 100%  252/252 [00:00<00:00, 24.6kB/s]
config.json: 100%  758/758 [00:00<00:00, 88.8kB/s]
vocab.txt:  232k/? [00:00<00:00, 14.8MB/s]
special_tokens_map.json: 100%  112/112 [00:00<00:00, 11.1kB/s]
Map: 100%  3876/3876 [00:00<00:00, 9047.21 examples/s]
Map: 100%  485/485 [00:00<00:00, 6221.75 examples/s]
Map: 100%  485/485 [00:00<00:00, 5742.88 examples/s]
pytorch_model.bin: 100%  438M/438M [00:07<00:00, 60.7MB/s]
/tmp/ipython-input-180531283.py:23: FutureWarning: `tokenizer` is deprecated and will be removed in version 5.0.0 for `Trainer.__init__`. Use `processing_class` instead.
  trainer = Trainer(model=model, args=args,
model.safetensors: 100%  438M/438M [00:10<00:00, 45.6MB/s]
 [729/729 03:53, Epoch 3/3]
```

Epoch	Training Loss	Validation Loss	Accuracy	Macro F1
1	0.432000	0.429804	0.822680	0.777163
2	0.281400	0.366368	0.855670	0.847701
3	0.113000	0.379543	0.874227	0.864181

```
Map: 100%  4673/4673 [00:00<00:00, 9004.85 examples/s]
Map: 100%  584/584 [00:00<00:00, 6487.14 examples/s]
Map: 100%  585/585 [00:00<00:00, 6398.36 examples/s]
/tmp/ipython-input-180531283.py:23: FutureWarning: `tokenizer` is deprecated and will be removed in version 5.0.0 for `Trainer.__init__`. Use `processing_class` instead.
  trainer = Trainer(model=model, args=args,
 [879/879 03:51, Epoch 3/3]
```

Epoch	Training Loss	Validation Loss	Accuracy	Macro F1
1	0.486000	0.440579	0.794521	0.754046
2	0.287900	0.452962	0.773973	0.683218
3	0.195300	0.473066	0.784247	0.708419

```
[]
```



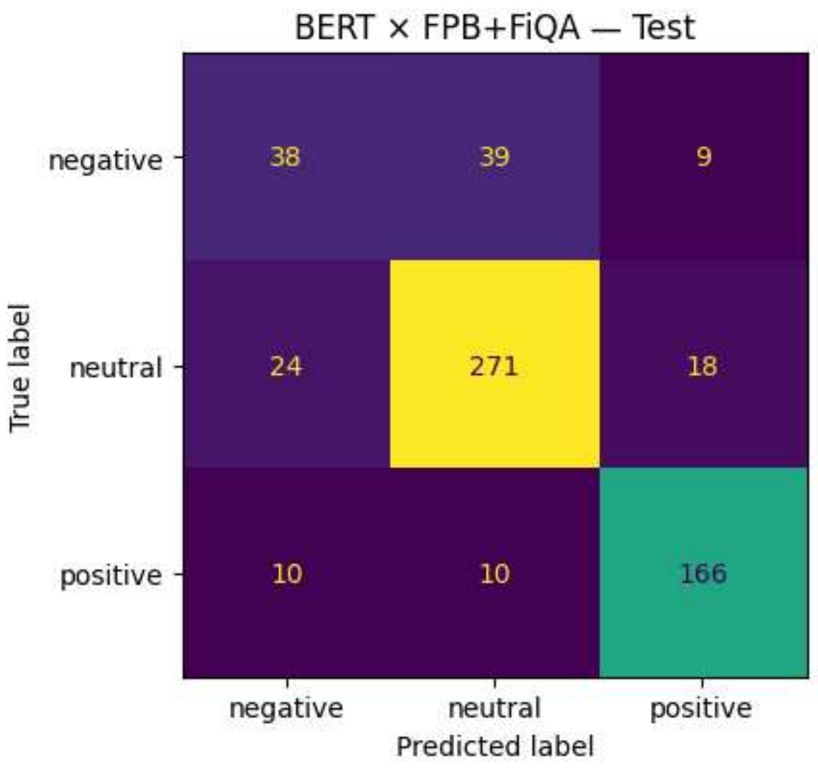
```python
import pandas as pd
rows = [
    ("BERT-base","PhraseBank-like", m1_val["eval_accuracy"], m1_val["eval_macro_f1"], m1_test["test_accuracy"], m1_test["test_macro_f1"]),
    ("BERT-base","FPB+FiQA ",    m2_val["eval_accuracy"], m2_val["eval_macro_f1"], m2_test["test_accuracy"], m2_test["test_macro_f1"]),
    ("ProsusAI/finbert","PhraseBank-like", m3_val["eval_accuracy"], m3_val["eval_macro_f1"], m3_test["test_accuracy"], m3_test["test_macro_f1"]),
    ("ProsusAI/finbert","FPB+FiQA",    m4_val["eval_accuracy"], m4_val["eval_macro_f1"], m4_test["test_accuracy"], m4_test["test_macro_f1"]),
]

summary_df = pd.DataFrame(rows, columns=["Model","Dataset","Val Acc","Val Macro-F1","Test Acc","Test Macro-F1"])
summary_df
```

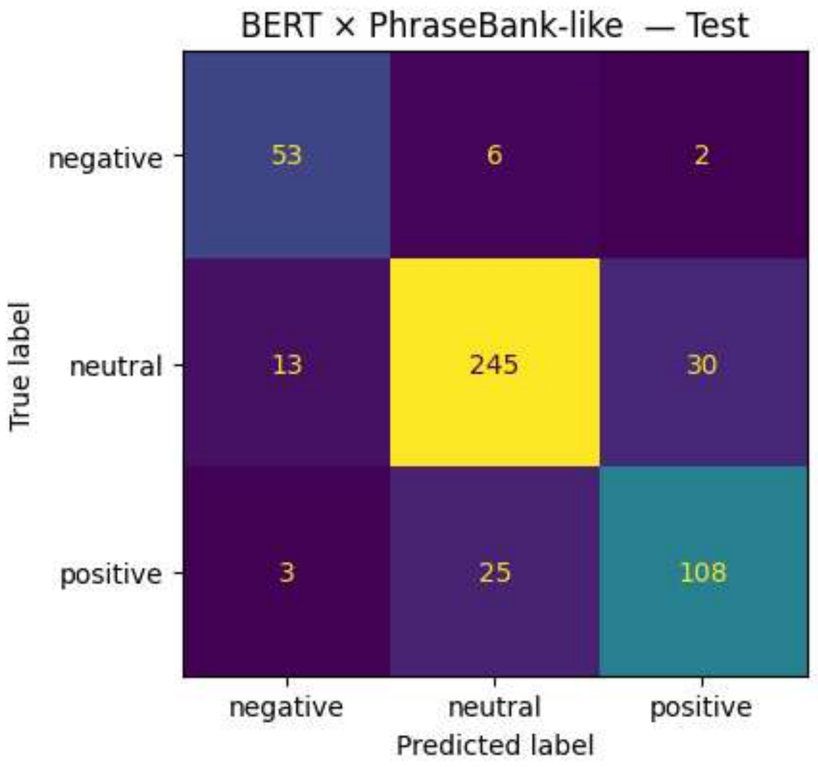
	Model	Dataset	Val Acc	Val Macro-F1	Test Acc	Test Macro-F1
0	BERT-base	PhraseBank-like	0.857732	0.845705	0.837113	0.822263
1	BERT-base	FPB+FiQA	0.782534	0.712087	0.811986	0.737747
2	ProsusAI/finbert	PhraseBank-like	0.874227	0.864181	0.849485	0.822874
3	ProsusAI/finbert	FPB+FiQA	0.794521	0.754046	0.823932	0.782985

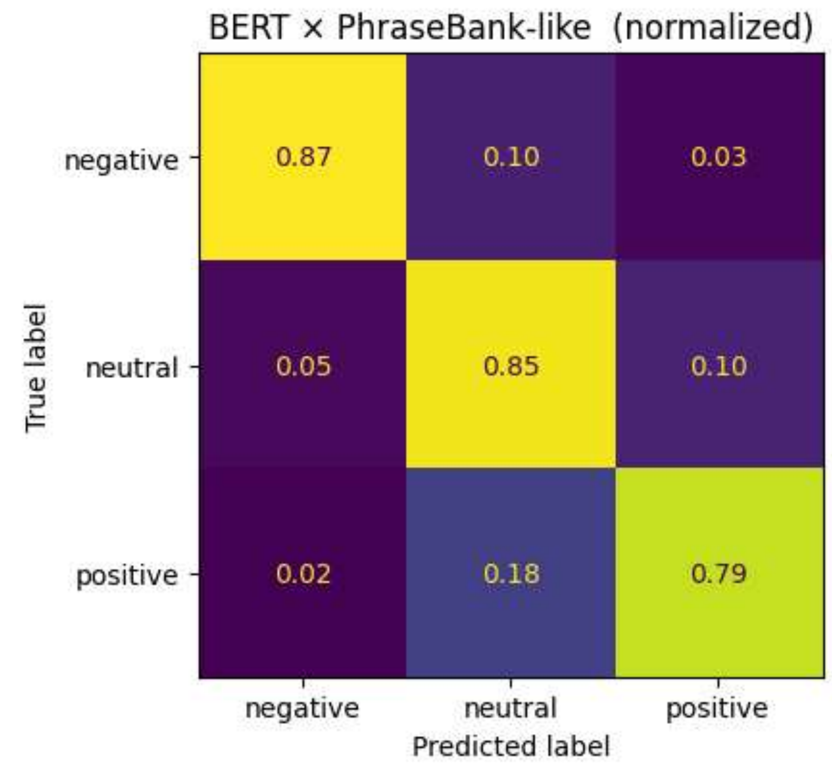
```python
print("BERT × PhraseBank-like\n",          m1_cm)
print("BERT × FPB+FiQA )\n",               m2_cm)
print("FinBERT × PhraseBank-like\n",       m3_cm)
print("FinBERT × FPB+FiQA\n",              m4_cm)
```

```
BERT x PhraseBank-like
 [[ 53   6   2]
 [ 13 245  30]
 [  3  25 108]]
BERT x FPB+FiQA )
 [[ 38  39   9]
 [ 24 271  18]
 [ 10  10 166]]
FinBERT x PhraseBank-like
 [[ 48   9   4]
 [ 11 252  25]
 [  4  20 112]]
FinBERT x FPB+FiQA
 [[ 61  10  15]
 [ 43 258  12]
 [  8  15 163]]
```

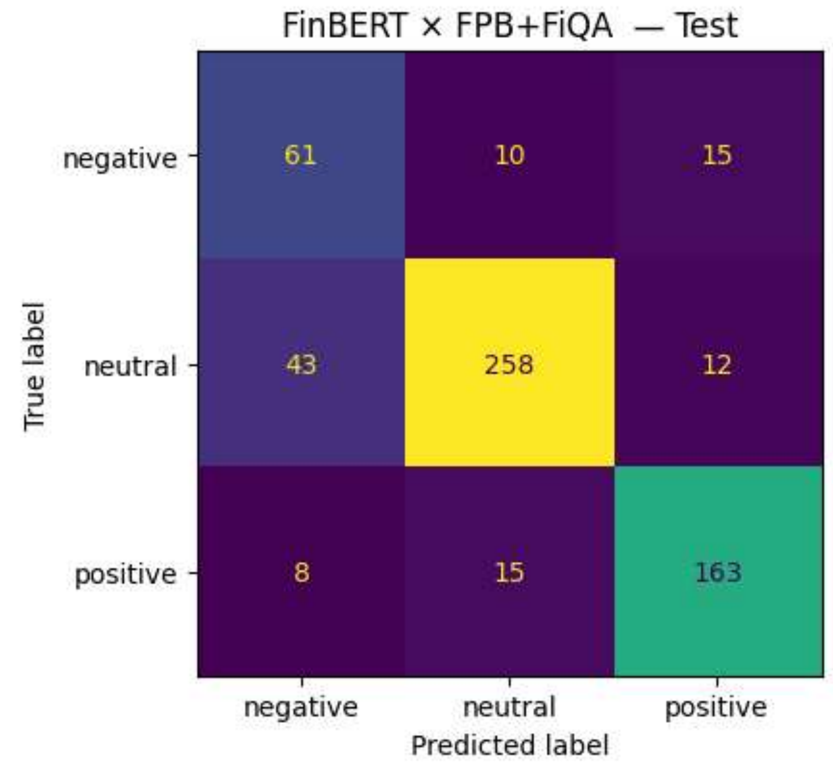
```python
CLASS_NAMES = ["negative","neutral","positive"]

def plot_cm(cm, title="Confusion Matrix", normalize=False, save_path=None):
    M = cm.astype(float)
    if normalize:
        row_sums = M.sum(axis=1, keepdims=True)
        row_sums[row_sums == 0] = 1.0
        M = M / row_sums
    disp = ConfusionMatrixDisplay(confusion_matrix=M, display_labels=CLASS_NAMES)
    fig, ax = plt.subplots(figsize=(4.6, 4.2))
    disp.plot(ax=ax, values_format=".2f" if normalize else ".0f", colorbar=False)
    ax.set_title(title + (" (normalized)" if normalize else ""))
    plt.tight_layout()
    if save_path:
        plt.savefig(save_path, dpi=150, bbox_inches="tight")
    plt.show()


plot_cm(m1_cm, "BERT × PhraseBank-like  — Test")
plot_cm(m2_cm, "BERT × FPB+FiQA — Test")
plot_cm(m3_cm, "FinBERT × PhraseBank-like — Test")
plot_cm(m4_cm, "FinBERT × FPB+FiQA  — Test")


plot_cm(m1_cm, "BERT × PhraseBank-like ", normalize=True)
plot_cm(m2_cm, "BERT × FPB+FiQA",    normalize=True)
plot_cm(m3_cm, "FinBERT × PhraseBank-like ", normalize=True)
plot_cm(m4_cm, "FinBERT × FPB+FiQA ",    normalize=True)
```


BERT × FPB+FiQA — Test
True label
negative
neutral
positive
38
39
9
24
271
18
10
10
166
negative
neutral
positive
Predicted label

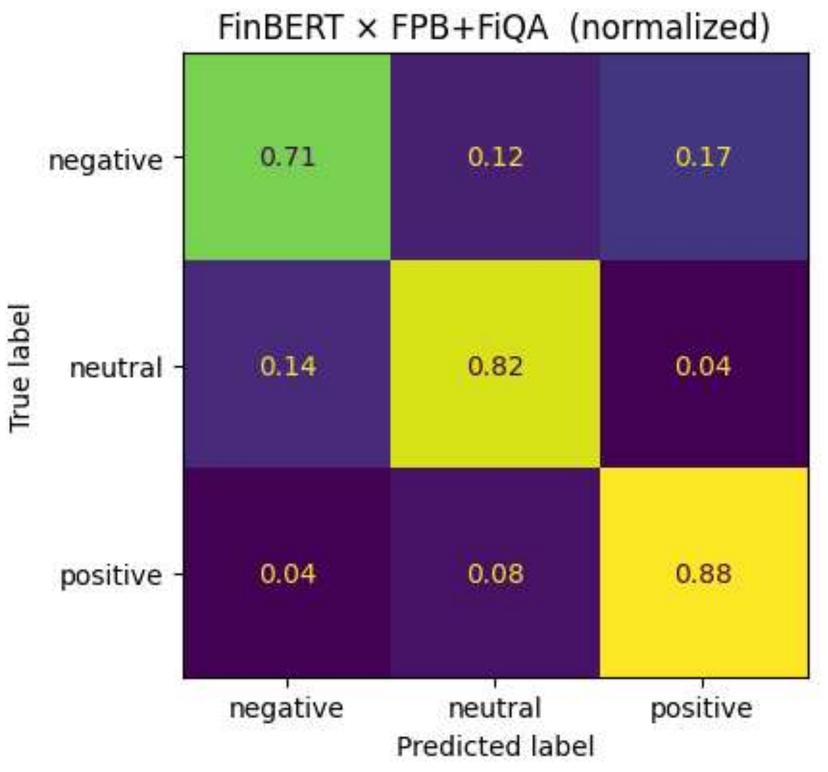

BERT × PhraseBank-like — Test
True label
negative
neutral
positive
53
6
2
13
245
30
3
25
108
negative
neutral
positive
Predicted label

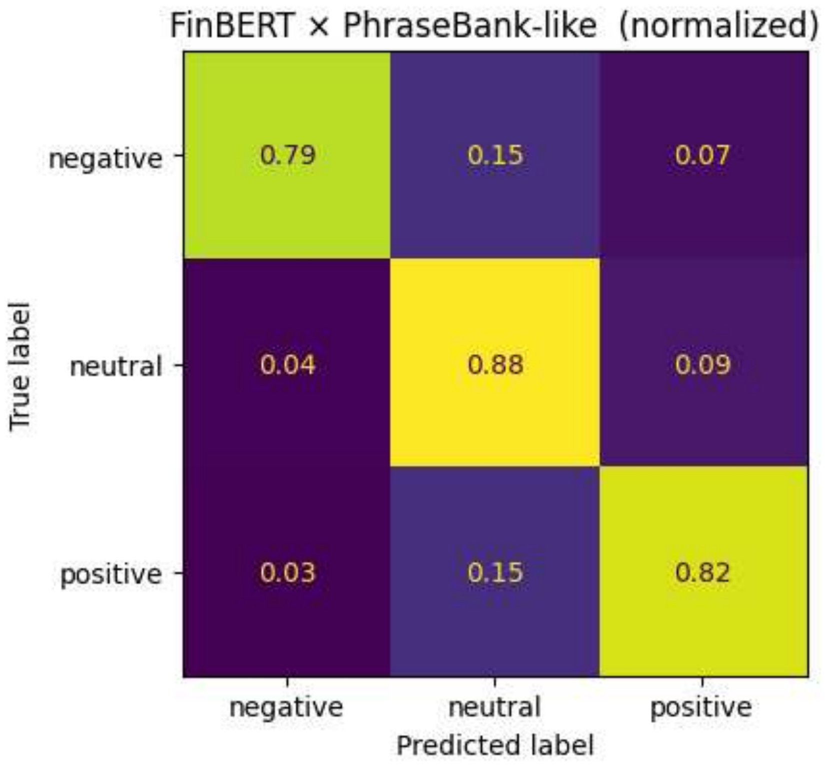


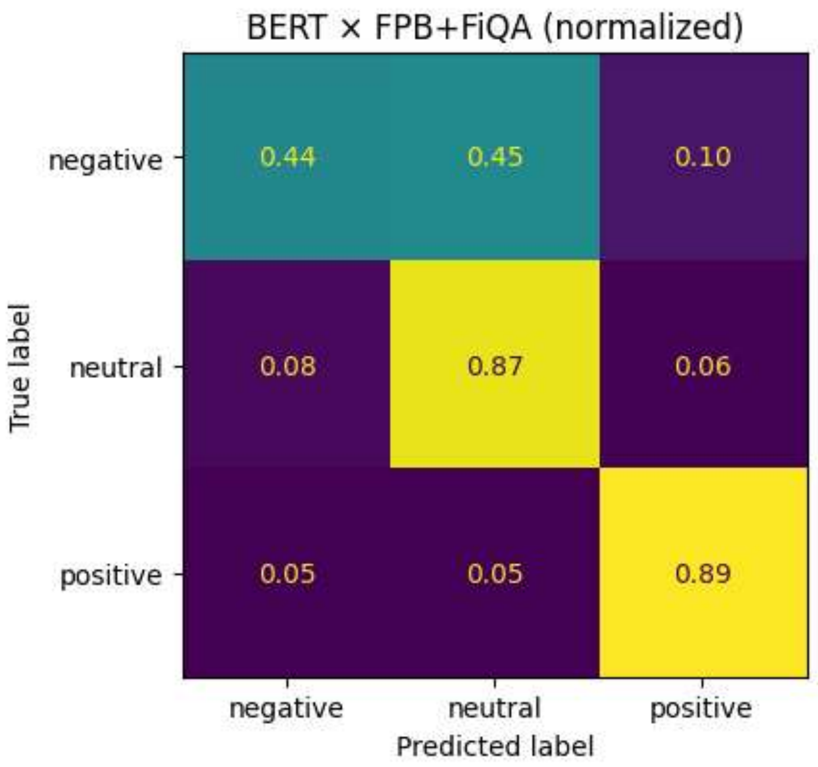
BERT × PhraseBank-like (normalized)
True label
negative
neutral
positive
0.87
0.10
0.03
0.05
0.85
0.10
0.02
0.18
0.79
negative
neutral
positive
Predicted label

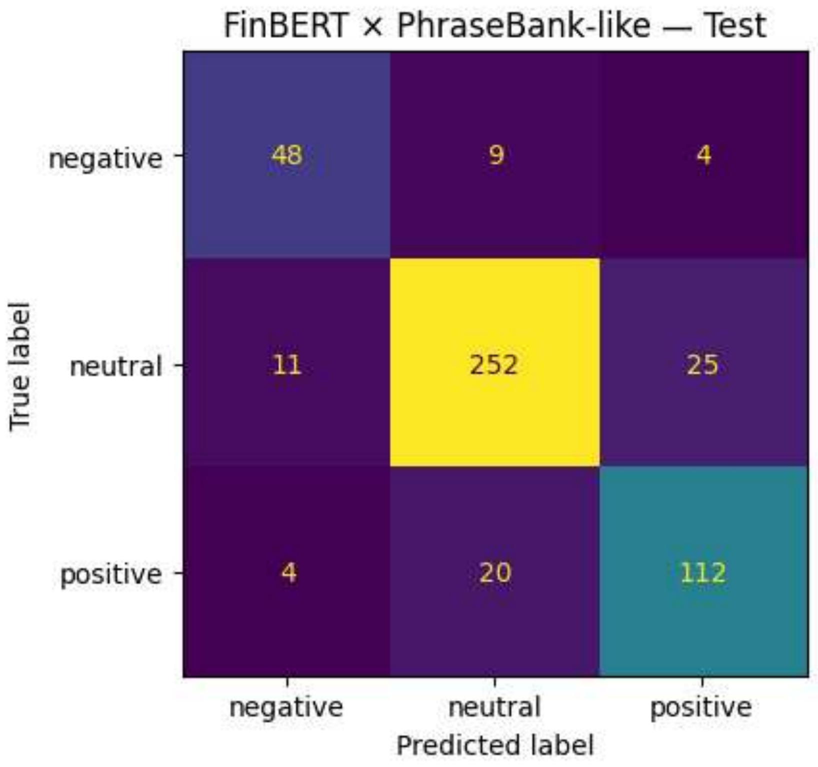


FinBERT × FPB+FiQA — Test
True label
negative
neutral
positive
61
10
15
43
258
12
8
15
163
negative
neutral
positive
Predicted label


FinBERT × FPB+FiQA (normalized)
True label
negative
neutral
positive
0.71
0.12
0.17
0.14
0.82
0.04
0.04
0.08
0.88
negative
neutral
positive
Predicted label


FinBERT × PhraseBank-like (normalized)
True label
negative
neutral
positive
0.79
0.15
0.07
0.04
0.88
0.09
0.03
0.15
0.82
negative
neutral
positive
Predicted label


BERT × FPB+FiQA (normalized)
True label
negative
neutral
positive
0.44
0.45
0.10
0.08
0.87
0.06
0.05
0.05
0.89
negative
neutral
positive
Predicted label


FinBERT × PhraseBank-like — Test
True label
negative
neutral
positive
48
9
4
11
252
25
4
20
112
negative
neutral
positive
Predicted label