```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
import pandas as pd
```

```
train_df = pd.read_excel("/content/drive/MyDrive/Books/train.xlsx")
eval_df = pd.read_excel("/content/drive/MyDrive/Books/evaluation.xlsx")
```

```
!pip install transformers
!pip install datasets
```

```
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.9/dist-packages (from requests->transformers) (3.4)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.9/dist-packages (from requests->transformers) (2022.12.7
Requirement already satisfied: charset-normalizer~=2.0.0 in /usr/local/lib/python3.9/dist-packages (from requests->transformers) (2.0
Installing collected packages: tokenizers, huggingface-hub, transformers
Successfully installed huggingface-hub-0.13.3 tokenizers-0.13.2 transformers-4.27.2
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting datasets
  Downloading datasets-2.10.1-py3-none-any.whl (469 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 469.0/469.0 KB 1.8 MB/s eta 0:00:00
Requirement already satisfied: fsspec[http]>=2021.11.1 in /usr/local/lib/python3.9/dist-packages (from datasets) (2023.3.0)
Requirement already satisfied: requests>=2.19.0 in /usr/local/lib/python3.9/dist-packages (from datasets) (2.27.1)
Collecting multiprocess
  Downloading multiprocess-0.70.14-py39-none-any.whl (132 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 132.9/132.9 KB 16.1 MB/s eta 0:00:00
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.9/dist-packages (from datasets) (1.22.4)
Requirement already satisfied: huggingface-hub<1.0.0,>=0.2.0 in /usr/local/lib/python3.9/dist-packages (from datasets) (0.13.3)
Collecting xxhash
  Downloading xxhash-3.2.0-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (212 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 212.2/212.2 KB 23.7 MB/s eta 0:00:00
Requirement already satisfied: tqdm>=4.62.1 in /usr/local/lib/python3.9/dist-packages (from datasets) (4.65.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.9/dist-packages (from datasets) (23.0)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.9/dist-packages (from datasets) (6.0)
Collecting responses<0.19
  Downloading responses-0.18.0-py3-none-any.whl (38 kB)
Collecting dill<0.3.7,>=0.3.0
  Downloading dill-0.3.6-py3-none-any.whl (110 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 110.5/110.5 KB 12.1 MB/s eta 0:00:00
Requirement already satisfied: pandas in /usr/local/lib/python3.9/dist-packages (from datasets) (1.4.4)
Collecting aiohttp
  Downloading aiohttp-3.8.4-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (1.0 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.0/1.0 MB 36.9 MB/s eta 0:00:00
Requirement already satisfied: pyarrow>=6.0.0 in /usr/local/lib/python3.9/dist-packages (from datasets) (9.0.0)
Collecting yarl<2.0,>=1.0
  Downloading yarl-1.8.2-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (264 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 264.6/264.6 KB 2.2 MB/s eta 0:00:00
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.9/dist-packages (from aiohttp->datasets) (22.2.0)
Collecting aiosignal>=1.1.2
  Downloading aiosignal-1.3.1-py3-none-any.whl (7.6 kB)
Requirement already satisfied: charset-normalizer<4.0,>=2.0 in /usr/local/lib/python3.9/dist-packages (from aiohttp->datasets) (2.0.1
Collecting async-timeout<5.0,>=4.0.0a3
  Downloading async_timeout-4.0.2-py3-none-any.whl (5.8 kB)
Collecting frozenlist>=1.1.1
  Downloading frozenlist-1.3.3-cp39-cp39-manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_17_x86_64.manylinux2014_x86_64.whl (158
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 158.8/158.8 KB 14.0 MB/s eta 0:00:00
Collecting multidict<7.0,>=4.5
  Downloading multidict-6.0.4-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (114 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 114.2/114.2 KB 2.7 MB/s eta 0:00:00
Requirement already satisfied: filelock in /usr/local/lib/python3.9/dist-packages (from huggingface-hub<1.0.0,>=0.2.0->datasets) (3.1
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.9/dist-packages (from huggingface-hub<1.0.0,>=0.
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.9/dist-packages (from requests>=2.19.0->datasets) (2022.1
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.9/dist-packages (from requests>=2.19.0->datasets) (1.
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.9/dist-packages (from requests>=2.19.0->datasets) (3.4)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.9/dist-packages (from pandas->datasets) (2022.7.1)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.9/dist-packages (from pandas->datasets) (2.8.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.9/dist-packages (from python-dateutil>=2.8.1->pandas->datasets) (1
Installing collected packages: xxhash, multidict, frozenlist, dill, async-timeout, yarl, responses, multiprocess, aiosignal, aiohttp
Successfully installed aiohttp-3.8.4 aiosignal-1.3.1 async-timeout-4.0.2 datasets-2.10.1 dill-0.3.6 frozenlist-1.3.3 multidict-6.0.4
```

```
import numpy as np
import pandas as pd
from transformers import AutoTokenizer, AutoModelForSequenceClassification, Trainer, TrainingArguments
from transformers import EvalPrediction
from sklearn.metrics import precision_recall_fscore_support, classification_report
from datasets import Dataset
```

```
import random

# ... (the rest of the functions remain the same)

# Replace these lines with the actual paths to your train and eval CSV files
# train_df = pd.read_csv("train.csv")
# eval_df = pd.read_csv("eval.csv")
def generate_negatives(df, multiplier=1):
    negative_df = df.copy()
    for _ in range(multiplier):
        negative_df['reason'] = negative_df['reason'].apply(lambda x: ' '.join(random.sample(x.split(), len(x.split()))))
    negative_df['label'] = 0
    return pd.concat([df, negative_df], ignore_index=True)

def preprocess_dataset(df, tokenizer):
    def encode(example):
        inputs = tokenizer(example['text'], example['reason'], padding=True, truncation=True, max_length=512, return_tensors='pt')
        return {k: v.squeeze(0) for k, v in inputs.items()}

    dataset = Dataset.from_pandas(df)
    dataset = dataset.map(encode, batched=True)
    dataset.set_format(type='torch', columns=['input_ids', 'attention_mask', 'label'])
    return dataset



def compute_metrics(eval_pred: EvalPrediction):
    predictions = eval_pred.predictions
    labels = eval_pred.label_ids
    preds = np.argmax(predictions, axis=1)
    precision, recall, f1, _ = precision_recall_fscore_support(labels, preds, average='binary')
    return {'precision': precision, 'recall': recall, 'f1': f1}

train_df
eval_df

train_df = generate_negatives(train_df, multiplier=1)

models = ['bert-base-uncased', 'distilbert-base-uncased', 'roberta-base']
model_results = {}

for model_name in models:
    print(f"Training and evaluating {model_name}")
    tokenizer = AutoTokenizer.from_pretrained(model_name)
    model = AutoModelForSequenceClassification.from_pretrained(model_name, num_labels=2)

    train_dataset = preprocess_dataset(train_df, tokenizer)
    eval_dataset = preprocess_dataset(eval_df, tokenizer)

    training_args = TrainingArguments(
        output_dir=f'./results/{model_name}',
        num_train_epochs=3,
        per_device_train_batch_size=8,
        per_device_eval_batch_size=8,
        evaluation_strategy='epoch',
        save_strategy='epoch',
        logging_dir=f'./logs/{model_name}',
    )

    trainer = Trainer(
        model=model,
        args=training_args,
        train_dataset=train_dataset,
        eval_dataset=eval_dataset,
        tokenizer=tokenizer,
        compute_metrics=compute_metrics,
    )

    trainer.train()  # This line was missing in your code

    # Error Analysis
    predictions = trainer.predict(eval_dataset)
    preds = np.argmax(predictions.predictions, axis=1)
    report = classification_report(eval_dataset['label'], preds, output_dict=True)
    model_results[model_name] = report

print("Error analysis:")
```

```
for model_name, report in model_results.items():
    print(f"Model: {model_name}")
    print(report)
```

```
Training and evaluating bert-base-uncased
Some weights of the model checkpoint at bert-base-uncased were not used when initializing BertForSequer
- This IS expected if you are initializing BertForSequenceClassification from the checkpoint of a mode
- This IS NOT expected if you are initializing BertForSequenceClassification from the checkpoint of a r
Some weights of BertForSequenceClassification were not initialized from the model checkpoint at bert-ba
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and int
You're using a BertTokenizerFast tokenizer. Please note that with a fast tokenizer, using the `__call__
```

[3093/3093 12:44, Epoch 3/3]

Epoch	Training Loss	Validation Loss	Precision	Recall	F1
1	0.176500	2.444418	0.333746	0.989004	0.499075
2	0.132100	3.138164	0.332625	0.991669	0.498159

```
import numpy as np
import pandas as pd
from transformers import AutoTokenizer, AutoModelForSequenceClassification, Trainer, TrainingArguments
from transformers import EvalPrediction
from sklearn.metrics import precision_recall_fscore_support, classification_report
from datasets import Dataset
import random

# ... (the rest of the functions remain the same)

def generate_negatives(df, multiplier=1):
    positive_df = df[df['label'] == 1]
    negative_df = df[df['label'] == 0].sample(len(positive_df), replace=True)
    balanced_df = pd.concat([positive_df, negative_df], ignore_index=True)
    return balanced_df.sample(frac=1).reset_index(drop=True)

def preprocess_dataset(df, tokenizer):
    def encode(example):
        inputs = tokenizer(example['text'], example['reason'], padding=True, truncation=True, max_length=512, return_tensors='pt')
        return {k: v.squeeze(0) for k, v in inputs.items()}

    dataset = Dataset.from_pandas(df)
    dataset = dataset.map(encode, batched=True)
    dataset.set_format(type='torch', columns=['input_ids', 'attention_mask', 'label'])
    return dataset



def compute_metrics(eval_pred: EvalPrediction):
    predictions = eval_pred.predictions
    labels = eval_pred.label_ids
    preds = np.argmax(predictions, axis=1)
    precision, recall, f1, _ = precision_recall_fscore_support(labels, preds, average='binary')
    return {'precision': precision, 'recall': recall, 'f1': f1}

# ... (preprocess_dataset and compute_metrics functions remain the same)

train_df
eval_df

train_df = generate_negatives(train_df, multiplier=1)

models = ['bert-base-uncased', 'distilbert-base-uncased', 'roberta-base']
model_results = {}

for model_name in models:
    print(f"Training and evaluating {model_name}")
    tokenizer = AutoTokenizer.from_pretrained(model_name)
    model = AutoModelForSequenceClassification.from_pretrained(model_name, num_labels=2)

    train_dataset = preprocess_dataset(train_df, tokenizer)
    eval_dataset = preprocess_dataset(eval_df, tokenizer)

    num_pos = len(train_df[train_df['label'] == 1])
    num_neg = len(train_df[train_df['label'] == 0])
    pos_weight = num_neg / num_pos

    training_args = TrainingArguments(
        output_dir=f'./results/{model_name}',
        num_train_epochs=3,
        per_device_train_batch_size=8,
        per_device_eval_batch_size=8,
        evaluation_strategy='epoch',
        save_strategy='epoch',
```

```
            logging_dir=f'./logs/{model_name}',
            report_to="none",
        )

        model.config.loss_function = 'CrossEntropyLoss'
        model.config.pos_weight = pos_weight

        trainer = Trainer(
            model=model,
            args=training_args,
            train_dataset=train_dataset,
            eval_dataset=eval_dataset,
            tokenizer=tokenizer,
            compute_metrics=compute_metrics,
        )

        trainer.train()

        # Error Analysis
        predictions = trainer.predict(eval_dataset)
        preds = np.argmax(predictions.predictions, axis=1)
        report = classification_report(eval_dataset['label'], preds, output_dict=True)
        model_results[model_name] = report

    print("Error analysis:")
    for model_name, report in model_results.items():
        print(f"Model: {model_name}")
        print(report)
    #This is to address the issue of poor class balance


    import numpy as np
    import pandas as pd
    from transformers import AutoTokenizer, AutoModelForSequenceClassification, Trainer, TrainingArguments
    from transformers import EvalPrediction
    from sklearn.metrics import precision_recall_fscore_support, classification_report
    from datasets import Dataset
    import random

    # ... (generate_negatives, preprocess_dataset, and compute_metrics functions remain the same)
    def generate_negatives(df, multiplier=1):
        positive_df = df[df['label'] == 1]
        negative_df = df[df['label'] == 0].sample(len(positive_df), replace=True)
        balanced_df = pd.concat([positive_df, negative_df], ignore_index=True)
        return balanced_df.sample(frac=1).reset_index(drop=True)

    def preprocess_dataset(df, tokenizer):
        def encode(example):
            inputs = tokenizer(example['text'], example['reason'], padding=True, truncation=True, max_length=512, return_tensors='pt')
            return {k: v.squeeze(0) for k, v in inputs.items()}

        dataset = Dataset.from_pandas(df)
        dataset = dataset.map(encode, batched=True)
        dataset.set_format(type='torch', columns=['input_ids', 'attention_mask', 'label'])
        return dataset



    def compute_metrics(eval_pred: EvalPrediction):
        predictions = eval_pred.predictions
        labels = eval_pred.label_ids
        preds = np.argmax(predictions, axis=1)
        precision, recall, f1, _ = precision_recall_fscore_support(labels, preds, average='binary')
        return {'precision': precision, 'recall': recall, 'f1': f1}



    train_df
    eval_df

    train_df = generate_negatives(train_df, multiplier=1)

    models = ['bert-base-uncased', 'distilbert-base-uncased', 'roberta-base', 'sentence-transformers/bert-base-nli-mean-tokens']
    model_results = {}

    for model_name in models:
        print(f"Training and evaluating {model_name}")
```

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

        train_dataset = preprocess_dataset(train_df, tokenizer)
        eval_dataset = preprocess_dataset(eval_df, tokenizer)

        num_pos = len(train_df[train_df['label'] == 1])
        num_neg = len(train_df[train_df['label'] == 0])
        pos_weight = num_neg / num_pos

        training_args = TrainingArguments(
            output_dir=f'./results/{model_name}',
            num_train_epochs=5,  # Increase the number of epochs
            learning_rate=2e-5,  # Fine-tune the learning rate
            per_device_train_batch_size=8,
            per_device_eval_batch_size=8,
            evaluation_strategy='epoch',
            save_strategy='epoch',
            logging_dir=f'./logs/{model_name}',
            report_to="none",
            lr_scheduler_type='cosine',  # Use cosine learning rate scheduler
        )

        model.config.loss_function = 'CrossEntropyLoss'
        model.config.pos_weight = pos_weight

        trainer = Trainer(
            model=model,
            args=training_args,
            train_dataset=train_dataset,
            eval_dataset=eval_dataset,
            tokenizer=tokenizer,
            compute_metrics=compute_metrics,
        )

        trainer.train()

        # Error Analysis
        predictions = trainer.predict(eval_dataset)
        preds = np.argmax(predictions.predictions, axis=1)
        report = classification_report(eval_dataset['label'], preds, output_dict=True)
        model_results[model_name] = report

    print("Error analysis:")
    for model_name, report in model_results.items():
        print(f"Model: {model_name}")
        print(report)
```

```
Training and evaluating bert-base-uncased
Some weights of the model checkpoint at bert-base-uncased were not used when initializing BertForSequer
- This IS expected if you are initializing BertForSequenceClassification from the checkpoint of a mode
- This IS NOT expected if you are initializing BertForSequenceClassification from the checkpoint of a r
Some weights of BertForSequenceClassification were not initialized from the model checkpoint at bert-ba
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and int
/usr/local/lib/python3.9/dist-packages/transformers/optimization.py:391: FutureWarning: This implementa
  warnings.warn(
You're using a BertTokenizerFast tokenizer. Please note that with a fast tokenizer, using the `__call__
```

[2580/2580 14:38, Epoch 5/5]

Epoch	Training Loss	Validation Loss	Precision	Recall	F1
1	0.255100	2.892632	0.332924	0.993002	0.498661
2	0.127500	4.043421	0.333259	0.992336	0.498953
3	0.083800	4.139670	0.333371	0.994335	0.499331
4	0.066200	4.345340	0.333408	0.993002	0.499204
5	0.057900	4.527711	0.333333	0.994002	0.499247

```
Training and evaluating distilbert-base-uncased
Some weights of the model checkpoint at distilbert-base-uncased were not used when initializing DistilB
- This IS expected if you are initializing DistilBertForSequenceClassification from the checkpoint of a
- This IS NOT expected if you are initializing DistilBertForSequenceClassification from the checkpoint
Some weights of DistilBertForSequenceClassification were not initialized from the model checkpoint at d
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and int
/usr/local/lib/python3.9/dist-packages/transformers/optimization.py:391: FutureWarning: This implementa
  warnings.warn(
You're using a DistilBertTokenizerFast tokenizer. Please note that with a fast tokenizer, using the `__
```

[2580/2580 07:28, Epoch 5/5]

Epoch	Training Loss	Validation Loss	Precision	Recall	F1
1	0.262600	3.250613	0.333184	0.993669	0.499038
2	0.116700	3.734712	0.333221	0.992003	0.498869
3	0.082100	3.919907	0.333743	0.996335	0.500000
4	0.070400	4.005634	0.333594	0.994668	0.499623
5	0.057000	4.144705	0.333445	0.994668	0.499456

```
Training and evaluating roberta-base
Some weights of the model checkpoint at roberta-base were not used when initializing RobertaForSequence
- This IS expected if you are initializing RobertaForSequenceClassification from the checkpoint of a mo
- This IS NOT expected if you are initializing RobertaForSequenceClassification from the checkpoint of
Some weights of RobertaForSequenceClassification were not initialized from the model checkpoint at robe
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and int
/usr/local/lib/python3.9/dist-packages/transformers/optimization.py:391: FutureWarning: This implementa
  warnings.warn(
You're using a RobertaTokenizerFast tokenizer. Please note that with a fast tokenizer, using the `__ca
```

[2580/2580 14:38, Epoch 5/5]

Epoch	Training Loss	Validation Loss	Precision	Recall	F1
1	0.262200	2.874046	0.334414	0.996335	0.500754
2	0.129600	3.738412	0.333778	0.999667	0.500459
3	0.094600	4.016052	0.333593	1.000000	0.500292
4	0.082000	4.448793	0.333964	0.999334	0.500626
5	0.071900	4.670692	0.334002	0.999334	0.500668

```
Training and evaluating sentence-transformers/bert-base-nli-mean-tokens
Downloading (…)okenizer_config.json: 100%  399/399 [00:00<00:00, 18.7kB/s]
Downloading (…)lve/main/config.json: 100%  625/625 [00:00<00:00, 28.9kB/s]
Downloading (…)solve/main/vocab.txt: 100%  232k/232k [00:00<00:00, 2.25MB/s]
Downloading (…)/main/tokenizer.json: 100%  466k/466k [00:00<00:00, 4.26MB/s]
Downloading (…)in/added_tokens.json: 100%  2.00/2.00 [00:00<00:00, 104B/s]
Downloading (…)cial_tokens_map.json: 100%  112/112 [00:00<00:00, 5.10kB/s]
```

```
import pandas as pd

# Prepare the data for the table
data = []
for model_name, report in model_results.items():
    precision = report['1']['precision']
    recall = report['1']['recall']
```

```
    f1_score = report['1']['f1-score']
    data.append([model_name, precision, recall, f1_score])

# Create the DataFrame and set the column names
results_df = pd.DataFrame(data, columns=['Model', 'Precision', 'Recall', 'F1 Score'])

# Display the DataFrame
print(results_df)
```

```
                                        Model  Precision    Recall  \
0                             bert-base-uncased   0.333333  0.994002
1                         distilbert-base-uncased   0.333445  0.994668
2                                 roberta-base   0.334002  0.999334
3  sentence-transformers/bert-base-nli-mean-tokens   0.333929  0.997001

   F1 Score
0  0.499247
1  0.499456
2  0.500668
3  0.500293
```

To analyze the errors made by each model and discuss possible reasons for these errors, you can start by examining the misclassified examples. This can be done by comparing the ground truth labels with the model's predictions. You can use the following code snippet to get the misclassified examples for each model:

Error Analysis

```
misclassified_examples = {}

for model_name in models:
    predictions = trainer.predict(eval_dataset)
    preds = np.argmax(predictions.predictions, axis=1)
    misclassified_indices = np.where(eval_dataset['label'] != preds)[0]
    misclassified_examples[model_name] = eval_df.iloc[misclassified_indices]
```

```
for model_name, misclassified_df in misclassified_examples.items():
    print(f"\nMisclassified examples for {model_name}:")
    print(misclassified_df)
```

```
 Misclassified examples for bert-base-uncased:
                                    text                       reason  label
 0  the app is crashing when i play a vedio  app crashes during playback      1

 Misclassified examples for distilbert-base-uncased:
                                    text                       reason  label
 0  the app is crashing when i play a vedio  app crashes during playback      1

 Misclassified examples for roberta-base:
                                    text                       reason  label
 0  the app is crashing when i play a vedio  app crashes during playback      1

 Misclassified examples for sentence-transformers/bert-base-nli-mean-tokens:
                                    text                       reason  label
 0  the app is crashing when i play a vedio  app crashes during playback      1
```

✓ 0s completed at 12:54 PM