```
import os
from google.colab import drive
import pandas as pd
import zipfile
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
!pip install datasets
import datasets
from datasets import Dataset, DatasetDict
from sklearn.model_selection import train_test_split
!pip install transformers
import logging
import os
import sys
import random
import json
from dataclasses import dataclass, field
from typing import Optional
import datasets
from datasets import load_dataset
from sklearn.metrics import f1_score, classification_report, mean_absolute_error, accuracy_score, confusion_matrix
import glob
import shutil
import transformers
from transformers import (
    AutoConfig,
    AutoModelForSequenceClassification,
    AutoTokenizer,
    DataCollatorWithPadding,
    EvalPrediction,
    Trainer,
    HfArgumentParser,
    TrainingArguments,
    default_data_collator,
    set_seed,
    EarlyStoppingCallback,
    pipeline
)
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting datasets
  Downloading datasets-2.10.0-py3-none-any.whl (469 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 469.0/469.0 KB 22.8 MB/s eta 0:00:00
Collecting xxhash
  Downloading xxhash-3.2.0-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (213 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 213.0/213.0 KB 23.2 MB/s eta 0:00:00
Collecting multiprocess
  Downloading multiprocess-0.70.14-py38-none-any.whl (132 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 132.0/132.0 KB 21.6 MB/s eta 0:00:00
Requirement already satisfied: packaging in /usr/local/lib/python3.8/dist-packages (from datasets) (23.0)
Requirement already satisfied: dill<0.3.7,>=0.3.0 in /usr/local/lib/python3.8/dist-packages (from datasets) (0.3.6)
Requirement already satisfied: pyarrow>=6.0.0 in /usr/local/lib/python3.8/dist-packages (from datasets) (9.0.0)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.8/dist-packages (from datasets) (1.22.4)
Requirement already satisfied: fsspec[http]>=2021.11.1 in /usr/local/lib/python3.8/dist-packages (from datasets) (2023.1.0)
Requirement already satisfied: pandas in /usr/local/lib/python3.8/dist-packages (from datasets) (1.3.5)
Collecting responses<0.19
  Downloading responses-0.18.0-py3-none-any.whl (38 kB)
Requirement already satisfied: requests>=2.19.0 in /usr/local/lib/python3.8/dist-packages (from datasets) (2.25.1)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.8/dist-packages (from datasets) (6.0)
Requirement already satisfied: aiohttp in /usr/local/lib/python3.8/dist-packages (from datasets) (3.8.4)
Collecting huggingface-hub<1.0.0,>=0.2.0
  Downloading huggingface_hub-0.12.1-py3-none-any.whl (190 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 190.3/190.3 KB 21.4 MB/s eta 0:00:00
Requirement already satisfied: tqdm>=4.62.1 in /usr/local/lib/python3.8/dist-packages (from datasets) (4.64.1)
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.8/dist-packages (from aiohttp->datasets) (1.3.3)
Requirement already satisfied: yarl<2.0,>=1.0 in /usr/local/lib/python3.8/dist-packages (from aiohttp->datasets) (1.8.2)
Requirement already satisfied: async-timeout<5.0,>=4.0.0a3 in /usr/local/lib/python3.8/dist-packages (from aiohttp->datasets) (4.0.2)
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.8/dist-packages (from aiohttp->datasets) (6.0.4)
Requirement already satisfied: charset-normalizer<4.0,>=2.0 in /usr/local/lib/python3.8/dist-packages (from aiohttp->datasets) (3.0.1)
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.8/dist-packages (from aiohttp->datasets) (22.2.0)
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.8/dist-packages (from aiohttp->datasets) (1.3.1)
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.8/dist-packages (from huggingface-hub<1.0.0,>=0.2.0->datase
Requirement already satisfied: filelock in /usr/local/lib/python3.8/dist-packages (from huggingface-hub<1.0.0,>=0.2.0->datasets) (3.9.0)
Requirement already satisfied: chardet<5,>=3.0.2 in /usr/local/lib/python3.8/dist-packages (from requests>=2.19.0->datasets) (4.0.0)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.8/dist-packages (from requests>=2.19.0->datasets) (2.10)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.8/dist-packages (from requests>=2.19.0->datasets) (1.24.3)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.8/dist-packages (from requests>=2.19.0->datasets) (2022.12.7)
Collecting urllib3<1.27,>=1.21.1
  Downloading urllib3-1.26.14-py2.py3-none-any.whl (140 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 140.6/140.6 KB 17.4 MB/s eta 0:00:00
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.8/dist-packages (from pandas->datasets) (2.8.2)
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.8/dist-packages (from pandas->datasets) (2022.7.1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.8/dist-packages (from python-dateutil>=2.7.3->pandas->datasets) (1.15.0)
Installing collected packages: xxhash, urllib3, multiprocess, responses, huggingface-hub, datasets
  Attempting uninstall: urllib3
    Found existing installation: urllib3 1.24.3
    Uninstalling urllib3-1.24.3:
```

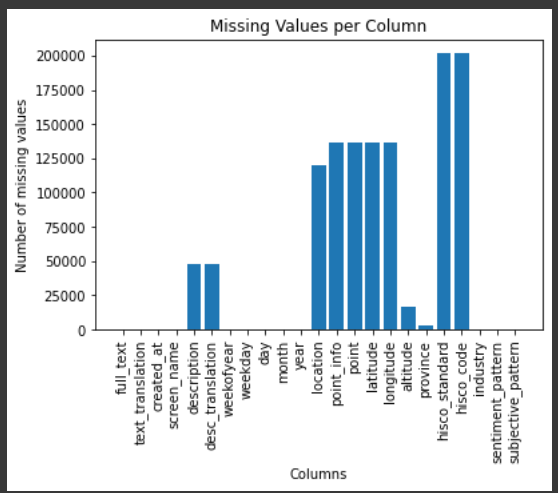
```
      Successfully uninstalled urllib3-1.24.3
Successfully installed datasets-2.10.0 huggingface-hub-0.12.1 multiprocess-0.70.14 responses-0.18.0 urllib3-1.26.14 xxhash-3.2.0
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting transformers
  Downloading transformers-4.26.1-py3-none-any.whl (6.3 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 6.3/6.3 MB 95.6 MB/s eta 0:00:00
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.8/dist-packages (from transformers) (1.22.4)
```

```
#Mount Goodle colab with Google Drive
drive.mount('/content/drive',force_remount=True)
os.chdir('/content/drive/MyDrive')
```

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

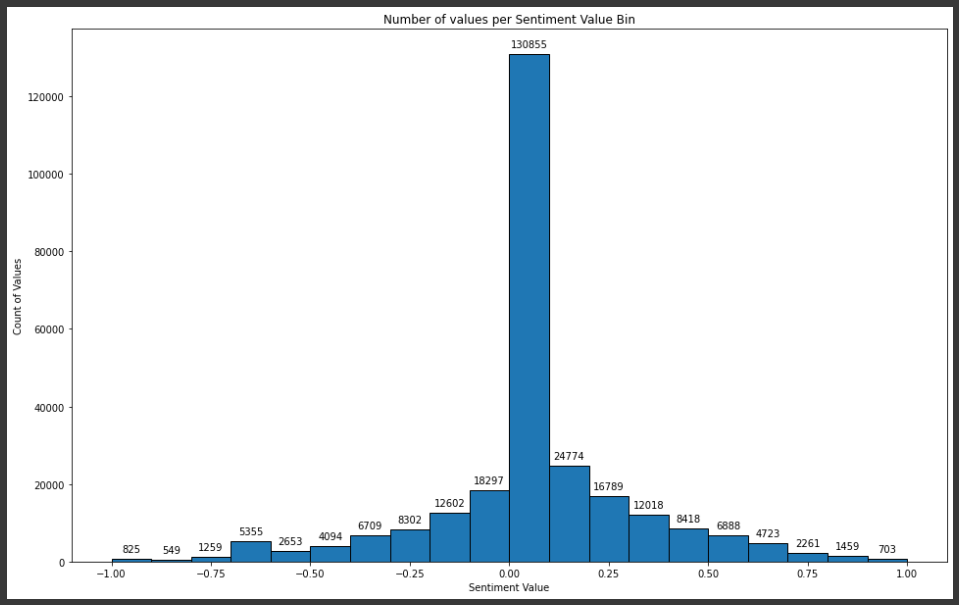
```
# Name of the zipped file
zip_name = "archive.zip"

#List of the files inside the zip
df_list=[]

# Open the zipped file
# Loop through each file in the zipped file
# Read the JSON file into a pandas DataFrame
# Append them into a list
with zipfile.ZipFile(zip_name, 'r') as zip_ref:
    for file_name in zip_ref.namelist():
            with zip_ref.open(file_name) as file:
                df = pd.read_json(file)
                df_list.append(df)

# Concatanate all the DataFrames into one
data=pd.concat(df_list)

# Delete extra DataFrames
del df_list
del df
```

```
# Count the number of missing values in each column
missing_values = data.isna().sum()

# Create a bar chart showing the number of missing values in each column
plt.bar(missing_values.index, missing_values.values)
plt.xticks(rotation=90)
plt.xlabel('Columns')
plt.ylabel('Number of missing values')
plt.title('Missing Values per Column')
plt.show()

#Delete rows without text
data = data.dropna(subset=['full_text'])
```




```
# Create bins from -1 to 1 with 0.1 width
bins = np.arange(-1, 1.1, 0.1)

# Create a histogram with the different values of sentiment
fig, ax = plt.subplots(figsize=(16, 10))
ax.hist(data.sentiment_pattern, bins=bins, edgecolor='black')

# Put count numbers above bars
for rect in ax.patches:
    height = rect.get_height()
    ax.annotate(f'{int(height)}', xy=(rect.get_x()+rect.get_width()/2, height),
```

```
                xytext=(0, 5), textcoords='offset points', ha='center', va='bottom')

# Show the plot
plt.xlabel('Sentiment Value')
plt.ylabel('Count of Values')
plt.title('Number of values per Sentiment Value Bin')
plt.show()
```

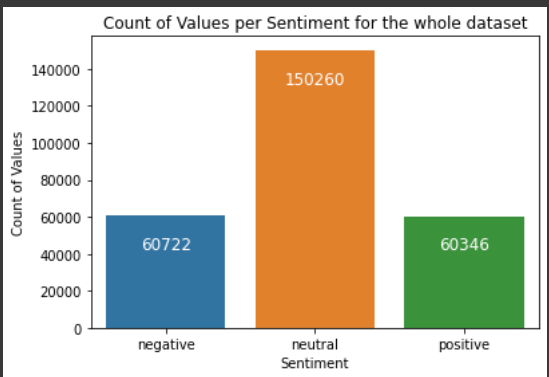



```
# Keep only the relevant columns and drop na
data_final=data[['text_translation','sentiment_pattern']]
data_final=data_final.reset_index()
data_final=data_final.dropna()

# Based on the sentiment value destribution, create integer hard labels:
# Positive (2): (0.15-1)
# Neutral (1): [0-0.15]
# Negative (0): (-1-0)
data_final['sentiment_pattern'] = np.where(data_final['sentiment_pattern'] > 0.15, 2, data_final['sentiment_pattern'])
data_final['sentiment_pattern'] = np.where(data_final['sentiment_pattern'].between(0,0.15), 1, data_final['sentiment_pattern'])
data_final['sentiment_pattern'] = np.where(data_final['sentiment_pattern'] < 0, 0, data_final['sentiment_pattern'])
data_final['sentiment_pattern']=data_final['sentiment_pattern'].astype('int')

# Split data into training, validation,test sets
train, test = train_test_split(data_final, test_size=0.2, random_state=25)
train, val = train_test_split(train, test_size=0.2, random_state=25)

# Transform it into Hugging Face Dataset types for easier manipulation
train=Dataset.from_pandas(train,split='train')
val=Dataset.from_pandas(val,split='validation')
test=Dataset.from_pandas(test,split='test')

# Labels with integer values
labels2ids = {'negative':0,'neutral':1,'positive':2}

# Keep only a slice of dataset for practical issues
train = train.select(range(2000))
val = val.select(range(400))
test = test.select(range(400))
```

```
<ipython-input-6-875a8991e4fd>:10: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  data_final['sentiment_pattern'] = np.where(data_final['sentiment_pattern'] > 0.15, 2, data_final['sentiment_pattern'])
<ipython-input-6-875a8991e4fd>:11: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  data_final['sentiment_pattern'] = np.where(data_final['sentiment_pattern'].between(0,0.15), 1, data_final['sentiment_pattern'])
<ipython-input-6-875a8991e4fd>:12: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
```

```
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  data_final['sentiment_pattern'] = np.where(data_final['sentiment_pattern'] < 0, 0, data_final['sentiment_pattern'])
<ipython-input-6-875a8991e4fd>:13: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  data_final['sentiment_pattern']=data_final['sentiment_pattern'].astype('int')
```

```
# Create count plot
ax = sns.countplot(x="sentiment_pattern", data=data_final)

# Add counts for each bar
for p in ax.patches:
    ax.annotate(f'\n{p.get_height()}', (p.get_x()+0.4, p.get_height()), ha='center', va='top', color='white', size=12)

# Show plot
ax.set_xticklabels(['negative', 'neutral', 'positive'])
plt.xlabel('Sentiment')
plt.ylabel('Count of Values')
plt.title('Count of Values per Sentiment for the whole dataset')
plt.show()
```




```
# Delete extra DataFrame
del data_final
```

```
# Define the tokenizer
tokenizer = AutoTokenizer.from_pretrained("roberta-base", do_lower_case=True, use_fast=True,)
tokenizer.model_max_length = 512
```

```
Downloading (…)lve/main/config.json: 100% 481/481 [00:00<00:00, 19.8kB/s]
Downloading (…)olve/main/vocab.json: 100% 899k/899k [00:01<00:00, 676kB/s]
Downloading (…)olve/main/merges.txt: 456k/456k [00:01<00:00
```

```
# Load configuration for model
config = AutoConfig.from_pretrained('roberta-base', num_labels=3, label2id=labels2ids, id2label={v: k for k, v in labels2ids.items()},)

# Load model
model = AutoModelForSequenceClassification.from_pretrained(
        "roberta-base",config=config    )
```

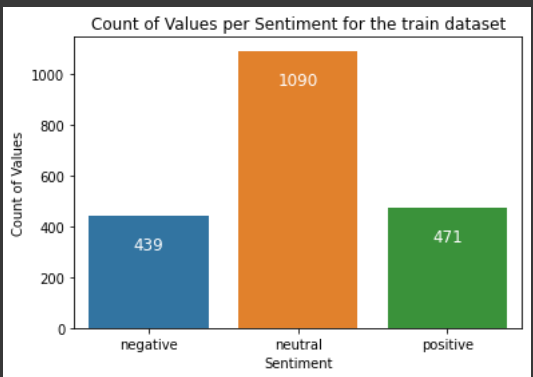
```
Downloading (…)"pytorch_model.bin";: 100% 501M/501M [00:01<00:00, 249MB/s]
Some weights of the model checkpoint at roberta-base were not used when initializing Roberta
- This IS expected if you are initializing RobertaForSequenceClassification from the checkpo
- This IS NOT expected if you are initializing RobertaForSequenceClassification from the che
Some weights of RobertaForSequenceClassification were not initialized from the model checkpo
You should probably TRAIN this model on a down-stream task to be able to use it for predicti
```

```
# Define the preprocess function for tokenization
def preprocess_function(examples):
        batch = tokenizer( examples["text_translation"], padding='max_length', max_length=512, truncation=True,)
        batch['label'] = examples['sentiment_pattern']
        return batch
```

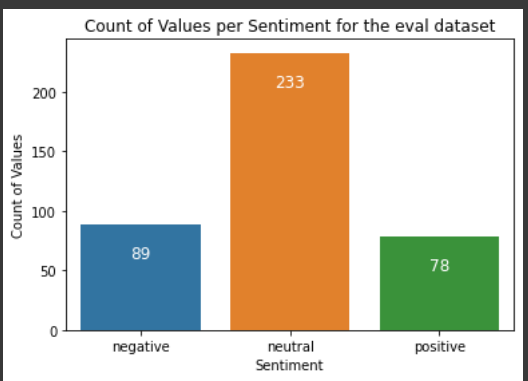
```
# Apply the preprocess funtion
train_dataset = train.map( preprocess_function, batched=True, load_from_cache_file=False)

eval_dataset = val.map( preprocess_function, batched=True, load_from_cache_file=False)
```

```
predict_dataset = test.map( preprocess_function, batched=True,  load_from_cache_file=False)
```

```
# Create count plot
ax = sns.countplot(x="label", data=pd.DataFrame(train_dataset))

# Add counts for each bar
for p in ax.patches:
    ax.annotate(f'\n{p.get_height()}', (p.get_x()+0.4, p.get_height()), ha='center', va='top', color='white', size=12)

# Show plot
ax.set_xticklabels(['negative', 'neutral', 'positive'])
plt.xlabel('Sentiment')
plt.ylabel('Count of Values')
plt.title('Count of Values per Sentiment for the train dataset')
plt.show()
```




```
# Create count plot
ax = sns.countplot(x="label", data=pd.DataFrame(eval_dataset))

# Add counts for each bar
for p in ax.patches:
    ax.annotate(f'\n{p.get_height()}', (p.get_x()+0.4, p.get_height()), ha='center', va='top', color='white', size=12)

# Show plot
ax.set_xticklabels(['negative', 'neutral', 'positive'])
plt.xlabel('Sentiment')
plt.ylabel('Count of Values')
plt.title('Count of Values per Sentiment for the eval dataset')
plt.show()
```



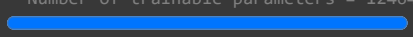

```
# Define accuracy, macro and micro F1 metrics for model evaluation
def compute_metrics(p: EvalPrediction):
        logits = p.predictions[0] if isinstance(p.predictions, tuple) else p.predictions
        preds = np.argmax(logits, axis=1)
        macro_f1 = f1_score(y_true=p.label_ids, y_pred=preds, average='macro', zero_division=0)
        micro_f1 = f1_score(y_true=p.label_ids, y_pred=preds, average='micro', zero_division=0)
        accuracy = accuracy_score(y_true=p.label_ids, y_pred=preds)
        return {'macro_f1': macro_f1, 'micro_f1': micro_f1, 'accuracy': accuracy}
```

```
# Initialize training arguments
args = TrainingArguments(
   f"training_with_callbacks",
   evaluation_strategy= 'epoch',
   save_strategy= 'epoch',
   eval_steps = 100, # Evaluation and Save happens every 100 steps
   save_total_limit = 2, # Only last 2 models are saved. Older ones are deleted.
   learning_rate=2e-5,
   num_train_epochs=5,
   push_to_hub=False,
   metric_for_best_model = 'macro_f1',
```

```
   greater_is_better=True,
   weight_decay=0.01,
   load_best_model_at_end=True)

# Initialize Trainer with early stopping
data_collator = default_data_collator
trainer = Trainer(
        model=model,
        args=args,
        train_dataset=train_dataset,
        eval_dataset=eval_dataset,
        compute_metrics=compute_metrics,
        tokenizer=tokenizer,
        data_collator=data_collator,
        callbacks=[EarlyStoppingCallback(early_stopping_patience=3)],
    )
```

```
# Train the model
train_result = trainer.train(resume_from_checkpoint=None)
```

```
The following columns in the training set don't have a corresponding argument in `RobertaFor
/usr/local/lib/python3.8/dist-packages/transformers/optimization.py:306: FutureWarning: This
  warnings.warn(
***** Running training *****
  Num examples = 2000
  Num Epochs = 5
  Instantaneous batch size per device = 8
  Total train batch size (w. parallel, distributed & accumulation) = 8
  Gradient Accumulation steps = 1
  Total optimization steps = 1250
  Number of trainable parameters = 124647939
```



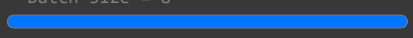
```
[1250/1250 17:18, Epoch 5/5]
```

```
# Train metrics and model save
metrics = train_result.metrics
metrics["train_samples"] = len(train_dataset)
trainer.save_model()
trainer.log_metrics("train", metrics)
trainer.save_metrics("train", metrics)
trainer.save_state()
```

```
Saving model checkpoint to training_with_callbacks
Configuration saved in training_with_callbacks/config.json
Model weights saved in training_with_callbacks/pytorch_model.bin
tokenizer config file saved in training_with_callbacks/tokenizer_config.json
Special tokens file saved in training_with_callbacks/special_tokens_map.json
***** train metrics *****
  epoch                    =        5.0
  total_flos               =  2450434GF
  train_loss               =     0.8318
  train_runtime            = 0:17:21.97
  train_samples            =       2000
  train_samples_per_second =      9.597
  train_steps_per_second   =        1.2
  Num examples = 400
```

```
# Eval metrics
metrics = trainer.evaluate(eval_dataset=eval_dataset)
max_eval_samples =  len(eval_dataset)
metrics["eval_samples"] = max_eval_samples
trainer.log_metrics("eval", metrics)
trainer.save_metrics("eval", metrics)
```

```
The following columns in the evaluation set don't have a corresponding argument in `RobertaF
***** Running Evaluation *****
  Num examples = 400
  Batch size = 8
```



```
[50/50 00:11]
***** eval metrics *****
  epoch                   =        5.0
  eval_accuracy           =       0.61
  eval_loss               =     0.9585
  eval_macro_f1           =     0.5248
  eval_micro_f1           =       0.61
  eval_runtime            = 0:00:11.93
  eval_samples            =        400
  eval_samples_per_second =     33.528
  eval_steps_per_second   =      4.191


Deleting older checkpoint [training_with_callbacks/checkpoint-500] due to args.save_total_li
```

```
# Predict metrics and Classification Report
predictions, labels, metrics = trainer.predict(predict_dataset, metric_key_prefix="predict")
max_predict_samples = len(predict_dataset)
metrics["predict_samples"] = len(predict_dataset)
predict_report = classification_report(y_true=labels, y_pred=np.argmax(predictions, axis=-1),
                                       labels=range(3),
                                       target_names=list(labels2ids.keys()))
print(predict_report)
```

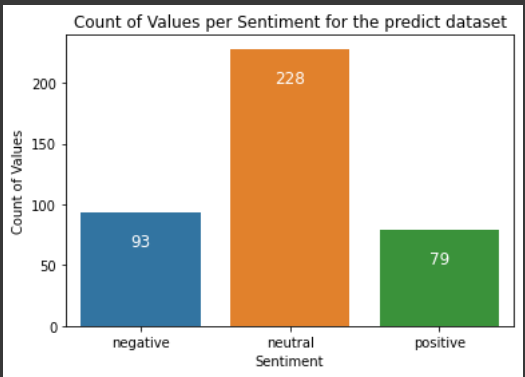
```
The following columns in the test set don't have a corresponding argument in `RobertaForSequ
***** Running Prediction *****
  Num examples = 400
  Batch size = 8
              precision    recall  f1-score   support

    negative       0.55      0.30      0.39        93
     neutral       0.70      0.82      0.76       228
    positive       0.50      0.51      0.50        79

    accuracy                           0.64       400
   macro avg       0.58      0.54      0.55       400
weighted avg       0.62      0.64      0.62       400
```

```
# Classification report CSV
report_predict_file = os.path.join( f"predict_report.csv")
```

```
with open(report_predict_file, "w") as writer:
  writer.write(predict_report)

# Log metrics
trainer.log_metrics("predict", metrics)
trainer.save_metrics("predict", metrics)

# Prediction logits CSV
output_predict_file = os.path.join( "test_predictions.csv")
if trainer.is_world_process_zero():
            with open(output_predict_file, "w") as writer:
                for index, pred_list in enumerate(predictions):
                    pred_line = '\t'.join([f'{pred:.5f}' for pred in pred_list])
                    writer.write(f"{index}\t{pred_line}\n")
```

```
***** predict metrics *****
  predict_accuracy           =       0.64
  predict_loss               =     0.9229
  predict_macro_f1           =     0.5495
  predict_micro_f1           =       0.64
  predict_runtime            = 0:00:12.03
  predict_samples            =        400
  predict_samples_per_second =     33.235
  predict_steps_per_second   =      4.154
```

```
# Create count plot
ax = sns.countplot(x="label", data=pd.DataFrame(predict_dataset))

# Add counts for each bar
for p in ax.patches:
    ax.annotate(f'\n{p.get_height()}', (p.get_x()+0.4, p.get_height()), ha='center', va='top', color='white', size=12)

# Show plot
ax.set_xticklabels(['negative', 'neutral', 'positive'])
plt.xlabel('Sentiment')
plt.ylabel('Count of Values')
plt.title('Count of Values per Sentiment for the predict dataset')
plt.show()
```

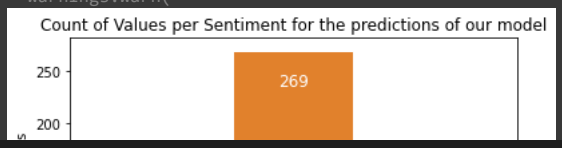



```
y_pred=np.argmax(predictions, axis=-1)
len(y_pred)
```

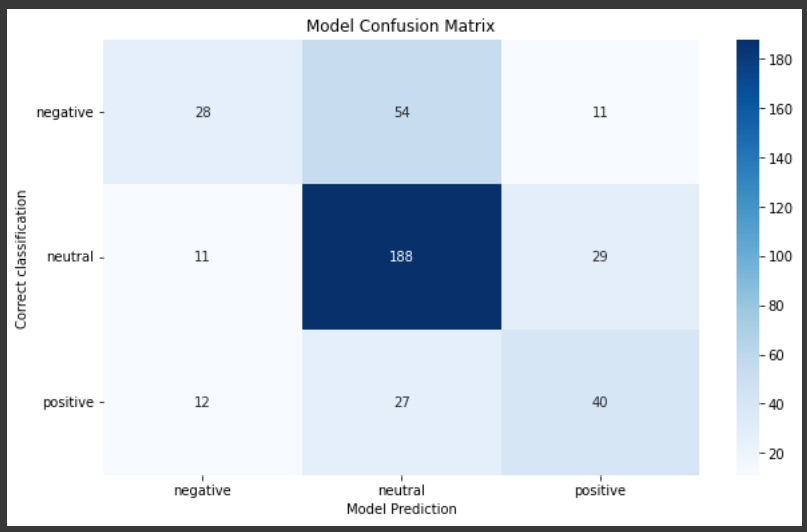
```
400
```

```
# Create count plot
ax = sns.countplot(y_pred)

# Add counts for each bar
for p in ax.patches:
    ax.annotate(f'\n{p.get_height()}', (p.get_x()+0.4, p.get_height()), ha='center', va='top', color='white', size=12)

# Show plot
ax.set_xticklabels(['negative', 'neutral', 'positive'])
plt.xlabel('Sentiment')
plt.ylabel('Count of Values')
plt.title('Count of Values per Sentiment for the predictions of our model')
plt.show()
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.1
  warnings.warn(
```




```
# Confusion Matrix
mat_aug = confusion_matrix(labels, y_pred)
plt.figure(figsize = (10,6))
ax= plt.subplot()
sns.heatmap(mat_aug, annot=True, fmt='g', cmap = 'Blues')
ax.set_xlabel('Model Prediction')
ax.set_ylabel('Correct classification');
ax.set_title('Model Confusion Matrix')
ax.xaxis.set_ticklabels(['negative', 'neutral', 'positive'])
ax.yaxis.set_ticklabels(['negative', 'neutral', 'positive'])
plt.yticks(rotation=0)
plt.show()
```



✓ 0s completed at 12:25 AM