Natural Language processing

TP – Sentiment analysis on BERT

This tutorial provides a close look at BERT (Bidirectional Encoder Representations from Transformers), one of the most popular transformer-based models, and instructs on its use in practice for sentiment analysis by fine-tuning the base model.

1. Import packages

```
!pip install datasets

import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix, roc_curve, auc
from datasets import load_dataset
from transformers import AutoTokenizer, AutoModelForSequenceClassification, TrainingArguments, Trainer

# Variables to set the number of epochs and samples
num_epochs = 10
num_samples = 100  # set this to -1 to use all data
```

(maybe setup less epochs soi t runs faster)

2. Load dataset

```
# Step 1: Load dataset and model tokenizer
dataset = load_dataset('imdb')
tokenizer = AutoTokenizer.from_pretrained('bert-base-uncased')
```

3. Explore the data

```
# Data Exploration
train_df = pd.DataFrame(dataset["train"])
sns.countplot(x='label', data=train_df)
plt.title('Class distribution')
plt.show()
```

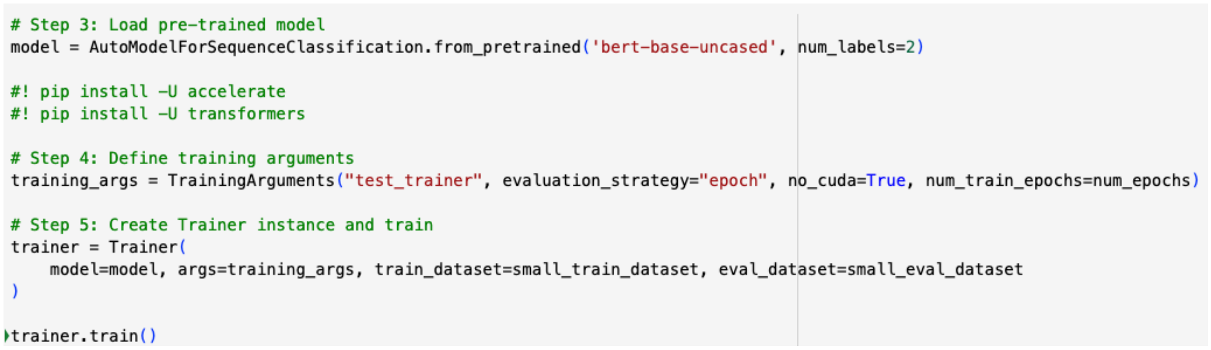
4. Pre treatment, mostly tokenizer

```
# Step 2: Preprocess the dataset
def tokenize_function(examples):
    return tokenizer(examples["text"], padding="max_length", truncation=True)

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

5. Train – test data. With num_smaples = -1, all data is used as train

```
[7] if num_samples == -1:
        small_train_dataset = tokenized_datasets["train"].shuffle(seed=42)
        small_eval_dataset = tokenized_datasets["test"].shuffle(seed=42)
    else:
        small_train_dataset = tokenized_datasets["train"].shuffle(seed=42).select(range(num_samples))
        small_eval_dataset = tokenized_datasets["test"].shuffle(seed=42).select(range(num_samples))
```

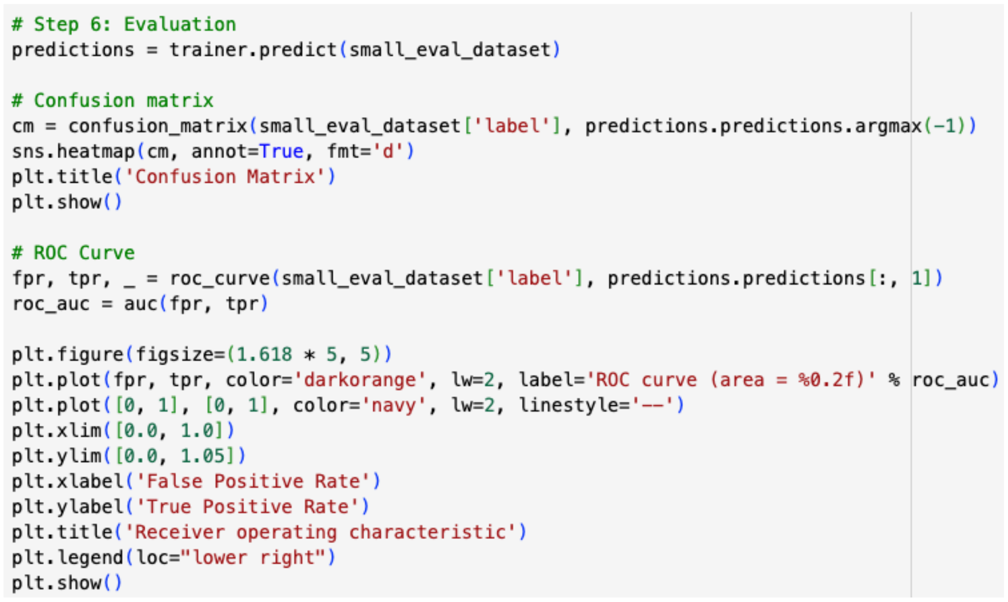
6. Load pre trained model and run the training. For this tutorial, we use the 'bert-base-uncased' version of BERT, which is trained on lower-case English text, is used for this tutorial.



```
# Step 3: Load pre-trained model
model = AutoModelForSequenceClassification.from_pretrained('bert-base-uncased', num_labels=2)

#! pip install -U accelerate
#! pip install -U transformers

# Step 4: Define training arguments
training_args = TrainingArguments("test_trainer", evaluation_strategy="epoch", no_cuda=True, num_train_epochs=num_epochs)

# Step 5: Create Trainer instance and train
trainer = Trainer(
    model=model, args=training_args, train_dataset=small_train_dataset, eval_dataset=small_eval_dataset
)

trainer.train()
```

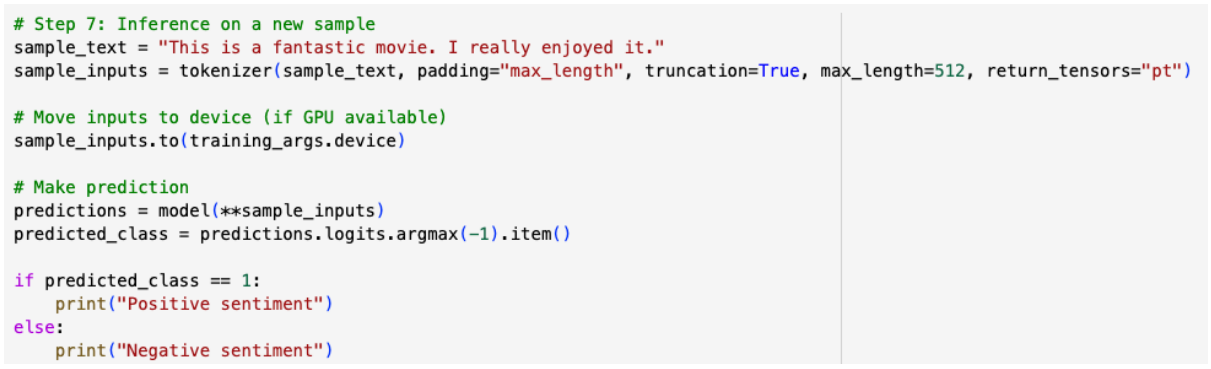
- Pip install commands can be used if an error comes from Trainer
- If you do the 10 epochs, i twill last more than 2 hours to run

7. Having trained our model, let's evaluate it.



```
# Step 6: Evaluation
predictions = trainer.predict(small_eval_dataset)

# Confusion matrix
cm = confusion_matrix(small_eval_dataset['label'], predictions.predictions.argmax(-1))
sns.heatmap(cm, annot=True, fmt='d')
plt.title('Confusion Matrix')
plt.show()

# ROC Curve
fpr, tpr, _ = roc_curve(small_eval_dataset['label'], predictions.predictions[:, 1])
roc_auc = auc(fpr, tpr)

plt.figure(figsize=(1.618 * 5, 5))
plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (area = %0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic')
plt.legend(loc="lower right")
plt.show()
```

8. Let’s see how it works with a particular example



```
# Step 7: Inference on a new sample
sample_text = "This is a fantastic movie. I really enjoyed it."
sample_inputs = tokenizer(sample_text, padding="max_length", truncation=True, max_length=512, return_tensors="pt")

# Move inputs to device (if GPU available)
sample_inputs.to(training_args.device)

# Make prediction
predictions = model(**sample_inputs)
predicted_class = predictions.logits.argmax(-1).item()

if predicted_class == 1:
    print("Positive sentiment")
else:
    print("Negative sentiment")
```