Fine Tuning Transformer (DistilBert, but generic to other models) for MultiClass Text Classification (Sentimental Analysis over IMDb)

Poc Introduction

Text classification is a common NLP task that assigns a label or class to text. Some of the largest companies run text classification in production for a wide range of practical applications. One of the most popular forms of text classification is sentiment analysis, which assigns a label like 🙂 positive, 🙁 negative, or 😐 neutral to a sequence of text.

This Poc will show you how to:

- Finetune DistilBERT on the IMDb dataset to determine whether a movie review is positive or negative.
- Use your finetuned model for inference (supports several model architectures...)

Flow of the notebook

The notebook will be divided into separate sections to provide a organized walk through for the process used. This process can be modified for individual use cases. The sections are:

1. Importing Python Libraries and preparing the environment
2. Importing and Pre-Processing the domain data
3. Preparing the Dataset
4. Creating the Neural Network for Fine Tuning
5. Fine Tuning and validating the Model
6. Saving the model and artifacts for Inference in Future (tbd)

Technical Details

This script leverages on multiple tools designed by other teams. Details of the tools used below. Please ensure that these elements are present in your setup to successfully implement this script.

- Data:
  - We are using IMDb data from datasets library Repository
  - Where each row has the following data-point:
    - label (0 negative, 1 positive)
    - text
- Language Model Used:
  - DistilBERT this is a smaller transformer model as compared to BERT or Roberta. It is created by process of distillation applied to Bert.
  - Blog-Post
  - Research Paper
  - Documentation for python
- Supports several model architectures...
- Hardware Requirements:
  - Python 3.6 and above
  - Pytorch, Transformers and All the stock Python ML Libraries
  - GPU enabled setup
- Script Objective:
  - The objective of this script is to fine tune DistilBERT (but generic to severall models) on the IMDb dataset to determine whether a movie review is positive or negative (label 0 "negative", 1 "positive").

Importing Python Libraries and preparing the environment

At this step we will be importing the libraries and modules needed to run our script. Libraries are:

- Numpy
- Pandas
- Pytorch
- Dataset
- Transformers
- AutoTokenizer, AutoModelForSequenceClassification, TrainingArguments, Trainer

- Evaluate

Followed by that we will prepare the device for CUDA execution. This configuration is needed if you want to leverage on onboard GPU.

```
# Installing libraries (Transformers, datasets, evaluate, torch) needed
!pip install transformers
!pip install datasets
!pip install evaluate
!pip install transformers[torch] #due to torch accelerator


# Importing the libraries needed
import numpy as np
import pandas as pd
import torch
import transformers
from datasets import load_dataset, Dataset #from torch.utils.data import Dataset
from transformers import AutoTokenizer, AutoModelForSequenceClassification, AutoModelForPreTraining, TrainingArguments,  Trainer, Data
import evaluate
import huggingface_hub

```

```
Requirement already satisfied: fsspec[http]>=2021.05.0 in /usr/local/lib/python3.10/dist-packages (from evaluate) (2023.6.0)
Requirement already satisfied: huggingface-hub>=0.7.0 in /usr/local/lib/python3.10/dist-packages (from evaluate) (0.17.1)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from evaluate) (23.1)
Collecting responses<0.19 (from evaluate)
  Downloading responses-0.18.0-py3-none-any.whl (38 kB)
Requirement already satisfied: pyarrow>=8.0.0 in /usr/local/lib/python3.10/dist-packages (from datasets>=2.0.0->evaluate) (9.0.0
Requirement already satisfied: aiohttp in /usr/local/lib/python3.10/dist-packages (from datasets>=2.0.0->evaluate) (3.8.5)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-packages (from datasets>=2.0.0->evaluate) (6.0.1)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from huggingface-hub>=0.7.0->evaluate) (3.12
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.10/dist-packages (from huggingface-hub>=0.7.
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests>=2.19.0->evalu
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests>=2.19.0->evaluate) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests>=2.19.0->evaluate) (
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests>=2.19.0->evaluate) (
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas->evaluate) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas->evaluate) (2023.3.post1)
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets>=2.0.0->evaluate
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets>=2.0.0->ev
Requirement already satisfied: async-timeout<5.0,>=4.0.0a3 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets>=2
Requirement already satisfied: yarl<2.0,>=1.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets>=2.0.0->evaluat
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets>=2.0.0->eval
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets>=2.0.0->evalu
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.1->pandas->evaluat
Installing collected packages: responses, evaluate
Successfully installed evaluate-0.4.0 responses-0.18.0
Requirement already satisfied: transformers[torch] in /usr/local/lib/python3.10/dist-packages (4.33.1)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from transformers[torch]) (3.12.2)
Requirement already satisfied: huggingface-hub<1.0,>=0.15.1 in /usr/local/lib/python3.10/dist-packages (from transformers[torch]
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist-packages (from transformers[torch]) (1.23.5)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from transformers[torch]) (23.1)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-packages (from transformers[torch]) (6.0.1)
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.10/dist-packages (from transformers[torch]) (2023.6.3
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from transformers[torch]) (2.31.0)
Requirement already satisfied: tokenizers!=0.11.3,<0.14,>=0.11.1 in /usr/local/lib/python3.10/dist-packages (from transformers[t
Requirement already satisfied: safetensors>=0.3.1 in /usr/local/lib/python3.10/dist-packages (from transformers[torch]) (0.3.3)
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.10/dist-packages (from transformers[torch]) (4.66.1)
Requirement already satisfied: torch!=1.12.0,>=1.10 in /usr/local/lib/python3.10/dist-packages (from transformers[torch]) (2.0.1
Collecting accelerate>=0.20.3 (from transformers[torch])
  Downloading accelerate-0.23.0-py3-none-any.whl (258 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 258.1/258.1 kB 5.4 MB/s eta 0:00:00
Requirement already satisfied: psutil in /usr/local/lib/python3.10/dist-packages (from accelerate>=0.20.3->transformers[torch])
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packages (from huggingface-hub<1.0,>=0.15.1->transformer
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.10/dist-packages (from huggingface-hub<1.0,>
Requirement already satisfied: sympy in /usr/local/lib/python3.10/dist-packages (from torch!=1.12.0,>=1.10->transformers[torch])
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-packages (from torch!=1.12.0,>=1.10->transformers[torc
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages (from torch!=1.12.0,>=1.10->transformers[torch]
Requirement already satisfied: triton==2.0.0 in /usr/local/lib/python3.10/dist-packages (from torch!=1.12.0,>=1.10->transformers
Requirement already satisfied: cmake in /usr/local/lib/python3.10/dist-packages (from triton==2.0.0->torch!=1.12.0,>=1.10->trans
Requirement already satisfied: lit in /usr/local/lib/python3.10/dist-packages (from triton==2.0.0->torch!=1.12.0,>=1.10->transfo
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->transformers[
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->transformers[torch]) (3.4
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests->transformers[torch]
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests->transformers[torch]
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from jinja2->torch!=1.12.0,>=1.10->tr
Requirement already satisfied: mpmath>=0.19 in /usr/local/lib/python3.10/dist-packages (from sympy->torch!=1.12.0,>=1.10->transf
Installing collected packages: accelerate
Successfully installed accelerate-0.23.0
```

```
# Setting up the device for GPU usage
from torch import cuda
device = 'cuda' if cuda.is_available() else 'cpu'
print ("Using device ", device)
```

```
Using device  cuda
```

Importing and Preparing the domain data

We will be working with the data and preparing for fine tuning purposes. *From DataSets IMDb*.

The Dataset will be something like this:

label	text
0	text_1
1	text_2
1	text_3
0	text_4

There are two fields in this dataset:

- text: the movie review text.
- label: a value that is either 0 for a negative review or 1 for a positive review.

If you like, you can create a smaller subset of the full dataset to fine-tune on to reduce the time it takes.

```
#Hugging face login
from huggingface_hub import notebook_login

notebook_login()

# Import IMDb dataset
imdb = load_dataset("imdb")

# option to reduce dataset size
# imdb = imdb.select (range(1000))


```

```
Token is valid (permission: write).
Your token has been saved in your configured git credential helpers (store)
Your token has been saved to /root/.cache/huggingface/token
Login successful
```

```
Downloading builder script: 100%  4.31k/4.31k [00:00<00:00, 176kB/s]
Downloading metadata: 100%  2.17k/2.17k [00:00<00:00, 131kB/s]
Downloading readme: 100%  7.59k/7.59k [00:00<00:00, 450kB/s]
Downloading data: 100%  84.1M/84.1M [00:09<00:00, 16.6MB/s]
Generating train split: 100%  25000/25000 [00:08<00:00, 5514.40 examples/s]
Generating test split: 100%  25000/25000 [00:06<00:00, 9780.38 examples/s]
Generating unsupervised split: 100%  50000/50000 [00:09<00:00, 8964.22 examples/s]
```

Pre-Processing the Dataset

We will start with defining few key variables that will be used later during the training/fine tuning stage.

Dataset Tokenization

- We are using the DistilBERT tokenizer to tokenize the data
- To process your dataset in one step, use Datasets map method to apply a preprocessing function over the entire dataset
- Format pytorch ("torch")
- Create a batch of examples using DataCollatorWithPadding
- To read further into the tokenizer, [refer to this document]

```
# Defining some key variables that will be used later on in the training
TRAIN_BATCH_SIZE = 16
VALID_BATCH_SIZE = 16
EPOCHS = 3
LEARNING_RATE = 2e-05
MAXIMUM_SEQ_LENGHT = 256
WEIGHT_DECAY = 0.01
id2label = {0: "NEGATIVE", 1: "POSITIVE"}
label2id = {"NEGATIVE": 0, "POSITIVE": 1}

# Dataset Tokenization
modelnameused=  "distilbert-base-uncased"
```

```
modelnameused = "distilbert-base-uncased"
tokenizer = AutoTokenizer.from_pretrained(modelnameused)

def preprocess_function(examples):
    return tokenizer(examples["text"], truncation=True)

tokenized_imdb = imdb.map(preprocess_function, batched=True)

#create a batch of examples using DataCollatorWithPadding. It's more efficient to dynamically pad the sentences to the longest length
data_collator = DataCollatorWithPadding(tokenizer=tokenizer)

#tokenized_contratsAPFit = tokenized_contratsAPFit.with_format ("torch")
```

```
Downloading (…)okenizer_config.json: 100%  28.0/28.0 [00:00<00:00, 230B/s]
Downloading (…)lve/main/config.json: 100%  483/483 [00:00<00:00, 27.5kB/s]
Downloading (…)solve/main/vocab.txt: 100%  232k/232k [00:00<00:00, 1.41MB/s]
Downloading (…)/main/tokenizer.json: 100%  466k/466k [00:00<00:00, 2.51MB/s]
Map: 100%  25000/25000 [00:23<00:00, 971.16 examples/s]
Map: 100%  25000/25000 [00:22<00:00, 1250.81 examples/s]
Map: 100%  50000/50000 [00:46<00:00, 905.40 examples/s]
```

Creating the Neural Network for Fine Tuning

Neural Network

- We will be creating a neural network with the `distilbert-base-cased`.
- Defining training_args and compute metrics
- You'll push this model to the Hub by setting push_to_hub=True (you need to be signed in to Hugging Face to upload your model). At the end of each epoch, the Trainer will evaluate the accuracy and save the training checkpoint.

```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

def draw_confusion_matrix (predictions , labels):
  print ("Confusion Matrix: Focus on  classification Challenges")
  cm = confusion_matrix(labels, predictions)
  ConfusionMatrixDisplay(cm).plot()

metric = evaluate.load("accuracy")
#accuracy = evaluate.load("accuracy")

#Call compute on metric to calculate the accuracy of your predictions. Before passing your predictions to compute, you need to convert
def compute_metrics(eval_pred):
    logits, labels = eval_pred
    predictions = np.argmax(logits, axis=-1)
    print ("predictions ", predictions)
    print ("labels      ", labels)
    draw_confusion_matrix (predictions , labels)
    return metric.compute(predictions=predictions, references=labels)

# Train with AutoModelForSequenceClassification
model = AutoModelForSequenceClassification.from_pretrained(
    modelnameused, num_labels=2, id2label=id2label, label2id=label2id
)


training_args = TrainingArguments (
    output_dir="test_trainer",
    evaluation_strategy="epoch",
    per_device_train_batch_size=TRAIN_BATCH_SIZE, #16,
    per_device_eval_batch_size=VALID_BATCH_SIZE, #16,
    num_train_epochs=EPOCHS, #3,
    learning_rate=LEARNING_RATE, #2e-5, #1e-5
    # maximum sequence length = MAXIMUM_SEQ_LENGHT
    weight_decay=WEIGHT_DECAY, #0.01,
    save_strategy="epoch",
    load_best_model_at_end=True,
    push_to_hub=True,
)

```

```
Downloading builder script: 100%                    4.20k/4.20k [00:00<00:00, 257kB/s]
Downloading model.safetensors: 100%                 268M/268M [00:04<00:00, 31.4MB/s]
```

Fine Tuning and validating the Model

After all the effort of loading and preparing the data and datasets, creating the model and defining its variabels. This is probably the easier steps in the process.

Here we create a Trainer object with your model, training arguments, training and test datasets, and evaluation function.

During the validation stage we pass the unseen data(Testing Dataset) to the model. This step determines how good the model performs on the unseen data.

See Confusion Matrix of last prediction iteraton

As you can see the model is predicting the correct sentiment ~ 93% accuracy.

```
#Create a Trainer object with your distilbert model, training arguments, training and test datasets, and evaluation function:
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=tokenized_imdb["train"],
    eval_dataset=tokenized_imdb["test"],
    tokenizer=tokenizer,
    data_collator=data_collator,
    compute_metrics=compute_metrics,
)

#Then fine-tune your model by calling train():
trainer.train()

#Once training is completed, share your model to the Hub with the push_to_hub() method so everyone can use your model:
#trainer.push_to_hub()
```