# Fine Tuning Transformer (DistilBert) for MultiClass Text Classification (Project types based on description)

## Introduction

In this project we will be fine tuning a transformer model (DistilBert) for the **Multiclass text classification** problem. Given a Project description the model will classify into one of the project categories out of the given list.

### 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 (to be implement)

### 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 Project Descriptions from Portuguese Public Administration site of Contract aggregator dataset available at base.gov Repository
    - We are referring only to a small example csv file from the data dump: `ContratosAP_v5.2_TrainPred.xlsx`
    - There are `751` rows of data. Where each row has the following data-point:
        - Objeto do Contrato: Contract Object
        - Contrato (Tipo): Type of Contract (see categories below)
- 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

- 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 project

## 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 execeution. This configuration is needed if you want to leverage on onboard GPU.

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

# Importing the libraries needed
import numpy as np
import pandas as pd
import torch
import transformers
from datasets import Dataset #from torch.utils.data import Dataset
from transformers import AutoTokenizer, AutoModelForSequenceClassification, AutoModelFo
import evaluate

```

```
Collecting transformers
  Downloading transformers-4.33.1-py3-none-any.whl (7.6 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 7.6/7.6 MB 50.0 MB/s eta 0:00:00
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-package
Collecting huggingface-hub<1.0,>=0.15.1 (from transformers)
  Downloading huggingface_hub-0.17.1-py3-none-any.whl (294 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 294.8/294.8 kB 30.4 MB/s eta 0:00:00
```

```
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist-pack
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-pack
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.10/dis
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-package
Collecting tokenizers!=0.11.3,<0.14,>=0.11.1 (from transformers)
  Downloading tokenizers-0.13.3-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 7.8/7.8 MB 110.2 MB/s eta 0:00:00
Collecting safetensors>=0.3.1 (from transformers)
  Downloading safetensors-0.3.3-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.3/1.3 MB 81.1 MB/s eta 0:00:00
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.10/dist-packa
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/pytho
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-pac
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/di
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/di
Installing collected packages: tokenizers, safetensors, huggingface-hub, transfor
Successfully installed huggingface-hub-0.17.1 safetensors-0.3.3 tokenizers-0.13.3
Collecting datasets
  Downloading datasets-2.14.5-py3-none-any.whl (519 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 519.6/519.6 kB 4.3 MB/s eta 0:00:00
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist-pack
Requirement already satisfied: pyarrow>=8.0.0 in /usr/local/lib/python3.10/dist-p
Collecting dill<0.3.8,>=0.3.0 (from datasets)
  Downloading dill-0.3.7-py3-none-any.whl (115 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 115.3/115.3 kB 3.6 MB/s eta 0:00:00
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: requests>=2.19.0 in /usr/local/lib/python3.10/dist
Requirement already satisfied: tqdm>=4.62.1 in /usr/local/lib/python3.10/dist-pac
Collecting xxhash (from datasets)
  Downloading xxhash-3.3.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 194.1/194.1 kB 20.1 MB/s eta 0:00:00
Collecting multiprocess (from datasets)
  Downloading multiprocess-0.70.15-py310-none-any.whl (134 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 134.8/134.8 kB 15.2 MB/s eta 0:00:00
Requirement already satisfied: fsspec[http]<2023.9.0,>=2023.1.0 in /usr/local/lib
Requirement already satisfied: aiohttp in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: huggingface-hub<1.0.0,>=0.14.0 in /usr/local/lib/p
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packag
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-pack
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.10/dist-pa
Requirement already satisfied: charset-normalizer<4.0,>=2.0 in /usr/local/lib/pyt
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.10/d
Requirement already satisfied: async-timeout<5.0,>=4.0.0a3 in /usr/local/lib/pyth
Requirement already satisfied: yarl<2.0,>=1.0 in /usr/local/lib/python3.10/dist-p
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.10/dis
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.10/dist
```

```
1 # Setting up the device for GPU usage
2 from torch import cuda
3 device = 'cuda' if cuda.is_available() else 'cpu'
4 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. *Assuming that the `ContratosAP_v5.2_TrainPred.xlsx` is already downloaded in your `data` folder.*

Import the file in a dataframe. Remove the unwanted columns, rename labels. The final Dataframe will be something like this:

| Objeto do Contrato | Contrato (Tipo) |
|---|---|
| contract_description_1 | 2 |
| contract_description_2 | 6 |
| contract_description_3 | 3 |
| contract_description_4 | 1 |
| contract_description_5 | 5 |
| contract_description_6 | 7 |

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

```
# Import the xls into pandas dataframe
PathFileName = '/content/sample_data/ContratosAP_v5.2_TrainPred.xlsx'
contratsAPFit = pd.read_excel(PathFileName,'DadosTreinoVal') #all dataset with contract
# Removing unwanted columns and only leaving project description and the category which
contratsAPFit = contratsAPFit[['Objeto do Contrato','Contrato (Tipo)']]

# Create a dataset from the pandas dataframe
dataset = Dataset.from_pandas(contratsAPFit)
# Rename column name (labels for Transformer Model)
dataset = dataset.rename_column ("Contrato (Tipo)", "labels")

# option to reduce dataset size (Max 751)
dataset = dataset.select (range(751))


```

## Pre-Processing the Dataset

We will start with defining few key variables that will be used later during the training/fine tuning stage. Followed by Spliting the dataset into a dictionary with dataset train and test (for the Transformer model).

- *Training Dataset* is used to fine tune the model: **80% of the original data**
- *Validation Dataset* is used to evaluate the performance of the model. The model has not seen this data during training.

### 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
- The tokenizer perform tokenization and generate the necessary outputs, namely: `ids`, `attention_mask`
- Remove "Objeto do Contrato" column
- `label` is the encoded category on the project description
- Format pytorch ("torch")
- 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

# split the dataset into a dictionary with dataset train and test
dataset = dataset.train_test_split(test_size=0.2) #, stratify_by_column="Labels")
print("dataset ", dataset)

# Dataset Tokenization
modelnameused=  "distilbert-base-multilingual-cased" #49% accuracy #"adalbertojunior/di
tokenizer = AutoTokenizer.from_pretrained(modelnameused)   # do_lower_case=False just f

def tokenize_function(examples):
    return tokenizer(examples["Objeto do Contrato"], padding="max_length", truncation=T

tokenized_contratsAPFit = dataset.map(tokenize_function, batched=True) #map for apply t

tokenized_contratsAPFit = tokenized_contratsAPFit.remove_columns ("Objeto do Contrato")
tokenized_contratsAPFit = tokenized_contratsAPFit.with_format ("torch")
```

```
dataset  DatasetDict({
    train: Dataset({
```

## Creating the Neural Network for Fine Tuning

### Neural Network

- We will be creating a neural network with the `distilbert-base-multilingual-cased`.
- Defining training_args and compute metrics

```
Downloading (…)okenizer_config.json: 100%                                    29.0/29.0 [00
```

```
# Train with AutoModelForSequenceClassification

# labels = 8
model = AutoModelForSequenceClassification.from_pretrained(modelnameused, num_labels=8)
#model = AutoModelForPreTraining.from_pretrained('adalbertojunior/distilbert-portuguese


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,
)

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")

#Call compute on metric to calculate the accuracy of your predictions. Before passing y
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)


```

```
Downloading model.safetensors: 100%                    542M/542M [00:01
Some weights of DistilBertForSequenceClassification were not initialized from the mo
You should probably TRAIN this model on a down-stream task to be able to use it for
```

## 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.

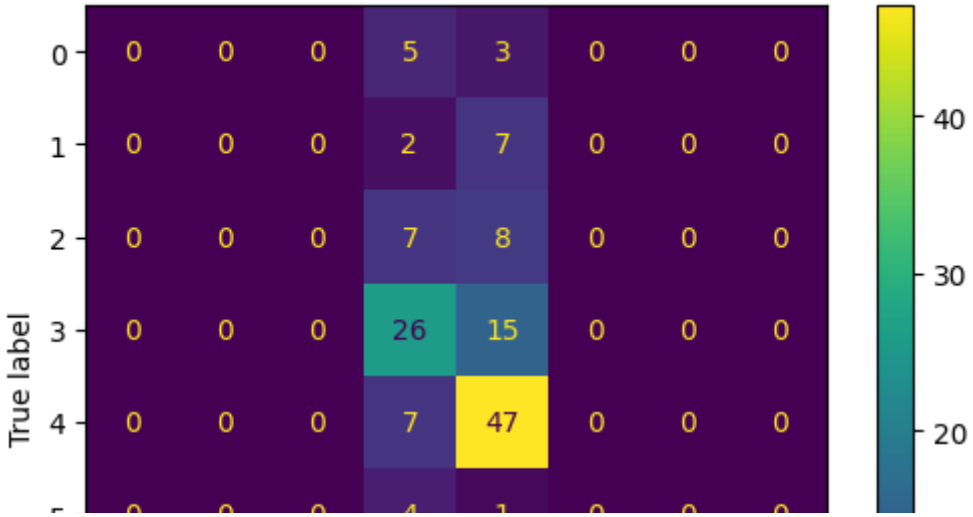
See Confusion Matrix of last prediction iteration

```
#Create a Trainer object with your distilbert model, training arguments, training and t
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=tokenized_contratsAPFit["train"], #tokenized_contratsAPFit_train_data
    eval_dataset=tokenized_contratsAPFit["test"], #tokenized_contratsAPFit_test_dataset
    compute_metrics=compute_metrics,
)

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

```

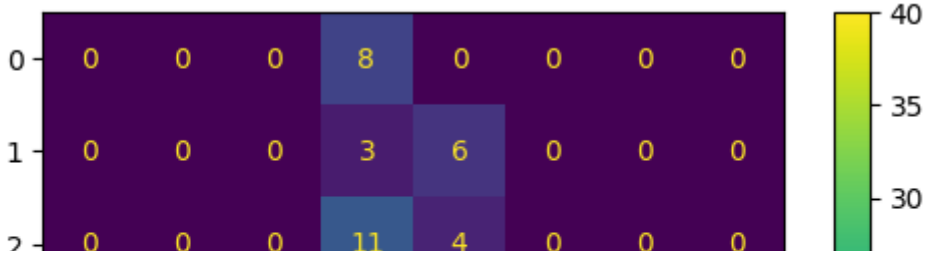
[114/114 01:39, Epoch 3/3]

| Epoch | Training Loss | Validation Loss | Accuracy |
|---|---|---|---|
| 1 | No log | 1.711388 | 0.483444 |
| 2 | No log | 1.551106 | 0.496689 |
| 3 | No log | 1.459548 | 0.509934 |

```
predictions  [4 4 4 4 4 4 4 3 4 4 3 3 3 4 4 4 4 3 4 4 3 3 4 4 4 4 4 3 3 4 4 4 4 3 4
 3 3 4 4 4 4 3 4 4 3 3 4 4 3 4 4 4 4 4 3 3 4 4 3 4 4 4 4 3 3 4 3 4 4 4 3 3 4 4
 3 4 4 3 3 4 4 3 3 3 3 3 4 4 4 3 4 3 4 4 3 4 4 3 3 3 4 4 4 4 4 3 3 4 4 4 4
 4 4 4 4 4 4 4 4 3 4 4 4 4 3 4 3 3 3 4 4 4 4 4 3 4 3 3 3 3 3 3 3 4 3 3 3 4
 4 4 4]
labels       [3 4 4 4 1 3 4 3 3 4 1 5 4 2 4 4 7 3 6 4 0 4 6 1 4 4 4 3 2 4 4 4 4 0 3
 7 3 4 0 4 3 7 4 7 2 3 4 4 3 6 1 7 7 4 4 3 4 3 3 3 6 4 2 3 3 2 4 4 5 4 3 1
 3 4 0 3 7 2 1 3 3 1 3 3 6 2 3 3 4 4 4 2 3 7 1 3 4 5 2 3 3 4 5 5 2 4 1 4 4
 4 2 6 2 4 2 4 4 3 3 4 4 4 7 6 0 3 3 4 4 4 4 0 0 4 6 0 3 4 3 3 3 6 2 3 2 4
 6 3 4]
Confusion Matrix: Focus on  classification Challenges
predictions  [4 4 3 4 4 3 4 3 3 3 3 3 3 3 4 4 3 3 4 4 3 3 4 4 4 4 4 3 3 4 4 4 4 3 4
 3 3 4 3 4 4 3 4 4 3 3 3 4 3 4 4 3 4 4 3 3 4 3 3 4 4 4 3 3 3 3 4 4 3 3 3 4
 3 4 3 3 3 4 4 3 3 3 3 3 4 3 3 3 4 3 4 3 3 4 3 3 3 3 4 3 3 4 4 3 3 4 4 4 4
 3 4 4 3 4 4 3 3 3 3 4 4 4 3 4 3 3 3 4 3 4 4 3 3 4 4 3 3 3 3 3 3 4 3 3 3 4
 4 4 4]
labels       [3 4 4 4 1 3 4 3 3 4 1 5 4 2 4 4 7 3 6 4 0 4 6 1 4 4 4 3 2 4 4 4 4 0 3
 7 3 4 0 4 3 7 4 7 2 3 4 4 3 6 1 7 7 4 4 3 4 3 3 3 6 4 2 3 3 2 4 4 5 4 3 1
 3 4 0 3 7 2 1 3 3 1 3 3 6 2 3 3 4 4 4 2 3 7 1 3 4 5 2 3 3 4 5 5 2 4 1 4 4
 4 2 6 2 4 2 4 4 3 3 4 4 4 7 6 0 3 3 4 4 4 4 0 0 4 6 0 3 4 3 3 3 6 2 3 2 4
 6 3 4]
Confusion Matrix: Focus on  classification Challenges
predictions  [4 4 3 4 4 3 4 3 3 4 3 3 3 3 4 4 3 3 4 4 3 3 4 4 4 4 4 3 3 4 4 4 4 3 4
 3 3 4 2 4 4 3 4 4 3 3 3 4 3 4 4 3 4 4 3 3 4 3 3 4 4 4 3 3 4 3 4 4 3 4 4 4
 3 4 3 3 3 4 4 3 3 3 3 3 4 3 3 3 4 4 4 3 3 4 3 3 3 3 4 3 4 4 4 3 3 4 4 4 4
 4 4 4 3 4 4 4 4 3 3 3 4 4 4 3 4 3 3 3 4 3 4 4 3 3 4 4 3 3 3 3 3 3 4 3 3 3 4
 4 4 4]
labels       [3 4 4 4 1 3 4 3 3 4 1 5 4 2 4 4 7 3 6 4 0 4 6 1 4 4 4 3 2 4 4 4 4 0 3
 7 3 4 0 4 3 7 4 7 2 3 4 4 3 6 1 7 7 4 4 3 4 3 3 3 6 4 2 3 3 2 4 4 5 4 3 1
 3 4 0 3 7 2 1 3 3 1 3 3 6 2 3 3 4 4 4 2 3 7 1 3 4 5 2 3 3 4 5 5 2 4 1 4 4
 4 2 6 2 4 2 4 4 3 3 4 4 4 7 6 0 3 3 4 4 4 4 0 0 4 6 0 3 4 3 3 3 6 2 3 2 4
 6 3 4]
Confusion Matrix: Focus on  classification Challenges
TrainOutput(global_step=114, training_loss=1.7045571845874452, metrics={'train_runti
'train_samples_per_second': 17.236, 'train_steps_per_second': 1.092, 'total_flos': 2
'train_loss': 1.7045571845874452, 'epoch': 3.0})
```




5
6
7
0 0 0 4 1 0 0 0
0 0 0 1 9 0 0 0
0 0 0 4 5 0 0 0
0 1 2 3 4 5 6 7
Predicted label
10
0


0
1
2
0 0 0 8 0 0 0 0
0 0 0 3 6 0 0 0
0 0 0 11 4 0 0 0
40
35
30