In [6]:

```
import torch

# If there's a GPU available...
if torch.cuda.is_available():

    # Tell PyTorch to use the GPU.
    device = torch.device("cuda")

    print('There are %d GPU(s) available.' % torch.cuda.device_count())

    print('We will use the GPU:', torch.cuda.get_device_name(0))
    !nvidia-smi

# If not...
else:
    print('No GPU available, using the CPU instead.')
    device = torch.device("cpu")
```

```
There are 1 GPU(s) available.
We will use the GPU: Tesla T4
Fri Jun 10 15:32:42 2022
+-----------------------------------------------------------------------------
----------+
| NVIDIA-SMI 460.32.03    Driver Version: 460.32.03    CUDA Version:
11.2     |
|-------------------------------+----------------------+------------
----------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Un
corr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  C
ompute M. |
|                               |                      |
MIG M. |
|===============================+======================+============
==========|
|   0  Tesla T4            Off  | 00000000:00:04.0 Off |
0 |
| N/A   36C    P8     9W /  70W |      3MiB / 15109MiB |      0%
Default |
|                               |                      |
N/A |
+-------------------------------+----------------------+------------
----------+

+-----------------------------------------------------------------------------
----------+
| Processes:
|
|  GPU   GI   CI        PID   Type   Process name                  G
PU Memory |
|        ID   ID                                                   U
sage      |
|=============================================================================
==========|
|  No running processes found
|
+-----------------------------------------------------------------------------
----------+
```

Importing the libraries needed

In [7]:

```
!pip install pyarabic
!pip install emoji
!pip install pystemmer
!pip install optuna==2.3.0
!pip install transformers==4.2.1
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.d
ev/colab-wheels/public/simple/
Requirement already satisfied: pyarabic in /usr/local/lib/python3.7/
dist-packages (0.6.14)
Requirement already satisfied: six>=1.14.0 in /usr/local/lib/python
3.7/dist-packages (from pyarabic) (1.15.0)
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.d
ev/colab-wheels/public/simple/
Requirement already satisfied: emoji in /usr/local/lib/python3.7/dis
t-packages (1.7.0)
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.d
ev/colab-wheels/public/simple/
Requirement already satisfied: pystemmer in /usr/local/lib/python3.
7/dist-packages (2.0.1)
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.d
ev/colab-wheels/public/simple/
Requirement already satisfied: optuna==2.3.0 in /usr/local/lib/pytho
n3.7/dist-packages (2.3.0)
Requirement already satisfied: colorlog in /usr/local/lib/python3.7/
dist-packages (from optuna==2.3.0) (6.6.0)
Requirement already satisfied: sqlalchemy>=1.1.0 in /usr/local/lib/p
ython3.7/dist-packages (from optuna==2.3.0) (1.4.36)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/pyt
hon3.7/dist-packages (from optuna==2.3.0) (21.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.7/dist
-packages (from optuna==2.3.0) (4.64.0)
Requirement already satisfied: cmaes>=0.6.0 in /usr/local/lib/python
3.7/dist-packages (from optuna==2.3.0) (0.8.2)
Requirement already satisfied: cliff in /usr/local/lib/python3.7/dis
t-packages (from optuna==2.3.0) (3.10.1)
Requirement already satisfied: alembic in /usr/local/lib/python3.7/d
ist-packages (from optuna==2.3.0) (1.8.0)
Requirement already satisfied: joblib in /usr/local/lib/python3.7/di
st-packages (from optuna==2.3.0) (1.1.0)
Requirement already satisfied: scipy!=1.4.0 in /usr/local/lib/python
3.7/dist-packages (from optuna==2.3.0) (1.4.1)
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dis
t-packages (from optuna==2.3.0) (1.21.6)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /usr/loca
l/lib/python3.7/dist-packages (from packaging>=20.0->optuna==2.3.0)
(3.0.9)
Requirement already satisfied: greenlet!=0.4.17 in /usr/local/lib/py
thon3.7/dist-packages (from sqlalchemy>=1.1.0->optuna==2.3.0) (1.1.
2)
Requirement already satisfied: importlib-metadata in /usr/local/lib/
python3.7/dist-packages (from sqlalchemy>=1.1.0->optuna==2.3.0) (4.1
1.4)
Requirement already satisfied: Mako in /usr/local/lib/python3.7/dist
-packages (from alembic->optuna==2.3.0) (1.2.0)
Requirement already satisfied: importlib-resources in /usr/local/li
b/python3.7/dist-packages (from alembic->optuna==2.3.0) (5.7.1)
Requirement already satisfied: autopage>=0.4.0 in /usr/local/lib/pyt
hon3.7/dist-packages (from cliff->optuna==2.3.0) (0.5.1)
Requirement already satisfied: pbr!=2.1.0,>=2.0.0 in /usr/local/lib/
python3.7/dist-packages (from cliff->optuna==2.3.0) (5.9.0)
Requirement already satisfied: stevedore>=2.0.1 in /usr/local/lib/py
thon3.7/dist-packages (from cliff->optuna==2.3.0) (3.5.0)
Requirement already satisfied: PrettyTable>=0.7.2 in /usr/local/lib/
python3.7/dist-packages (from cliff->optuna==2.3.0) (3.3.0)
Requirement already satisfied: cmd2>=1.0.0 in /usr/local/lib/python
3.7/dist-packages (from cliff->optuna==2.3.0) (2.4.1)
```

```
Requirement already satisfied: PyYAML>=3.12 in /usr/local/lib/python
3.7/dist-packages (from cliff->optuna==2.3.0) (3.13)
Requirement already satisfied: typing-extensions in /usr/local/lib/p
ython3.7/dist-packages (from cmd2>=1.0.0->cliff->optuna==2.3.0) (4.
2.0)
Requirement already satisfied: attrs>=16.3.0 in /usr/local/lib/pytho
n3.7/dist-packages (from cmd2>=1.0.0->cliff->optuna==2.3.0) (21.4.0)
Requirement already satisfied: pyperclip>=1.6 in /usr/local/lib/pyth
on3.7/dist-packages (from cmd2>=1.0.0->cliff->optuna==2.3.0) (1.8.2)
Requirement already satisfied: wcwidth>=0.1.7 in /usr/local/lib/pyth
on3.7/dist-packages (from cmd2>=1.0.0->cliff->optuna==2.3.0) (0.2.5)
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.
7/dist-packages (from importlib-metadata->sqlalchemy>=1.1.0->optuna=
=2.3.0) (3.8.0)
Requirement already satisfied: MarkupSafe>=0.9.2 in /usr/local/lib/p
ython3.7/dist-packages (from Mako->alembic->optuna==2.3.0) (2.0.1)
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.d
ev/colab-wheels/public/simple/
Requirement already satisfied: transformers==4.2.1 in /usr/local/li
b/python3.7/dist-packages (4.2.1)
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dis
t-packages (from transformers==4.2.1) (1.21.6)
Requirement already satisfied: filelock in /usr/local/lib/python3.7/
dist-packages (from transformers==4.2.1) (3.7.0)
Requirement already satisfied: sacremoses in /usr/local/lib/python3.
7/dist-packages (from transformers==4.2.1) (0.0.53)
Requirement already satisfied: importlib-metadata in /usr/local/lib/
python3.7/dist-packages (from transformers==4.2.1) (4.11.4)
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.
7/dist-packages (from transformers==4.2.1) (4.64.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.
7/dist-packages (from transformers==4.2.1) (21.3)
Requirement already satisfied: tokenizers==0.9.4 in /usr/local/lib/p
ython3.7/dist-packages (from transformers==4.2.1) (0.9.4)
Requirement already satisfied: requests in /usr/local/lib/python3.7/
dist-packages (from transformers==4.2.1) (2.23.0)
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/p
ython3.7/dist-packages (from transformers==4.2.1) (2019.12.20)
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.
7/dist-packages (from importlib-metadata->transformers==4.2.1) (3.8.
0)
Requirement already satisfied: typing-extensions>=3.6.4 in /usr/loca
l/lib/python3.7/dist-packages (from importlib-metadata->transformers
==4.2.1) (4.2.0)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /usr/loca
l/lib/python3.7/dist-packages (from packaging->transformers==4.2.1)
(3.0.9)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.2
1.1 in /usr/local/lib/python3.7/dist-packages (from requests->transf
ormers==4.2.1) (1.24.3)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/
python3.7/dist-packages (from requests->transformers==4.2.1) (2022.
5.18.1)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python
3.7/dist-packages (from requests->transformers==4.2.1) (2.10)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/p
ython3.7/dist-packages (from requests->transformers==4.2.1) (3.0.4)
Requirement already satisfied: click in /usr/local/lib/python3.7/dis
t-packages (from sacremoses->transformers==4.2.1) (7.1.2)
Requirement already satisfied: joblib in /usr/local/lib/python3.7/di
st-packages (from sacremoses->transformers==4.2.1) (1.1.0)
```

```
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-
packages (from sacremoses->transformers==4.2.1) (1.15.0)
```

In [8]:

```
import numpy as np
import pandas as pd
import pyarabic.araby as ar

import re , emoji, Stemmer, functools, operator, string
import torch , optuna, gc, random, os

import matplotlib.pyplot as plt
import seaborn as sns

from tqdm import tqdm_notebook as tqdm
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, accuracy_score, f1_score, con
fusion_matrix, precision_score , recall_score
from transformers import AutoConfig, AutoModelForSequenceClassification, AutoTok
enizer
from transformers.data.processors import SingleSentenceClassificationProcessor
from transformers import Trainer , TrainingArguments
from transformers.trainer_utils import EvaluationStrategy
from transformers.data.processors.utils import InputFeatures
from torch.utils.data import Dataset
from torch.utils.data import DataLoader
from sklearn.utils import resample
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, classification_report, accuracy_sc
ore

import gensim
from gensim.models import KeyedVectors

from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences

import tensorflow as tf
from keras.models import Sequential
from tensorflow.keras.layers import SpatialDropout1D, Conv1D, Bidirectional, LST
M, Dense, Input, Dropout, GlobalMaxPooling1D
from keras.layers.embeddings import Embedding
from tensorflow.keras.callbacks import ModelCheckpoint, ReduceLROnPlateau, Early
Stopping
from tensorflow.keras.optimizers import Adam

import itertools
from numpy import loadtxt
from keras.models import load_model

import logging

logging.basicConfig(level=logging.WARNING)
logger = logging.getLogger(__name__)
```

the function of the preprocessing

In [9]:

```
st =  Stemmer.Stemmer('arabic')
def data_cleaning (x):
    x = re.sub('@[^\s]+', ' ', x)
    x = re.sub('((www\.[^\s]+)|(https?://[^\s]+))',' ',x)

    emoji_pattern = re.compile("["
                               u"\U0001F600-\U0001F64F"  # emoticons
                               u"\U0001F300-\U0001F5FF"  # symbols & pictographs
                               u"\U0001F680-\U0001F6FF"  # transport & map symbo
ls
                               u"\U0001F1E0-\U0001F1FF"  # flags (iOS)
                               u"\U00002500-\U00002BEF"  # chinese char
                               u"\U00002702-\U000027B0"
                               u"\U00002702-\U000027B0"
                               u"\U000024C2-\U0001F251"
                               u"\U0001f926-\U0001f937"
                               u"\U00010000-\U0010ffff"
                               u"\u2640-\u2642"
                               u"\u2600-\u2B55"
                               u"\u200d"
                               u"\u23cf"
                               u"\u23e9"
                               u"\u231a"
                               u"\ufe0f"  # dingbats
                               u"\u3030""]+", flags=re.UNICODE)
    emoji_pattern.sub(r'', x)

    ar_punctuations = '''`÷×؛<>_()*&^%][ـ،/:"؟.,'{}~¦+|!”…“–ـ'''
    en_punctuations = string.punctuation
    punctuations = ar_punctuations + en_punctuations
    x = x.translate(str.maketrans('', '', punctuations))

    arabic_diacritics = re.compile("""  ّ    | # Tashdid
                                ً    | # Fatha
                                ٌ    | # Tanwin Fath
                                ٍ    | # Damma
                                ْ    | # Tanwin Damm
                                ٓ    | # Kasra
                                ٔ    | # Tanwin Kasr
                                ٕ    | # Sukun
                                ـ     # Tatwil/Kashida
                            """, re.VERBOSE)
    x = re.sub(arabic_diacritics, '', str(x))

#     x = re.sub("[إأآا]" ,"ا", x)
#     x = re.sub("ى" ,"ي", x)
#     x = re.sub("ة" ,"ه", x)
#     x = re.sub("گ" ,"ك", x)
#     x = re.sub(r'(.)\1+', r'\1', x)

    return x
```

Connecting to google drive

In [10]:

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

```
Mounted at /content/gdrive
```

Uploading the dataset

In [11]:

```
Text_Col_Train = "review"
Sentiment_Col_Train = "sentiment"
Train_Data_File = "/content/gdrive/MyDrive/thesis/LABR.xlsx"

train_data = pd.DataFrame()

train_data = pd.read_excel(Train_Data_File)

train_data.head(3)
```

Out[11]:

	rating	Unnamed: 1	Unnamed: 2	Unnamed: 3	review
0	4.0	338670838.0	7878381.0	13431841.0	عزازيل الذي صنعناه ،الكامن في أنفسنا يذكرني يو...
1	4.0	39428407.0	1775679.0	3554772.0	من أمتع ما قرأت من روايات بلا شك. وحول الشك تد...
2	4.0	32159373.0	1304410.0	3554772.0	رواية تتخذ من التاريخ ،جوًا لها اختار المؤلف ف...

In [12]:

```
print(train_data.rating.value_counts())
```

```
5.0    23705
4.0    19019
3.0    12168
2.0     5265
1.0     2909
Name: rating, dtype: int64
```

printing the fiels with missed values

In [13]:

```
train_data.isnull().sum()
```

Out[13]:

```
rating        0
Unnamed: 1    0
Unnamed: 2    0
Unnamed: 3    0
review        0
dtype: int64
```

printing the number of the duplicated rows

In [14]:

```
print("On a  {} doublons dans Data.".format(train_data.duplicated().sum()))
```

```
On a  2464 doublons dans Data.
```

In [15]:

```
train_data.drop_duplicates(inplace = True)
```

In [16]:

```
print("On a  {} doublons dans Data.".format(train_data.duplicated().sum()))
```

```
On a  0 doublons dans Data.
```

checking the types of the fiels in the data

In [17]:

```
train_data.dtypes
```

Out[17]:

```
rating         float64
Unnamed: 1     float64
Unnamed: 2     float64
Unnamed: 3     float64
review          object
dtype: object
```

function for printing the pie

In [18]:

```
def pie(data,col):
    labels = data[col].value_counts().keys().tolist()
    n = len(labels)
    if n==2:
        colors = ['#66b3ff', '#fb3999']
    elif n==3:
        colors = ['#66b3ff', '#fb3999', '#ffcc99']
    elif n==4:
        colors = ['#66b3ff', '#fb3999', '#ffcc99',"#66f3ff"]
    elif n==5:
        colors = ['#66b3ff', '#fb3999', '#ffcc99',"#66f3ff",'#adcc99']
    elif n==6:
        colors = ['#66b3ff', '#fb3999', '#ffcc99',"#66f3ff",'#adcc99',"#db7f23"]

    fig1, f1 = plt.subplots()
    f1.pie(data[col].value_counts(), labels=labels, colors = colors, autopct='%
1.1f%%',shadow=False, startangle=60)
    f1.axis('equal')
    plt.tight_layout()
    plt.show()

def histo(data,col):
    plt.figure(figsize = (10, 8))
    sns.histplot(data=data, x=col, hue = data[col], fill=True)
```

Counting the % of each classe

In [19]:

```
train_data.rating.value_counts(normalize = True)
```

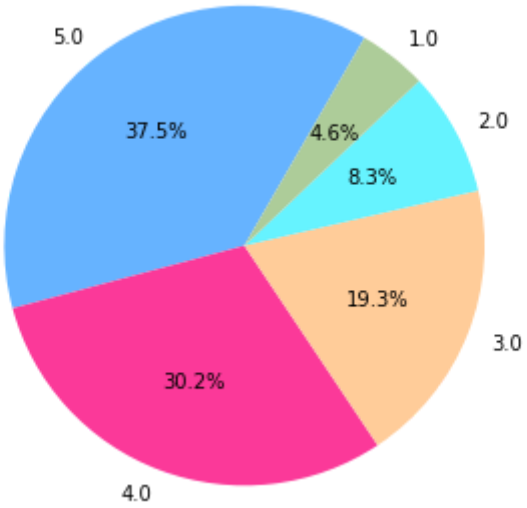
Out[19]:

```
5.0    0.375433
4.0    0.301508
3.0    0.193310
2.0    0.083479
1.0    0.046269
Name: rating, dtype: float64
```

Printing the distribution of the classes

In [20]:

```
pie(train_data, "rating")
```



Repartitionning the data to 2 classes

In [21]:

```
positive_reviews = train_data[train_data["rating"] > 3]
positive_reviews["sentiment"] = "Positive"

negative_reviews = train_data[train_data["rating"] < 3]
negative_reviews["sentiment"] = "Negative"

train_data = pd.concat([positive_reviews, negative_reviews], ignore_index = True
)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: Sett
ingWithCopyWarning:
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/pand
as-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-co
py

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:5: Sett
ingWithCopyWarning:
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/pand
as-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-co
py
  """
```

preprocessing the reviews and printing the time spent & Deleting unused fields

In [22]:

```
# Cleaning Training Data
train_data[Text_Col_Train] = train_data[Text_Col_Train].apply(lambda x:   data_c
leaning(str(x)))

# Removing un-needed feilds
train_data.drop(['rating', 'Unnamed: 1', 'Unnamed: 2', 'Unnamed: 3'], axis = 1,
inplace = True)
train_data.head(3)
```

Out[22]:

	review	sentiment
0	عزازيل الذي صنعناه الكامن في أنفسنا يذكرني يوس...	Positive
1	من أمتع ما قرأت من روايات بلا شك وحول الشك تدن...	Positive
2	رواية تتخذ من التاريخ جوا لها اختار المؤلف فتر...	Positive

Spliting Data (Train , Evaluation)

In [23]:

```
# First setting the max_len , will be useful later for BERT Model
Extra_Len = 6 # an extra padding in length , found to be useful for increasing F
-score
Max_Len = 512
print(Max_Len)

#Spliting the Training data
Test_Size = 0.20

Rand_Seed = 42

train_set, evaluation_set = train_test_split( train_data, test_size= Test_Size,
random_state= Rand_Seed)

y=pd.get_dummies(train_data.sentiment)

train_set, X_test, y_train, y_test = train_test_split(train_data, y, test_size =
0.20, random_state = 42)

print("Train set: ")
print(train_set[Sentiment_Col_Train].value_counts())
print("----------------------------")
print ("Evaluation set: ")
print (evaluation_set[Sentiment_Col_Train].value_counts())
```

```
512
Train set:
Positive    32859
Negative     6250
Name: sentiment, dtype: int64
----------------------------
Evaluation set:
Positive    8165
Negative    1613
Name: sentiment, dtype: int64
```

Preparing BERTModel Classes

In [24]:

```
Model_Used = "UBC-NLP/MARBERT"
Task_Name = "classification"

class Dataset:
    def __init__(
        self,
        name,
        train,
        test,
        label_list,
    ):
        self.name = name
        self.train = train
        self.test = test
        self.label_list = label_list

class BERTModelDataset(Dataset):
    def __init__(self, text, target, model_name, max_len, label_map):
      super(BERTModelDataset).__init__()
      self.text = text
      self.target = target
      self.tokenizer_name = model_name
      self.tokenizer = AutoTokenizer.from_pretrained(model_name)
      self.max_len = max_len
      self.label_map = label_map

    def __len__(self):
      return len(self.text)

    def __getitem__(self,item):
      text = str(self.text[item])
      text = " ".join(text.split())

      encoded_review = self.tokenizer.encode_plus(
      text,
      max_length= self.max_len,
      add_special_tokens= True,
      return_token_type_ids=False,
      pad_to_max_length=True,
      truncation='longest_first',
      return_attention_mask=True,
      return_tensors='pt'
    )
      input_ids = encoded_review['input_ids'].to(device)
      attention_mask = encoded_review['attention_mask'].to(device)

      return InputFeatures(input_ids=input_ids.flatten(), attention_mask=attenti
on_mask.flatten(), label=self.label_map[self.target[item]])
```

Defining Needed Methods for training and evaluation

In [25]:

```
def model_init():
  return AutoModelForSequenceClassification.from_pretrained(Model_Used, return_d
ict=True, num_labels=len(label_map))

def compute_metrics(p): #p should be of type EvalPrediction
  preds = np.argmax(p.predictions, axis=1)
  assert len(preds) == len(p.label_ids)
  print(classification_report(p.label_ids,preds))
  #print(confusion_matrix(p.label_ids,preds))

  macro_f1_pos_neg = f1_score(p.label_ids,preds,average='macro',labels=[1,2])
  macro_f1 = f1_score(p.label_ids,preds,average='macro')
  macro_precision = precision_score(p.label_ids,preds,average='macro')
  macro_recall = recall_score(p.label_ids,preds,average='macro')
  acc = accuracy_score(p.label_ids,preds)
  return {
      'macro_f1' : macro_f1,
      'macro_f1_pos_neg' : macro_f1_pos_neg,
      'macro_precision': macro_precision,
      'macro_recall': macro_recall,
      'accuracy': acc
  }

def set_seed(seed):
    torch.manual_seed(seed)
    torch.cuda.manual_seed_all(seed)
    torch.backends.cudnn.deterministic = True
    torch.backends.cudnn.benchmark = False
    np.random.seed(seed)
    random.seed(seed)
    os.environ['PYTHONHASHSEED'] = str(seed)
```

Build Train and Evaluation Datasets

In [26]:

```
label_list = list(train_set[Sentiment_Col_Train].unique())

print(label_list)
print(train_set[Sentiment_Col_Train].value_counts())

data_set = Dataset( "LABR", train_set, evaluation_set, label_list )

label_map = { v:index for index, v in enumerate(label_list) }
print(label_map)

train_dataset = BERTModelDataset(train_set[Text_Col_Train].to_list(),
                                 train_set[Sentiment_Col_Train].to_list(),Model_
Used,Max_Len,label_map)

evaluation_dataset = BERTModelDataset(evaluation_set[Text_Col_Train].to_list(),
                                      evaluation_set[Sentiment_Col_Train].to_lis
t(),Model_Used,Max_Len,label_map)
```

```
['Positive', 'Negative']
Positive    32859
Negative     6250
Name: sentiment, dtype: int64
{'Positive': 0, 'Negative': 1}
```

Define Training Arguments

In [27]:

```
#define training arguments
training_args = TrainingArguments("./train")
training_args.lr_scheduler_type = 'cosine'
training_args.evaluate_during_training = True
training_args.adam_epsilon =1e-8
training_args.learning_rate = 1.78255000000000001e-05 # use this with org data
training_args.fp16 = True
training_args.per_device_train_batch_size = 16
training_args.per_device_eval_batch_size = 128
training_args.gradient_accumulation_steps = 2
training_args.num_train_epochs= 2
training_args.warmup_steps = 0
training_args.evaluation_strategy = EvaluationStrategy.EPOCH
training_args.logging_steps = 200
training_args.save_steps = 100000
training_args.seed = 42
training_args.disable_tqdm = False
```

Build The Trainer

In [28]:

```
training_args.dataloader_pin_memory = False
gc.collect()
torch.cuda.empty_cache()
set_seed(Rand_Seed)

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

print(training_args.seed)
```

```
Some weights of the model checkpoint at UBC-NLP/MARBERT were not use
d when initializing BertForSequenceClassification: ['cls.prediction
s.bias', 'cls.predictions.transform.dense.weight', 'cls.predictions.
transform.dense.bias', 'cls.predictions.transform.LayerNorm.weight',
'cls.predictions.transform.LayerNorm.bias', 'cls.predictions.decode
r.weight', 'cls.seq_relationship.weight', 'cls.seq_relationship.bia
s']
- This IS expected if you are initializing BertForSequenceClassifica
tion from the checkpoint of a model trained on another task or with
another architecture (e.g. initializing a BertForSequenceClassificat
ion model from a BertForPreTraining model).
- This IS NOT expected if you are initializing BertForSequenceClassi
fication from the checkpoint of a model that you expect to be exactl
y identical (initializing a BertForSequenceClassification model from
a BertForSequenceClassification model).
Some weights of BertForSequenceClassification were not initialized f
rom the model checkpoint at UBC-NLP/MARBERT and are newly initialize
d: ['classifier.weight', 'classifier.bias']
You should probably TRAIN this model on a down-stream task to be abl
e to use it for predictions and inference.
```

```
42
```

Train

In [29]:

```
all_results = []

print(Max_Len)
print(training_args.learning_rate)
print(training_args.adam_epsilon)
print(training_args.warmup_steps)
trainer.train()

results = trainer.evaluate()
all_results.append(results)
print(results)
```

```
512
1.78255e-05
1e-08
0


/usr/local/lib/python3.7/dist-packages/transformers/tokenization_uti
ls_base.py:2143: FutureWarning: The `pad_to_max_length` argument is
deprecated and will be removed in a future version, use `padding=Tru
e` or `padding='longest'` to pad to the longest sequence in the batc
h, or use `padding='max_length'` to pad to a max length. In this cas
e, you can give a specific length with `max_length` (e.g. `max_lengt
h=45`) or leave max_length to None to pad to the maximal input size
of the model (e.g. 512 for Bert).
  FutureWarning,
```

[2444/2444 50:40, Epoch 1/2]

Epoch	Training Loss	Validation Loss	Macro F1	Macro F1 Pos Neg	Macro Precision	Macro Recall	Accuracy	Runtime
0	0.207700	0.191087	0.866039	0.386425	0.894353	0.843435	0.930865	124.536700
1	0.128400	0.230411	0.873862	0.393296	0.897671	0.854162	0.934240	124.477900

```
              precision    recall  f1-score   support

           0       0.94      0.97      0.96      8165
           1       0.84      0.71      0.77      1613

    accuracy                           0.93      9778
   macro avg       0.89      0.84      0.87      9778
weighted avg       0.93      0.93      0.93      9778


/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classificati
on.py:1580: UndefinedMetricWarning: F-score is ill-defined and being
set to 0.0 in labels with no true nor predicted samples. Use `zero_d
ivision` parameter to control this behavior.
  _warn_prf(average, "true nor predicted", "F-score is", len(true_su
m))
/usr/local/lib/python3.7/dist-packages/transformers/tokenization_uti
ls_base.py:2143: FutureWarning: The `pad_to_max_length` argument is
deprecated and will be removed in a future version, use `padding=Tru
e` or `padding='longest'` to pad to the longest sequence in the batc
h, or use `padding='max_length'` to pad to a max length. In this cas
e, you can give a specific length with `max_length` (e.g. `max_lengt
h=45`) or leave max_length to None to pad to the maximal input size
of the model (e.g. 512 for Bert).
  FutureWarning,
```

```
              precision    recall  f1-score   support

           0       0.95      0.97      0.96      8165
           1       0.85      0.73      0.79      1613

    accuracy                           0.93      9778
   macro avg       0.90      0.85      0.87      9778
weighted avg       0.93      0.93      0.93      9778


/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classificati
on.py:1580: UndefinedMetricWarning: F-score is ill-defined and being
set to 0.0 in labels with no true nor predicted samples. Use `zero_d
ivision` parameter to control this behavior.
  _warn_prf(average, "true nor predicted", "F-score is", len(true_su
m))
/usr/local/lib/python3.7/dist-packages/transformers/tokenization_uti
ls_base.py:2143: FutureWarning: The `pad_to_max_length` argument is
deprecated and will be removed in a future version, use `padding=Tru
e` or `padding='longest'` to pad to the longest sequence in the batc
h, or use `padding='max_length'` to pad to a max length. In this cas
e, you can give a specific length with `max_length` (e.g. `max_lengt
h=45`) or leave max_length to None to pad to the maximal input size
of the model (e.g. 512 for Bert).
  FutureWarning,
```

[77/77 02:03]

```
              precision    recall  f1-score   support

           0       0.95      0.97      0.96      8165
           1       0.85      0.73      0.79      1613

    accuracy                           0.93      9778
   macro avg       0.90      0.85      0.87      9778
weighted avg       0.93      0.93      0.93      9778

{'eval_loss': 0.23041146993637085, 'eval_macro_f1': 0.87386151677575
04, 'eval_macro_f1_pos_neg': 0.3932957185529372, 'eval_macro_precisi
on': 0.8976711966715547, 'eval_macro_recall': 0.8541620080872305, 'e
val_accuracy': 0.9342401309061158, 'eval_runtime': 124.6487, 'eval_s
amples_per_second': 78.444, 'epoch': 2.0}

/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classificati
on.py:1580: UndefinedMetricWarning: F-score is ill-defined and being
set to 0.0 in labels with no true nor predicted samples. Use `zero_d
ivision` parameter to control this behavior.
  _warn_prf(average, "true nor predicted", "F-score is", len(true_su
m))
```

Results

In [30]:

```
all_results
```

Out[30]:

```
[{'epoch': 2.0,
  'eval_accuracy': 0.9342401309061158,
  'eval_loss': 0.23041146993637085,
  'eval_macro_f1': 0.8738615167757504,
  'eval_macro_f1_pos_neg': 0.3932957185529372,
  'eval_macro_precision': 0.8976711966715547,
  'eval_macro_recall': 0.8541620080872305,
  'eval_runtime': 124.6487,
  'eval_samples_per_second': 78.444}]
```

In [31]:

```
from statistics import mean
mean([x['eval_macro_f1'] for x in all_results])
```

Out[31]:

```
0.8738615167757504
```