In [2]:
```
# To access Google Drive:
from google.colab import drive

drive.mount("/content/gdrive")
```

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

In [3]:
```
REVIEW_PATH = "/content/gdrive/MyDrive/reviews.csv"
```

In [4]:
```
import pandas as pd
import numpy as np
import tensorflow as tf
```

In [5]:
```
BATCH_SIZE=8
```

In [6]:
```
data = pd.read_csv(REVIEW_PATH)
```

In [7]:
```
data.head()
```

Out[7]:

| | reviewId | userName | |
|---|---|---|---|
| **0** | gp:AOqpTOEhZuqSqqWnaKRgv-9ABYdajFUB0WugPGh-SG-... | Eric Tie | lh.googleus |
| **1** | gp:AOqpTOH0WP4IQKBZ2LrdNmFy_YmpPCVrV3diEU9KGm3... | john alpha | lh.googleus |
| **2** | gp:AOqpTOEMCkJB8Iq1p-r9dPwnSYadA5BkPWTf32Z1azu... | Sudhakar .S | lh.googleus |
| **3** | gp:AOqpTOGFrUWuKGycpje8kszj3uwHN6tU_fd4gLVFy9z... | SKGflorida@bellsouth.net DAVID S | lh.googleus |
| **4** | gp:AOqpTOHIs7DW8wmDFzTkHwxuqFkdNQtKHmO6Pt9jhZE... | Louann Stoker | https://play-l |

In [7]:
```

```

In [8]:
```
data = data[["content","score"]].rename(columns={"content":"text","score":"lab
```

In [9]:
```
data.head()
```

Out[9]:

| | text | label |
|---|---|---|
| **0** | I cannot open the app anymore | 1 |
| **1** | I have been begging for a refund from this app... | 1 |
| **2** | Very costly for the premium version (approx In... | 1 |
| **3** | Used to keep me organized, but all the 2020 UP... | 1 |
| **4** | Dan Birthday Oct 28 | 1 |

In [10]:
```
data['label'] = data['label'].replace([1,2,3,4,5],[0,1,2,3,4])
```

In [11]:
```
data['label'].value_counts()
```

Out[11]:
```
label
4    2879
3    2775
0    2506
1    2344
2    1991
Name: count, dtype: int64
```

In [12]:
```
!pip install datasets
```

```
Collecting datasets
  Downloading datasets-2.18.0-py3-none-any.whl (510 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 510.5/510.5 kB 3.9 MB/s eta 0:0
0:00
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-pac
kages (from datasets) (3.13.4)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist-
packages (from datasets) (1.25.2)
Requirement already satisfied: pyarrow>=12.0.0 in /usr/local/lib/python3.10/d
ist-packages (from datasets) (14.0.2)
Requirement already satisfied: pyarrow-hotfix in /usr/local/lib/python3.10/di
st-packages (from datasets) (0.6)
Collecting dill<0.3.9,>=0.3.0 (from datasets)
  Downloading dill-0.3.8-py3-none-any.whl (116 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 116.3/116.3 kB 15.4 MB/s eta 0:
00:00
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packa
ges (from datasets) (2.0.3)
Requirement already satisfied: requests>=2.19.0 in /usr/local/lib/python3.10/
dist-packages (from datasets) (2.31.0)
Requirement already satisfied: tqdm>=4.62.1 in /usr/local/lib/python3.10/dist
-packages (from datasets) (4.66.2)
Collecting xxhash (from datasets)
  Downloading xxhash-3.4.1-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x8
6_64.whl (194 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 194.1/194.1 kB 23.8 MB/s eta 0:
00:00
Collecting multiprocess (from datasets)
  Downloading multiprocess-0.70.16-py310-none-any.whl (134 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 134.8/134.8 kB 17.7 MB/s eta 0:
00:00
Requirement already satisfied: fsspec[http]<=2024.2.0,>=2023.1.0 in /usr/loca
l/lib/python3.10/dist-packages (from datasets) (2023.6.0)
Requirement already satisfied: aiohttp in /usr/local/lib/python3.10/dist-pack
ages (from datasets) (3.9.3)
Requirement already satisfied: huggingface-hub>=0.19.4 in /usr/local/lib/pyth
on3.10/dist-packages (from datasets) (0.20.3)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-pa
ckages (from datasets) (24.0)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-
packages (from datasets) (6.0.1)
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.10/
dist-packages (from aiohttp->datasets) (1.3.1)
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.10/dis
t-packages (from aiohttp->datasets) (23.2.0)
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.1
0/dist-packages (from aiohttp->datasets) (1.4.1)
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.
10/dist-packages (from aiohttp->datasets) (6.0.5)
Requirement already satisfied: yarl<2.0,>=1.0 in /usr/local/lib/python3.10/di
st-packages (from aiohttp->datasets) (1.9.4)
Requirement already satisfied: async-timeout<5.0,>=4.0 in /usr/local/lib/pyth
on3.10/dist-packages (from aiohttp->datasets) (4.0.3)
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/p
ython3.10/dist-packages (from huggingface-hub>=0.19.4->datasets) (4.11.0)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/pyt
hon3.10/dist-packages (from requests>=2.19.0->datasets) (3.3.2)
```

```
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist
-packages (from requests>=2.19.0->datasets) (3.6)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.1
0/dist-packages (from requests>=2.19.0->datasets) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.1
0/dist-packages (from requests>=2.19.0->datasets) (2024.2.2)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/pytho
n3.10/dist-packages (from pandas->datasets) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist
-packages (from pandas->datasets) (2023.4)
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/di
st-packages (from pandas->datasets) (2024.1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-pac
kages (from python-dateutil>=2.8.2->pandas->datasets) (1.16.0)
Installing collected packages: xxhash, dill, multiprocess, datasets
Successfully installed datasets-2.18.0 dill-0.3.8 multiprocess-0.70.16 xxhash
-3.4.1
```

In [12]:

```

```

## Train Test Split

In [13]:

```python
from sklearn.model_selection import train_test_split

train_data, test_data = train_test_split(data, test_size=0.2)
```

In [14]:

```python
from datasets import DatasetDict, Dataset
```

In [15]:

```python
train_dataset = Dataset.from_pandas(train_data)
test_dataset = Dataset.from_pandas(test_data)

# Create a DatasetDict to store both train and test datasets
dataset= DatasetDict({'train': train_dataset, 'test': test_dataset})
```

In [16]:

```python
dataset
```

Out[16]:

```
DatasetDict({
    train: Dataset({
        features: ['text', 'label', '__index_level_0__'],
        num_rows: 9996
    })
    test: Dataset({
        features: ['text', 'label', '__index_level_0__'],
        num_rows: 2499
    })
})
```

## Loading Tokenizer

In [17]:
```
from transformers import (BertTokenizerFast,TFBertTokenizer,BertTokenizer,Robe
                          DataCollatorWithPadding,TFRobertaForSequenceClassifi
                          TFBertModel,create_optimizer)
from transformers import TFAutoModel, AutoTokenizer
```

In [18]:
```
# model_id="nlptown/bert-base-multilingual-uncased-sentiment"
# tokenizer = BertTokenizerFast.from_pretrained(model_id)
tokenizer =  AutoTokenizer.from_pretrained("bert-base-uncased")
```

```
/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_token.py:88: U
serWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings ta
b (https://huggingface.co/settings/tokens), set it as secret in your Google C
olab and restart your session.
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access p
ublic models or datasets.
  warnings.warn(

tokenizer_config.json:   0%|          | 0.00/48.0 [00:00<?, ?B/s]

config.json:   0%|          | 0.00/570 [00:00<?, ?B/s]

vocab.txt:   0%|          | 0.00/232k [00:00<?, ?B/s]

tokenizer.json:   0%|          | 0.00/466k [00:00<?, ?B/s]
```

#### Passing review data into tokenizer

In [19]:
```
output=tokenizer(dataset['train'][:2]['text'],padding=True,truncation=True,max
# # print(output)
```

#### Checking the decoded version to tokenized data

Here the [CLS] shows the start of the sentence and [SEP] shows the end of the sentence [PAD] is added to end the seqence to make it reach the maximum length. in other words it ensure that all of the sequence are of the same length

```
In [20]: tokenizer.decode(output['input_ids'][0])
```

```
Out[20]: '[CLS] it was good until they updated it and took out the today, someday, upc
oming categories out, which is how i organized everything [SEP] [PAD] [PAD]
[PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD]
[PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD]
[PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD]
[PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD]
[PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD]
[PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD]'
```

### Applying to whole dataset

```
In [21]: def preprocess_function(examples):
           return tokenizer(examples["text"],padding=True,truncation=True,max_length=12
```

```
In [22]: tokenized_dataset = dataset.map(preprocess_function, batched=True)
         tokenized_dataset
```

```
Map:   0%|          | 0/9996 [00:00<?, ? examples/s]

Map:   0%|          | 0/2499 [00:00<?, ? examples/s]
```

```
Out[22]: DatasetDict({
    train: Dataset({
        features: ['text', 'label', '__index_level_0__', 'input_ids', 'token_
type_ids', 'attention_mask'],
        num_rows: 9996
    })
    test: Dataset({
        features: ['text', 'label', '__index_level_0__', 'input_ids', 'token_
type_ids', 'attention_mask'],
        num_rows: 2499
    })
})
```

## Creating tensorflow dataset from tokenized dataset

```
In [23]: tf_train_datasets = tokenized_dataset["train"].to_tf_dataset(
             columns=['input_ids', 'token_type_ids', 'attention_mask', 'label'],
             shuffle=True,
             batch_size=BATCH_SIZE,
         )
```

```
In [24]: tf_val_datasets = tokenized_dataset["test"].to_tf_dataset(
             columns=['input_ids', 'token_type_ids', 'attention_mask', 'label'],
             shuffle=True,
             batch_size=BATCH_SIZE,
             #collate_fn=data_collator
         )
```

Separating the Labels from the dataset

```
In [25]: def swap_positions(dataset):
           return {'input_ids':dataset['input_ids'],
                   'token_type_ids':dataset['token_type_ids'],
                   'attention_mask':dataset['attention_mask'],},dataset['label']
```

```
In [26]: tf_train_dataset=tf_train_datasets.map(swap_positions).prefetch(tf.data.AUTOTU
         tf_val_dataset=tf_val_datasets.map(swap_positions).prefetch(tf.data.AUTOTUNE)
```

## Modelling Using TFBertForSequenceClassification

```
In [27]: # from transformers import AutoModelForSequenceClassification
```

In [28]:
```
# model=TFBertForSequenceClassification.from_pretrained("nlptown/bert-base-mul
model = TFAutoModel.from_pretrained("bert-base-uncased")
model.summary()
```

```
model.safetensors:   0%|          | 0.00/440M [00:00<?, ?B/s]

Some weights of the PyTorch model were not used when initializing the TF 2.0
model TFBertModel: ['cls.predictions.bias', 'cls.seq_relationship.weight', 'c
ls.predictions.transform.dense.weight', 'cls.predictions.transform.LayerNorm.
bias', 'cls.seq_relationship.bias', 'cls.predictions.transform.dense.bias',
'cls.predictions.transform.LayerNorm.weight']
- This IS expected if you are initializing TFBertModel from a PyTorch model t
rained on another task or with another architecture (e.g. initializing a TFBe
rtForSequenceClassification model from a BertForPreTraining model).
- This IS NOT expected if you are initializing TFBertModel from a PyTorch mod
el that you expect to be exactly identical (e.g. initializing a TFBertForSequ
enceClassification model from a BertForSequenceClassification model).
All the weights of TFBertModel were initialized from the PyTorch model.
If your task is similar to the task the model of the checkpoint was trained o
n, you can already use TFBertModel for predictions without further training.

Model: "tf_bert_model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 bert (TFBertMainLayer)      multiple                  109482240

=================================================================
Total params: 109482240 (417.64 MB)
Trainable params: 109482240 (417.64 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

In [29]:
```
class BERTForClassification(tf.keras.Model):

    def __init__(self, bert_model, num_classes):
        super().__init__()
        self.bert = bert_model
        # self.drop = tf.keras.layers.Dropout(rate=0.3)
        self.fc = tf.keras.layers.Dense(num_classes, activation='softmax')

    def call(self, inputs):
        x = self.bert(inputs)[1]
        # pooled_output = outputs[1]  # Using [CLS] token output
        # output = self.dropout(pooled_output)
        return self.fc(x)
```

## Training

```
In [30]: # num_epochs = 5
         # batches_per_epoch = len(tokenized_dataset["train"]) // BATCH_SIZE
         # total_train_steps = int(batches_per_epoch * num_epochs)
```

```
In [31]: # optimizer, schedule = create_optimizer(init_lr=2e-5,num_warmup_steps=0, num_
```

```
In [32]: # print(optimizer)
```

```
<tf_keras.src.optimizers.adam.Adam object at 0x7d3a904fb0d0>
```

```
In [33]: classifier = BERTForClassification(model, num_classes=5)

         classifier.compile(
             optimizer=tf.keras.optimizers.Adam(),
             loss=tf.keras.losses.SparseCategoricalCrossentropy(),
             metrics=['accuracy']
         )
```

```
In [ ]: history=classifier.fit(
            tf_train_dataset,
            validation_data=tf_val_dataset,
            epochs=3)
```

```
Epoch 1/3
 232/1250 [====>.........................] - ETA: 3:15:08 - loss: 1.7368 - ac
curacy: 0.2064
```

```
In [ ]: # model.compile(loss= tf.keras.losses.SparseCategoricalCrossentropy(),
        # optimizer=optimizer,
        #     metrics=['accuracy'],)
        #     #run_eagerly=True)

        #     # CategoricalCrossentropy
```

```
In [ ]: # history=model.fit(
        #     tf_train_dataset,
        #     validation_data=tf_val_dataset,
        #     epochs=5)
```

```
In [ ]: import matplotlib.pyplot as plt
        from sklearn.metrics import confusion_matrix, roc_curve
        import seaborn as sns
```

In [ ]:
```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model_loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()
```

In [ ]:
```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])

plt.title('model_accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()
```

In [ ]:
```
classifier.evaluate(tf_val_dataset)
```

In [ ]:
```

```