```
!pip install transformers
```

```
from transformers import BertTokenizer, TFBertForSequenceClassification
from transformers import InputExample, InputFeatures

model = TFBertForSequenceClassification.from_pretrained("bert-base-uncased")
tokenizer = BertTokenizer.from_pretrained("bert-base-uncased")
```

```
Downloading: 100%  570/570 [00:00<00:00, 11.4kB/s]
Downloading: 100%  511M/511M [00:17<00:00, 30.9MB/s]
All model checkpoint layers were used when initializing TFBertForSequenceClassificati

Some layers of TFBertForSequenceClassification were not initialized from the model ch
You should probably TRAIN this model on a down-stream task to be able to use it for p
Downloading: 100%  226k/226k [00:00<00:00, 840kB/s]
Downloading: 100%  28.0/28.0 [00:00<00:00, 634B/s]
Downloading: 100%  455k/455k [00:00<00:00, 866kB/s]
```

```
import tensorflow as tf
import pandas as pd

URL = "https://ai.stanford.edu/~amaas/data/sentiment/aclImdb_v1.tar.gz"

dataset = tf.keras.utils.get_file(fname="aclImdb_v1.tar.gz",
                                  origin=URL,
                                  untar=True,
                                  cache_dir='.',
                                  cache_subdir='')
# operations on files and collections of files.
import os
import shutil
# Create main directory path ("/aclImdb")
main_dir = os.path.join(os.path.dirname(dataset), 'aclImdb')
# Create sub directory path ("/aclImdb/train")
train_dir = os.path.join(main_dir, 'train')
# Remove unsup folder since this is a supervised learning task
remove_dir = os.path.join(train_dir, 'unsup')
shutil.rmtree(remove_dir)
# View the final train folder
print(os.listdir(train_dir))
# We create a training dataset and a validation
# dataset from our "aclImdb/train" directory with a 80/20 split.
train = tf.keras.preprocessing.text_dataset_from_directory(
    'aclImdb/train', batch_size=30000, validation_split=0.2,
    subset='training', seed=123)
test = tf.keras.preprocessing.text_dataset_from_directory(
    'aclImdb/train', batch_size=30000, validation_split=0.2,
    subset='validation', seed=123)
for i in train.take(1):
```

```
    train_feat = i[0].numpy()
    train_lab = i[1].numpy()

train = pd.DataFrame([train_feat, train_lab]).T
train.columns = ['DATA_COLUMN', 'LABEL_COLUMN']
train['DATA_COLUMN'] = train['DATA_COLUMN'].str.decode("utf-8")
train.head()

for j in test.take(1):
    test_feat = j[0].numpy()
    test_lab = j[1].numpy()

test = pd.DataFrame([test_feat, test_lab]).T
test.columns = ['DATA_COLUMN', 'LABEL_COLUMN']
test['DATA_COLUMN'] = test['DATA_COLUMN'].str.decode("utf-8")
test.head()
InputExample(guid=None,
                          text_a = "Hello, world",
                          text_b = None,
                          label = 1)
def convert_data_to_examples(train, test, DATA_COLUMN, LABEL_COLUMN):
    train_InputExamples = train.apply(lambda x: InputExample(guid=None, # Globally unique I




    validation_InputExamples = test.apply(lambda x: InputExample(guid=None, # Globally uniqu




    return train_InputExamples, validation_InputExamples

    train_InputExamples, validation_InputExamples = convert_data_to_examples(train,




def convert_examples_to_tf_dataset(examples, tokenizer, max_length=128):
        features = [] # -> will hold InputFeatures to be converted later

        for e in examples:
            # Documentation is really strong for this method, so please take a look
            input_dict = tokenizer.encode_plus(
                    e.text_a,
                    add_special_tokens=True,
                    max_length=max_length, # truncates if len(s) > max_length
                    return_token_type_ids=True,
                    return_attention_mask=True,
                    pad_to_max_length=True, # pads to the right by default # CHECK TH
                    truncation=True
            )

            input_ids, token_type_ids, attention_mask = (input_dict["input_ids"],
                    input_dict["token_type_ids"], input_dict['attention_mask'])
```

```
            features.append(
                    InputFeatures(
                            input_ids=input_ids,  attention_mask=attention_mask,  token_type_id
                    )
            )

    def  gen():
            for  f  in  features:
                    yield  (
                            {
                                    "input_ids":  f.input_ids,
                                    "attention_mask":  f.attention_mask,
                                    "token_type_ids":  f.token_type_ids,
                            },
                            f.label,
                    )

    return  tf.data.Dataset.from_generator(
            gen,
            ({"input_ids":  tf.int32,  "attention_mask":  tf.int32,  "token_type_ids":  tf.int3
            (
                    {
                            "input_ids":  tf.TensorShape([None]),
                            "attention_mask":  tf.TensorShape([None]),
                            "token_type_ids":  tf.TensorShape([None]),
                    },
                    tf.TensorShape([]),
            ),
    )


DATA_COLUMN  =  'DATA_COLUMN'
LABEL_COLUMN  =  'LABEL_COLUMN'
```

```
Downloading data from https://ai.stanford.edu/~amaas/data/sentiment/aclImdb_v1.tar.gz
84131840/84125825 [==============================] - 2s 0us/step
84140032/84125825 [==============================] - 2s 0us/step
['urls_pos.txt', 'urls_unsup.txt', 'unsupBow.feat', 'neg', 'pos', 'urls_neg.txt', 'labeledBow
Found 25000 files belonging to 2 classes.
Using 20000 files for training.
Found 25000 files belonging to 2 classes.
Using 5000 files for validation.
```

```
train_InputExamples,  validation_InputExamples  =  convert_data_to_examples(train,  test,  DATA_COL

train_data  =  convert_examples_to_tf_dataset(list(train_InputExamples),  tokenizer)
train_data  =  train_data.shuffle(100).batch(32).repeat(2)

validation_data  =  convert_examples_to_tf_dataset(list(validation_InputExamples),  tokenizer)
validation_data  =  validation_data.batch(32)

model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=3e-5,  epsilon=1e-08,  clipnorm=1.0)
```

```
                    loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
                    metrics=[tf.keras.metrics.SparseCategoricalAccuracy('accuracy')])

model.fit(train_data,  epochs=2,  validation_data=validation_data)
```

```
/usr/local/lib/python3.7/dist-packages/transformers/tokenization_utils_base.py:2218: FutureWa
  FutureWarning,
Epoch 1/2
1250/1250 [==============================] - 1993s 2s/step - loss: 0.2709 - accuracy: 0.8838
Epoch 2/2
1250/1250 [==============================] - 1962s 2s/step - loss: 0.0740 - accuracy: 0.9747
<keras.callbacks.History at 0x7f79d97d8610>
```

```
pred_sentences = ["Since our earliest days, Tesla has been built upon a culture of op

                  "We planned very early—with Gigafactory Shanghai as the

                  "Tesla is on a mission to accelerate the world's tran
]


tf_batch = tokenizer(pred_sentences, max_length=128, padding=True, truncation=True, return_te
tf_outputs = model(tf_batch)
tf_predictions = tf.nn.softmax(tf_outputs[0], axis=-1)
labels = ['Negative','Positive']
label = tf.argmax(tf_predictions, axis=1)
label = label.numpy()
for i in range(len(pred_sentences)):
    print(labels[label[i]])

#https://www.tesla.com/ns_videos/2020-tesla-impact-report.pdf
```

```
Positive
Positive
Negative
```

✓ 0 秒 完成時間：上午12:13