Double-click (or enter) to edit

```
!pip install transformers
```

```
Collecting transformers
  Downloading transformers-4.31.0-py3-none-any.whl (7.4 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 7.4/7.4 MB 61.1 MB/s eta 0:00:00
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from transformers) (3.12.2)
Collecting huggingface-hub<1.0,>=0.14.1 (from transformers)
  Downloading huggingface_hub-0.16.4-py3-none-any.whl (268 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 268.8/268.8 kB 35.7 MB/s eta 0:00:00
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist-packages (from transformers) (1.22.4)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from transformers) (23.1)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-packages (from transformers) (6.0.1)
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.10/dist-packages (from transformers) (2022.10.31)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from transformers) (2.27.1)
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_x86_64.whl (7.8 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 7.8/7.8 MB 115.1 MB/s eta 0:00:00
Collecting safetensors>=0.3.1 (from transformers)
  Downloading safetensors-0.3.1-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (1.3 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.3/1.3 MB 82.3 MB/s eta 0:00:00
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.10/dist-packages (from transformers) (4.65.0)
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packages (from huggingface-hub<1.0,>=0.14.1->transformers)
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.10/dist-packages (from huggingface-hub<1.0,>=0.
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests->transformers) (1.26
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests->transformers) (2023.7.
Requirement already satisfied: charset-normalizer~=2.0.0 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) (3.4)
Installing collected packages: tokenizers, safetensors, huggingface-hub, transformers
Successfully installed huggingface-hub-0.16.4 safetensors-0.3.1 tokenizers-0.13.3 transformers-4.31.0
```

```
from transformers import pipeline
```

```
classifier = pipeline("sentiment-analysis")
```

```
No model was supplied, defaulted to distilbert-base-uncased-finetuned-sst-2-english and re
Using a pipeline without specifying a model name and revision in production is not recomme
```

```
Downloading (…)lve/main/config.json:                 629/629 [00:00<00:00,
100%                                                 24.3kB/s]
Downloading model.safetensors:                       268M/268M [00:01<00:00,
100%                                                 253MB/s]
Downloading (…)okenizer_config.json:                 48.0/48.0 [00:00<00:00,
100%                                                 3.58kB/s]
Downloading (…)solve/main/vocab.txt:                 232k/232k [00:00<00:00,
```

```
classifier("I am very happy to learn transformers and BERT today")
```

```
[{'label': 'POSITIVE', 'score': 0.9997857213020325}]
```

```
classifier('I am very excited with Datascience opportunity')
```

```
[{'label': 'POSITIVE', 'score': 0.9996871948242188}]
```

```
classifier("I dont line burger")
```

```
[{'label': 'NEGATIVE', 'score': 0.9924160242080688}]
```

```
results = classifier(["we are very happy to learn hugging face library.",
           "we hope you dont hate it."])

for result in results:
  print(f"label : {result['label']}, with_score: {round(result['score'], 4)}")
```

```
label : POSITIVE, with_score: 0.9998
label : NEGATIVE, with_score: 0.9796
```

```
classifier1 = pipeline("sentiment-analysis", model = 'nlptown/bert-base-multilingual-uncased-sentiment')
```

```
Downloading (…)lve/main/config.json:        953/953 [00:00<00:00,
100%                                         62.2kB/s]
```

nlptown/bert-base-multilingual-uncased-sentiment:

This is a bert-base-multilingual-uncased model finetuned for sentiment analysis on product reviews in six languages: English, Dutch, German, French, Spanish, and Italian. It predicts the sentiment of the review as a number of stars (between 1 and 5).

This model is intended for direct use as a sentiment analysis model for product reviews in any of the six languages above or for further finetuning on related sentiment analysis tasks.

```
classifier1('je suis génial') #
```

```
[{'label': '5 stars', 'score': 0.8312121629714966}]
```

```
from transformers import AutoTokenizer, TFAutoModelForSequenceClassification
```

```
model_name = "nlptown/bert-base-multilingual-uncased-sentiment"
```

This model only exists in py-torch, so we use 'from_pt' flag to import that into tensorflow.

```
model = TFAutoModelForSequenceClassification.from_pretrained(model_name, from_pt=True)
```

```
All PyTorch model weights were used when initializing TFBertForSequenceClassification.

All the weights of TFBertForSequenceClassification were initialized from the PyTorch model.
If your task is similar to the task the model of the checkpoint was trained on, you can already use TFBertForSequenceClassification
```

```
tokenizer = AutoTokenizer.from_pretrained(model_name)
```

```
classifier1 = pipeline('sentiment-analysis', model = model, tokenizer = tokenizer)
```

```
classifier1("I am a good Data Scientist")
```

```
[{'label': '4 stars', 'score': 0.4875314235687256}]
```

Under the hood: Pretrained models

Let's see what happens beneath the hood when using those pipelines. As we saw the model and tokenizer are created using the pre_trained method

```
from transformers import AutoTokenizer, TFAutoModelForSequenceClassification
```

```
model_name = "distilbert-base-uncased-finetuned-sst-2-english"
```

```
tf_model = TFAutoModelForSequenceClassification.from_pretrained(model_name)

tokenizer = AutoTokenizer.from_pretrained(model_name)
```

```
All PyTorch model weights were used when initializing TFDistilBertForSequenceClassification.

All the weights of TFDistilBertForSequenceClassification were initialized from the PyTorch model.
If your task is similar to the task the model of the checkpoint was trained on, you can already use TFDistilBertForSequenceClassifi
```

```
inputs = tokenizer("I am happy to learn the transformer library")
```

Tokenizer is used to print all the indexs of words

```
print(inputs)
```

```
{'input_ids': [101, 1045, 2572, 3407, 2000, 4553, 1996, 10938, 2121, 3075, 102], 'attention_mask': [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1
```

```
tf_batch = tokenizer(["I am happy to learn the transformer model"],
    padding = True,
    truncation = True,
    max_length = 512,
    return_tensors = "tf"

)
```

padding will help us in creating the words of sentence in the same dimention

```
for key, value in tf_batch.items():
  print(f"{key}.{value.numpy().tolist()}")
```

```
input_ids.[[101, 1045, 2572, 3407, 2000, 4553, 1996, 10938, 2121, 2944, 102]]
attention_mask.[[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]]
```

Colab paid products - Cancel contracts here

✓ 0s completed at 9:03 AM ● ✕