[COSCUP 2022] Google Colab + Hugging Face：帶你快速認識NLP

作者：江尚軒 (Andy Chiang)

這是 **Google Colab + Hugging Face：帶你快速認識NLP** 的範例程式碼。

Before you start...

1. 請先點 **檔案 >> 在雲端硬碟中儲存副本**
2. 然後切換到你自己的副本
3. 接著點 **執行階段 >> 變更執行階段類型**
4. 硬體加速器選 **GPU**
5. 點擊右上角 **連線** 就可以開始執行了!

Install packages

因為 Colab 內建沒有 transformers 和 datasets 套件，所以要先用底下的指令下載。

(如果出現紅字，是因為套件版本有衝突，重跑一次就好了)

```
!pip install transformers datasets
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/si
Requirement already satisfied: transformers in /usr/local/lib/python3.7/dist-packages (4.20.1
Requirement already satisfied: datasets in /usr/local/lib/python3.7/dist-packages (2.3.2)
Requirement already satisfied: importlib-metadata in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.7/dist-packages (from tra
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.7/dist-packages (fro
Requirement already satisfied: filelock in /usr/local/lib/python3.7/dist-packages (from trans
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.7/dist-packages (from tr
Requirement already satisfied: tokenizers!=0.11.3,<0.13,>=0.11.1 in /usr/local/lib/python3.7/
Requirement already satisfied: huggingface-hub<1.0,>=0.1.0 in /usr/local/lib/python3.7/dist-p
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.7/dist-packages (from tr
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.7/dist-packages (f
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-packages (from trans
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.7/dist-pa
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /usr/local/lib/python3.7/dist-pack
Requirement already satisfied: pyarrow>=6.0.0 in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: xxhash in /usr/local/lib/python3.7/dist-packages (from dataset
Requirement already satisfied: aiohttp in /usr/local/lib/python3.7/dist-packages (from datase
Requirement already satisfied: responses<0.19 in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: dill<0.3.6 in /usr/local/lib/python3.7/dist-packages (from dat
Requirement already satisfied: pandas in /usr/local/lib/python3.7/dist-packages (from dataset
Requirement already satisfied: fsspec[http]>=2021.05.0 in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: multiprocess in /usr/local/lib/python3.7/dist-packages (from d
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packages (f
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/pyth
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (from r
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: charset-normalizer<3.0,>=2.0 in /usr/local/lib/python3.7/dist-
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: yarl<2.0,>=1.0 in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.7/dist-packages (fr
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.7/dist-packages (f
Requirement already satisfied: asynctest==0.13.0 in /usr/local/lib/python3.7/dist-packages (f
Requirement already satisfied: async-timeout<5.0,>=4.0.0a3 in /usr/local/lib/python3.7/dist-p
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-packages (from impo
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.7/dist-packag
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/dist-packages (from p
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages (from pytho
```

Pipeline

使用 `pipeline()` 是完成NLP任務最簡單的方式。

各種NLP任務

首先，引用 `pipeline()` 並指定任務為**情感分析 (sentiment-analysis)**，它會自動幫你抓最好的模型下來。

```
from transformers import pipeline

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

```
No model was supplied, defaulted to distilbert-base-uncased-finetuned-sst-2-english (
Downloading: 100%                                    629/629 [00:00<00:00, 17.4kB/s]
Downloading: 100%                                    255M/255M [00:04<00:00, 56.9MB/s]
Downloading: 100%                                    48.0/48.0 [00:00<00:00, 1.59kB/s]
Downloading: 100%                                    226k/226k [00:00<00:00, 966kB/s]
```

然後就隨便給模型一句話 "This movie is interesting."。 你看! 它就能正確分析出這句話是正面的，而且分數有0.99。

```
classifier("This movie is interesting!")
```

```
[{'label': 'POSITIVE', 'score': 0.99983811378479}]
```

換給它 "This movie is boring."，也能正確分析這句話是負面的。

```
classifier("This  movie  is  boring!")
```

```
[{'label': 'NEGATIVE', 'score': 0.9998012185096741}]
```

可見使用 `pipeline()` 就是如此簡單，只要兩三行程式碼就完成了，你甚至不需要了解資料預處理或模型架構! 因為 Hugging Face 都已經幫你封裝好了。

接著，我們換做別的任務。

我們先試試看**文本生成** **(text-generation)**。

```
generator  =  pipeline("text-generation")
```

```
No model was supplied, defaulted to gpt2 (https://huggingface.co/gpt2)
Downloading: 100%                                    665/665 [00:00<00:00, 19.6kB/s]
Downloading: 100%                                    523M/523M [00:09<00:00, 57.9MB/s]
Downloading: 100%                                    0.99M/0.99M [00:00<00:00, 1.67MB/s]
Downloading: 100%                                    446k/446k [00:00<00:00, 899kB/s]
Downloading: 100%                                    1.29M/1.29M [00:00<00:00, 1.73MB/s]
```

給模型三隻小豬的第一句，看看它會生成什麼東西出來。

生成的結果雖然很荒謬，但至少還看起來滿流暢的，對吧?

```
generator("Once  upon  a  time  there  were  three  little  pigs.")
```

```
The attention mask and the pad token id were not set. As a consequence, you may observe unexp
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
[{'generated_text': "Once upon a time there were three little pigs. She was pregnant with
two, one her father and one his mother. The baby boy turned five months old on the day of
the baby's birth. He and his mother got divorced. He had to"}]
```

最後試試看**問答** **(question-answering)**。

```
qa  =  pipeline("question-answering")
```

```
No model was supplied, defaulted to distilbert-base-cased-distilled-squad (https://hu
```

給模型一段有關哈利波特的介紹，然後問它哈利波特的作者是誰? 它真的就回答J. K. Rowling!

```
context = """
Harry Potter is a series of seven fantasy novels written by British author J. K.
The novels chronicle the lives of a young wizard, Harry Potter, and his friends He
all of whom are students at Hogwarts School of Witchcraft and Wizardry. The main s
Harry's struggle against Lord Voldemort, a dark wizard who intends to become immorta
governing body known as the Ministry of Magic and subjugate all wizards and Muggles
"""
question = "Who is the author of Harry Potter?"

qa(question, context)
```

```
{'answer': 'J. K. Rowling',
 'end': 88,
 'score': 0.9453211426734924,
 'start': 75}
```

使用自訂模型

剛才是使用預設的模型，其實也可以使用自己想要的模型。

回到**情感分析 (sentiment-analysis)** 的任務，剛才是英文的，現在想用中文的要怎麼做呢?

1. 首先到 Hugging Face 的 models 頁面，右邊會看到許多模型，左邊則有篩選標籤。
2. 因為情感分析算是一種文本分類的任務，所以要選 **Text Classification**。
3. 再來，語言選擇 **Chinese**。
4. 找一個情感分析的模型來用，我選 **IDEA-CCNL/Erlangshen-Roberta-110M-Sentiment** 當作例子
5. 我們要使用此模型，必須用到 **AutoModelForSequenceClassification** 和 **AutoTokenizer** 這兩個物件，稍微介紹一下：
    - **AutoModelForSequenceClassification**：指定任務的模型，像這裡就是 SequenceClassification。如果不確定模型的架構是哪一種 (如BERT、RoBERTa...)，就使用 AutoModel ，套件會自動幫你決定。
    - **AutoTokenizer**：因為模型不可能直接吃文字，所以 Tokenizer 的功能是把句子切成一個個單字，然後轉成對應的 token。
6. 下載模型需要等待一下。

```
from transformers import AutoModelForSequenceClassification, AutoTokenizer

model = AutoModelForSequenceClassification.from_pretrained("IDEA-CCNL/Erlangshen-Roberta-110M-
tokenizer = AutoTokenizer.from_pretrained("IDEA-CCNL/Erlangshen-Roberta-110M-Sentiment")
```

```
Downloading: 100%                                   749/749 [00:00<00:00, 22.0kB/s]

Downloading: 100%                                   107k/107k [00:00<00:00, 1.32MB/s]

Downloading: 100%                                   390M/390M [00:09<00:00, 27.3MB/s]
```

和剛才一樣，只是 model 和 tokenizer 的部分改成自訂的

```
from transformers import pipeline

classifier = pipeline("sentiment-analysis", model=model, tokenizer=tokenizer)
```

然後就可以用啦!

```
classifier("這部電影真有趣!")
```

```
[{'label': 'Positive', 'score': 0.7309864163398743}]
```

```
classifier("這部電影真無聊!")
```

```
[{'label': 'Negative', 'score': 0.9995669722557068}]
```

Data preprocessing

接下來，我們試試看自己 Fine-tune 一個情感分析模型。但在這之前，必須先做資料預處理。

我們利用 Hugging Face 的 **datasets** 套件，下載 yelp_review_full 這個資料集。資料來源是 Yelp 網站上用戶對於各店家的評價。

```
from datasets import load_dataset

dataset = load_dataset("yelp_review_full")
```

```
Downloading builder script:                              4.41k/? [00:00<00:00, 80.1kB/s]

Downloading metadata:                                  2.04k/? [00:00<00:00, 20.1kB/s]
Downloading and preparing dataset yelp_review_full/yelp_review_full (download: 187.06
Downloading data: 100%                                  196M/196M [00:05<00:00, 50.0MB/

Generating train split: 100%                              648769/650000 [00:34<00:00, 19

Generating test split: 97%                              48664/50000 [00:06<00:00, 15988
Dataset yelp_review_full downloaded and prepared to /root/.cache/huggingface/datasets
100%                                          2/2 [00:00<00:00, 33.91it/s]
```

取其中一筆資料來看，可以發現有兩個欄位，text 是用戶評論，label 則是評分 (0~4 代表 1~5 顆星)

```
dataset["train"][0]
```

```
{'label': 4,
 'text': "dr. goldberg offers everything i look for in a general practitioner.  he's nice
and easy to talk to without being patronizing; he's always on time in seeing his patients;
he's affiliated with a top-notch hospital (nyu) which my parents have explained to me is
very important in case something happens and you need surgery; and you can get referrals to
see specialists without having to see him first.  really, what more do you need?  i'm
sitting here trying to think of any complaints i have about him, but i'm really drawing a
blank."}
```

然後看一下 train 和 test 資料集的大小

```
print(len(dataset["train"]))
print(len(dataset["test"]))
```

```
650000
50000
```

因為資料集太大會訓練很久，所以我 train 和 test 的資料只取4000和1000就好。

```
dataset["train"] = dataset["train"].shuffle().select(range(4000))
dataset["test"] = dataset["test"].shuffle().select(range(1000))
```

一樣會用到剛才提到的 **AutoTokenizer**，用來將文字轉成數字。

bert-base-cased 則是待會要用到的模型名稱。

```
from transformers import AutoTokenizer

tokenizer = AutoTokenizer.from_pretrained("bert-base-cased")
```

```
Downloading: 100%                                   29.0/29.0 [00:00<00:00, 653B/s]
Downloading: 100%                                   570/570 [00:00<00:00, 12.4kB/s]
Downloading: 100%                                   208k/208k [00:00<00:00, 1.07MB/s]
Downloading: 100%                                   426k/426k [00:00<00:00, 1.59MB/s]
```

然後，我們寫一個 function，輸入是文字，輸出是處理好的 token。

tokenizer 這裡多加了 **padding** 和 **truncation** 這兩個參數。這是為了讓模型的輸入長度相同，太短的句子會補上 padding，太長的句子則截斷。

```
def  tokenize_function(examples):
        return  tokenizer(examples["text"],  padding=True,  truncation=True)
```

然後使用 `map()` 將資料集的文字都轉為 token。

```
tokenized_datasets  =  dataset.map(tokenize_function,  batched=True)
```

```
Parameter 'function'=<function tokenize_function at 0x7f92e48eb950> of the transform
100%                                                    4/4 [00:01<00:00, 2.08ba/s]

100%                                                    1/1 [00:00<00:00, 2.02ba/s]
```

Fine-tune a model

資料處理完後，就可以來 Fine-tune 模型了。

首先，我們要用 **bert-base-cased** 這個模型，這就是最原始、完全沒有 Fine-tune 過的 BERT 模型。 `num_labels=5` 表示有標籤有5類 (0~4)

```
from  transformers  import  AutoModelForSequenceClassification

model  =  AutoModelForSequenceClassification.from_pretrained("bert-base-cased",  num_labels=5)
```

```
Downloading: 100%                                          416M/416M [00:07<00:00, 55.9MB/s]
Some weights of the model checkpoint at bert-base-cased were not used when initializi
- This IS expected if you are initializing BertForSequenceClassification from the che
- This IS NOT expected if you are initializing BertForSequenceClassification from the
Some weights of BertForSequenceClassification were not initialized from the model che
You should probably TRAIN this model on a down-stream task to be able to use it for p
```

使用 `load_metric("accuracy")` 引用正確率作為衡量指標。

```
import  numpy  as  np
from  datasets  import  load_metric

metric  =  load_metric("accuracy")
```

```
Downloading builder script:                                    4.21k/? [00:00<00:00, 16.1kB/s]
```

一樣寫個 function，輸入預測結果和正確標籤，計算正確率並回傳結果。

```
def  compute_metrics(eval_pred):
        logits, labels = eval_pred
        predictions = np.argmax(logits, axis=-1)
        return metric.compute(predictions=predictions, references=labels)
```

引用 **TrainingArguments** 作為訓練的超參數，`output_dir="test_trainer"` 表示存檔名稱，`evaluation_strategy="epoch"` 則表示每一個 epoch 就會衡量一次正確率，其餘超參數預設就好。

```
from transformers import TrainingArguments

training_args = TrainingArguments(output_dir="test_trainer", evaluation_strategy="epoch")
```

建立 **Trainer** 物件，將剛剛的一堆東西丟進去...

```
from transformers import Trainer

trainer = Trainer(
        model=model,
        args=training_args,
        train_dataset=tokenized_datasets["train"],
        eval_dataset=tokenized_datasets["test"],
        compute_metrics=compute_metrics,
)
```

呼叫 `trainer.train()`，然後就結束了。就會自動開始訓練模型了，是不是很簡單啊!?

(4000筆 train data 需要訓練大約22分鐘，正確率大約60%。雖然看起來不高，但如果使用更多訓練資料，正確率應該會再更高。)

```
trainer.train()
```

```
The following columns in the training set don't have a corresponding argument in `Ber
/usr/local/lib/python3.7/dist-packages/transformers/optimization.py:310: FutureWarnir
  FutureWarning,
***** Running training *****
  Num examples = 4000
  Num Epochs = 3
  Instantaneous batch size per device = 8
  Total train batch size (w. parallel, distributed & accumulation) = 8
  Gradient Accumulation steps = 1
  Total optimization steps = 1500
```

[1500/1500 22:17, Epoch 3/3]

Epoch	Training Loss	Validation Loss	Accuracy
1	1.307200	0.988094	0.577000
2	0.936900	0.970908	0.596000
3	0.621400	1.052324	0.606000

```
Saving model checkpoint to test_trainer/checkpoint-500
Configuration saved in test_trainer/checkpoint-500/config.json
Model weights saved in test_trainer/checkpoint-500/pytorch_model.bin
The following columns in the evaluation set don't have a corresponding argument in `E
***** Running Evaluation *****
  Num examples = 1000
  Batch size = 8
Saving model checkpoint to test_trainer/checkpoint-1000
Configuration saved in test_trainer/checkpoint-1000/config.json
Model weights saved in test_trainer/checkpoint-1000/pytorch_model.bin
The following columns in the evaluation set don't have a corresponding argument in `E
```

Save a model

```
Saving model checkpoint to test_trainer/checkpoint-1500
```

訓練完後，可以使用 save_model() 這個函數將模型下載下來，模型會下載在 Colab 的虛擬機器中。

```
  Num examples = 1000
```

```
trainer.save_model("./test_model")
```

```
Saving model checkpoint to ./test_model
Configuration saved in ./test_model/config.json
Model weights saved in ./test_model/pytorch_model.bin

TrainOutput(global_step=1500, training_loss=0.9551491088867188, metrics={'train_runti
```

使用 from_pretrained() 就可以載入模型來使用。

```
from transformers import AutoModelForSequenceClassification

pt_model = AutoModelForSequenceClassification.from_pretrained("./test_model")
```

```
loading configuration file ./test_model/config.json
Model config BertConfig {
  "_name_or_path": "./test_model",
  "architectures": [
    "BertForSequenceClassification"
  ],
  "attention_probs_dropout_prob": 0.1,
  "classifier_dropout": null,
  "gradient_checkpointing": false,
  "hidden_act": "gelu",
  "hidden_dropout_prob": 0.1,
  "hidden_size": 768,
  "id2label": {
    "0": "LABEL_0",
    "1": "LABEL_1",
    "2": "LABEL_2",
    "3": "LABEL_3",
    "4": "LABEL_4"
  },
  "initializer_range": 0.02,
  "intermediate_size": 3072,
  "label2id": {
    "LABEL_0": 0,
    "LABEL_1": 1,
    "LABEL_2": 2,
    "LABEL_3": 3,
    "LABEL_4": 4
  },
  "layer_norm_eps": 1e-12,
  "max_position_embeddings": 512,
  "model_type": "bert",
  "num_attention_heads": 12,
  "num_hidden_layers": 12,
  "pad_token_id": 0,
  "position_embedding_type": "absolute",
  "problem_type": "single_label_classification",
  "torch_dtype": "float32",
  "transformers_version": "4.20.1",
  "type_vocab_size": 2,
  "use_cache": true,
  "vocab_size": 28996
}

loading weights file ./test_model/pytorch_model.bin
All model checkpoint weights were used when initializing BertForSequenceClassification.

All the weights of BertForSequenceClassification were initialized from the model checkpoint a
If your task is similar to the task the model of the checkpoint was trained on, you can alrea
```

Hugging Face 還有一個很強大的功能，就在於 Pytorch 和 TensorFlow 的模型可以輕易轉換。

剛才下載的是 Pytorch 的模型，但只要在使用 TensorFlow 的模型載入時加上 `from_pt=True`，就可以轉換成功了!

```
from transformers import TFAutoModelForSequenceClassification

tf_model = TFAutoModelForSequenceClassification.from_pretrained("./test_model", from_pt=True)
```

```
loading configuration file ./test_model/config.json
Model config BertConfig {
  "_name_or_path": "./test_model",
  "architectures": [
    "BertForSequenceClassification"
  ],
  "attention_probs_dropout_prob": 0.1,
  "classifier_dropout": null,
  "gradient_checkpointing": false,
  "hidden_act": "gelu",
  "hidden_dropout_prob": 0.1,
  "hidden_size": 768,
  "id2label": {
    "0": "LABEL_0",
    "1": "LABEL_1",
    "2": "LABEL_2",
    "3": "LABEL_3",
    "4": "LABEL_4"
  },
  "initializer_range": 0.02,
  "intermediate_size": 3072,
  "label2id": {
    "LABEL_0": 0,
    "LABEL_1": 1,
    "LABEL_2": 2,
    "LABEL_3": 3,
    "LABEL_4": 4
  },
  "layer_norm_eps": 1e-12,
  "max_position_embeddings": 512,
  "model_type": "bert",
  "num_attention_heads": 12,
  "num_hidden_layers": 12,
  "pad_token_id": 0,
  "position_embedding_type": "absolute",
  "problem_type": "single_label_classification",
  "torch_dtype": "float32",
  "transformers_version": "4.20.1",
  "type_vocab_size": 2,
  "use_cache": true,
  "vocab_size": 28996
}

loading weights file ./test_model/pytorch_model.bin
Loading PyTorch weights from /content/test_model/pytorch_model.bin
PyTorch checkpoint contains 108,314,629 parameters
Loaded 108,314,117 parameters in the TF 2.0 model.
Some weights of the PyTorch model were not used when initializing the TF 2.0 model TFBertForS
- This IS expected if you are initializing TFBertForSequenceClassification from a PyTorch moc
- This IS NOT expected if you are initializing TFBertForSequenceClassification from a PyTorch
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 alrea
```

Share a model

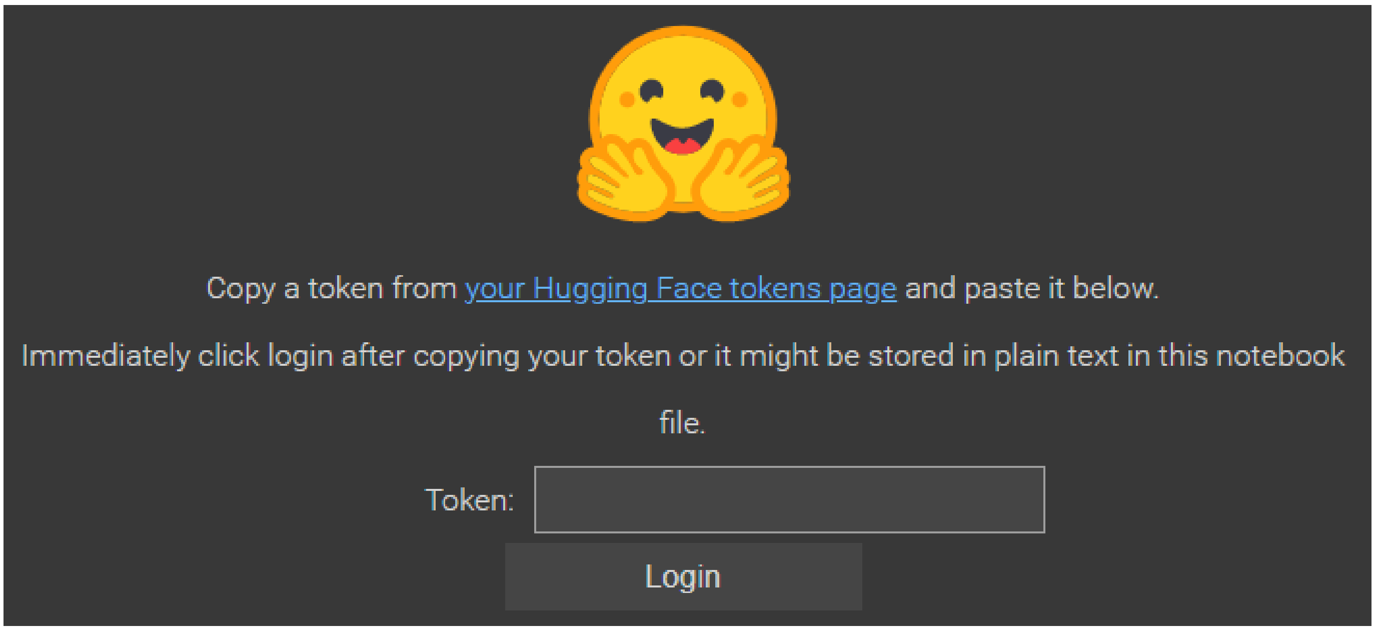
都來 COSCUP 了，就來講點開源的事情吧! 相信我，這非常的簡單!

如果你想把 Fine-tune 好的模型分享給全世界的人，首先你需要下載 **huggingface_hub** 套件。

```
!pip  install  huggingface_hub
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/si
Collecting huggingface_hub
  Downloading huggingface_hub-0.8.1-py3-none-any.whl (101 kB)
    |████████████████████████████████| 101 kB 4.5 MB/s
Requirement already satisfied: filelock in /usr/local/lib/python3.7/dist-packages (from huggi
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-packages (from huggi
Requirement already satisfied: importlib-metadata in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.7/dist-pa
Requirement already satisfied: packaging>=20.9 in /usr/local/lib/python3.7/dist-packages (frc
Requirement already satisfied: tqdm in /usr/local/lib/python3.7/dist-packages (from huggingfa
Collecting pyyaml>=5.1
  Downloading PyYAML-6.0-cp37-cp37m-manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_12_x86
    |████████████████████████████████| 596 kB 28.5 MB/s
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /usr/local/lib/python3.7/dist-pack
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-packages (from impc
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/pyth
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (from r
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packages (f
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packages (
Installing collected packages: pyyaml, huggingface-hub
  Attempting uninstall: pyyaml
    Found existing installation: PyYAML 3.13
    Uninstalling PyYAML-3.13:
      Successfully uninstalled PyYAML-3.13
Successfully installed huggingface-hub-0.8.1 pyyaml-6.0
```

下一步要登入，執行後會出現如同下面的提示，要求你輸入一個 access token。



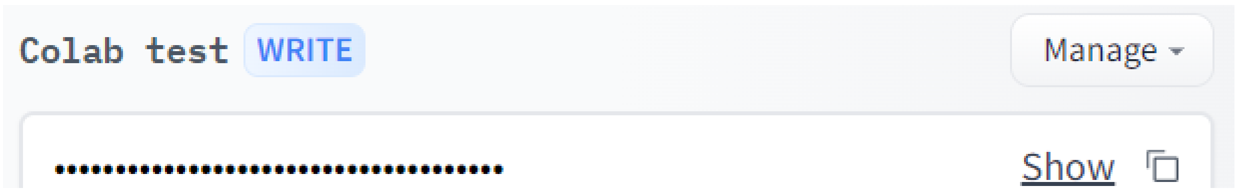
Copy a token from your Hugging Face tokens page and paste it below.

Immediately click login after copying your token or it might be stored in plain text in this notebook file.

Token:

Login

點擊提示裡的連結，會傳到 Hugging Face 的網站，新增一個權限是 **write** 的 token。



```
from huggingface_hub import notebook_login

notebook_login()
```

```
Login successful
Your token has been saved to /root/.huggingface/token
Authenticated through git-credential store but this isn't the helper defined on y
You might have to re-authenticate when pushing to the Hugging Face Hub. Run the f

git config --global credential.helper store
```

再來就只要把剛才 Fine-tune 好的模型 push 到 Hugging Face 上就行了，它會自動地幫你創立新的 repo。

```
pt_model.push_to_hub("my-test-model")
```

```
/usr/local/lib/python3.7/dist-packages/huggingface_hub/utils/_deprecation.py:43: Futu
  FutureWarning,
/usr/local/lib/python3.7/dist-packages/huggingface_hub/hf_api.py:644: FutureWarning:
  FutureWarning,
Cloning https://huggingface.co/AndyChiang/my-test-model into local empty directory.
Configuration saved in my-test-model/config.json
Model weights saved in my-test-model/pytorch_model.bin
Upload file pytorch_model.bin: 100%                              413M/413M [05:24<00:
To https://huggingface.co/AndyChiang/my-test-model
   da541a5..9035379  main -> main

'https://huggingface.co/AndyChiang/my-test-model/commit/9035379b12a638eeb8e54d38cdc3
```

雖然說 Transformers 可以自動轉換 Pytorch 和 TensorFlow 的模型，但會比較花時間。所以為了使用者方便，基本上建議你將兩種模型都 push 上去比較好。

```
tf_model.push_to_hub("my-test-model")
```

```
Configuration saved in my-test-model/config.json
Model weights saved in my-test-model/tf_model.h5
Upload file tf_model.h5: 100%                              413M/413M [05:18<00:00, 90
To https://huggingface.co/AndyChiang/my-test-model
   9c4ac9f..e6392a9  main -> main

'https://huggingface.co/AndyChiang/my-test-model/commit/e6392a97d572dd50121bb398803a
```

也把剛才用的 tokenizer 給 push 上去。

```
tokenizer.push_to_hub("my-test-model")
```

```
tokenizer config file saved in my-test-model/tokenizer_config.json
Special tokens file saved in my-test-model/special_tokens_map.json
To https://huggingface.co/AndyChiang/my-test-model
   9035379..9c4ac9f  main -> main

'https://huggingface.co/AndyChiang/my-test-model/commit/9c4ac9fee5b0a98c227c918bef62
```

以後就可以下載自己的模型來用啦! 名稱是 `<user name>/<model name>`

```
from transformers import AutoModelForSequenceClassification, AutoTokenizer

my_model = AutoModelForSequenceClassification.from_pretrained("AndyChiang/my-test-model")
my_tokenizer = AutoTokenizer.from_pretrained("AndyChiang/my-test-model")
```

```
loading configuration file https://huggingface.co/AndyChiang/my-test-model/resolve/main/confi
Model config BertConfig {
  "_name_or_path": "AndyChiang/my-test-model",
  "architectures": [
    "BertForSequenceClassification"
  ],
  "attention_probs_dropout_prob": 0.1,
  "classifier_dropout": null,
  "gradient_checkpointing": false,
  "hidden_act": "gelu",
  "hidden_dropout_prob": 0.1,
  "hidden_size": 768,
  "id2label": {
    "0": "LABEL_0",
    "1": "LABEL_1",
    "2": "LABEL_2",
    "3": "LABEL_3",
    "4": "LABEL_4"
  },
  "initializer_range": 0.02,
  "intermediate_size": 3072,
  "label2id": {
    "LABEL_0": 0,
    "LABEL_1": 1,
    "LABEL_2": 2,
    "LABEL_3": 3,
    "LABEL_4": 4
  },
  "layer_norm_eps": 1e-12,
  "max_position_embeddings": 512,
  "model_type": "bert",
  "num_attention_heads": 12,
  "num_hidden_layers": 12,
  "pad_token_id": 0,
  "position_embedding_type": "absolute",
  "problem_type": "single_label_classification",
  "torch_dtype": "float32",
  "transformers_version": "4.20.1",
  "type_vocab_size": 2,
  "use_cache": true,
  "vocab_size": 28996
}

loading weights file https://huggingface.co/AndyChiang/my-test-model/resolve/main/pytorch_moc
All model checkpoint weights were used when initializing BertForSequenceClassification.

All the weights of BertForSequenceClassification were initialized from the model checkpoint a
If your task is similar to the task the model of the checkpoint was trained on, you can alrea
loading file https://huggingface.co/AndyChiang/my-test-model/resolve/main/vocab.txt from cach
loading file https://huggingface.co/AndyChiang/my-test-model/resolve/main/tokenizer.json from
loading file https://huggingface.co/AndyChiang/my-test-model/resolve/main/added_tokens.json f
loading file https://huggingface.co/AndyChiang/my-test-model/resolve/main/special_tokens_map.
loading file https://huggingface.co/AndyChiang/my-test-model/resolve/main/tokenizer_config.js
```

用自己的模型來測試情感分析。

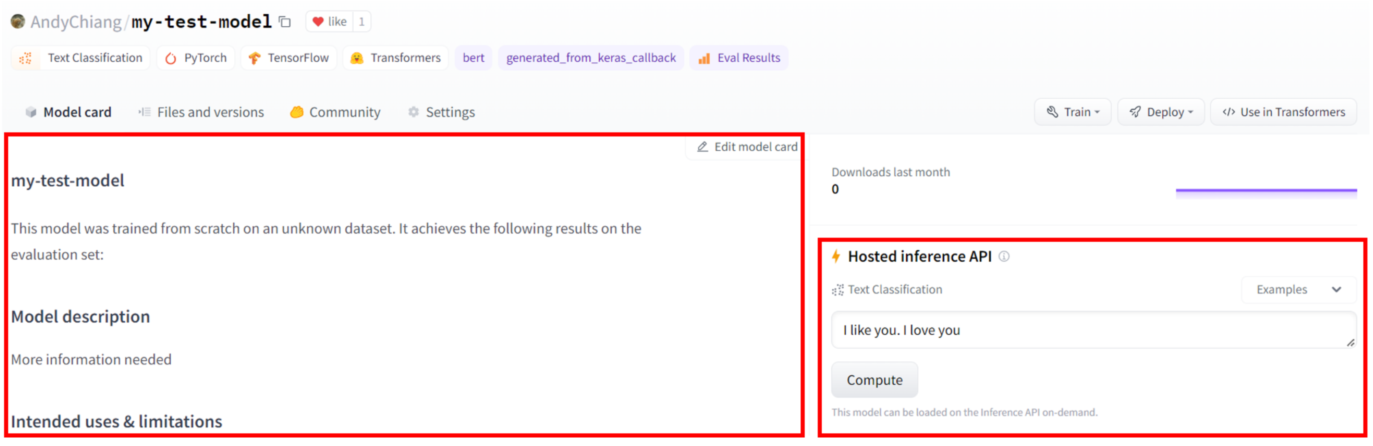
```
from transformers import pipeline

classifier = pipeline("sentiment-analysis", model=my_model, tokenizer=my_tokenizer)
```

結果還不錯!

```
classifier("This restaurant is great! The food there is delicious, too.")
```

```
[{'label': 'LABEL_4', 'score': 0.9106563329696655}]
```

最後提一下網站的部分，你可以到 Hugging Face 的網站上查看自己的模型，以我的模型為例。你可以編輯 **model card** (類似 GitHub README)，寫上一些關於模型的資訊，也可以用 **Hosted inference API** 來做些簡單的測試。



my-test-model

This model was trained from scratch on an unknown dataset. It achieves the following results on the evaluation set:

Model description

More information needed

Intended uses & limitations

I like you. I love you

This model can be loaded on the Inference API on-demand.

Summary

恭喜你看完整個教學了! 相信你現在已經有基礎的 NLP 知識，也可以自己 Fine-tune 模型了! 不過這還只是基礎而已，NLP 還有很多東西沒講到，但剩下的就留給有興趣的讀者自行研究吧! 也歡迎分享你的學習心得~

哦對了! 很推薦對 NLP 有興趣的人去看 Transformers 的 document，內容非常豐富且完整。此教學大部分也是參考這個 document 的哦🤗