環境: tensorflow2.4.0/ python3.8.5/ protobuf3.19.0/ numpy1.19.2/ bert-for-tf2 0.14.9

Tensorflow- transform BERT model to tflite and deployed on TPU

Tensorflow1.x

- 轉換

- 如果是要用 tensorflow1.x 轉的話，github 上有人寫好了

Tensorflow2.x

- 轉換

方法(一):

- 使用 Optimum

方法(二):

- 使用 **tflite_model_maker**
- 從 Hugging Face Transformers 庫取出模型

```
from transformers import AutoTokenizer, TFMobileBertForQuestionAnswering
import tensorflow as tf

desired_model = "vumichien/mobilebert-uncased-squad-v2" #要調用的模型
tokenizer = AutoTokenizer.from_pretrained(desired_model)
model = TFMobileBertForQuestionAnswering.from_pretrained(desired_model)

question, text = "Who was Jim Henson?", "Jim Henson was a nice puppet"

inputs = tokenizer(question, text, return_tensors="tf")
outputs = model(**inputs)

answer_start_index = int(tf.math.argmax(outputs.start_logits, axis=-1)[0])
answer_end_index = int(tf.math.argmax(outputs.end_logits, axis=-1)[0])
```

```
predict_answer_tokens = inputs.input_ids[0, answer_start_index :
answer_end_index + 1]
tokenizer.decode(predict_answer_tokens)
```

- 使用 TFLite 模型

➢ 參考自 github。設定一個讀取模型用的 class

```
import tensorflow as tf
import bert
import numpy as np
import tflite_runtime.interpreter as tflite
import platform

EDGETPU_SHARED_LIB = {'Linux': 'libedgetpu.so.1' ,
                      'Darwin': 'libedgetpu.1.dylib',
                      'Windows': 'edgetpu.dll'}[platform.system()]
def make_interpreter(model_file):
    model_file , *device = model_file.split('@')
    return tflite.Interpreter(model_path = model_file ,
                              experimental_delegates =
[tflite.load_delegate(EDGETPU_SHARED_LIB ,{'device': device[0]} if device
else {})])

class MobileBERT:
    def __init__(self, tflite_path, tokenizer_file_path):
        self.max_length = 384
        self.interpreter = make_interpreter(tflite_path)
        self.tokenizer =
bert.bert_tokenization.FullTokenizer(tokenizer_file_path, True)
        self.interpreter.allocate_tensors()
        self.input_details = self.interpreter.get_input_details()
        self.output_details = self.interpreter.get_output_details()

    def get_summary(self):
        print("Inputs:",self.input_details,"\nOutputs:",self.output_details
)

    def get_masks(self,tokens):
```

環境: tensorflow2.4.0/ python3.8.5/ protobuf3.19.0/ numpy1.19.2/ bert-for-tf2 0.14.9

```
        if len(tokens)>self.max_length:
            raise IndexError("Token length more than max seq length!")
        return np.asarray([1]*len(tokens) + [0] * (self.max_length -
len(tokens)))


    def get_segments(self,tokens):
        if len(tokens)>self.max_length:
            raise IndexError("Token length more than max seq length!")
        segments = []
        current_segment_id = 0
        for token in tokens:
            segments.append(current_segment_id)
            if token == "[SEP]":
                current_segment_id = 1
        return np.asarray(segments + [0] * (self.max_length - len(tokens)))


    def get_ids(self,tokens):
        token_ids = self.tokenizer.convert_tokens_to_ids(tokens)
        input_ids = token_ids + [0] * (self.max_length-len(token_ids))
        return np.asarray(input_ids)

    def compile_text(self,text):
        text = text.lower().replace("-"," ")
        return ["[CLS]"] + self.tokenizer.tokenize(text) + ["[SEP]"]

    def run(self,query,context):
        stokens =  self.compile_text(query) + self.compile_text(context)

        if len(stokens)>self.max_length:
            raise IndexError("Token length more than max seq length!")
            print("Max exceeded")
        input_ids = tf.dtypes.cast(self.get_ids(stokens),tf.int32)
        input_masks = tf.dtypes.cast(self.get_masks(stokens),tf.int32)
        input_segments =
tf.dtypes.cast(self.get_segments(stokens),tf.int32)
```

```
        self.interpreter.set_tensor(self.input_details[0]['index'],
[input_ids])
        self.interpreter.set_tensor(self.input_details[1]['index'],
[input_masks])
        self.interpreter.set_tensor(self.input_details[2]['index'],
[input_segments])

        with tf.device('/CPU:0'):
            self.interpreter.invoke()

        end_logits =
self.interpreter.get_tensor(self.output_details[0]['index'])
        start_logits =
self.interpreter.get_tensor(self.output_details[1]['index'])

        end = tf.argmax(end_logits,output_type=tf.dtypes.int32).numpy()[0]
        start =
tf.argmax(start_logits,output_type=tf.dtypes.int32).numpy()[0]

        answers = "
".join(stokens[start:end+1]).replace("[CLS]","").replace("[SEP]","").replac
e(" ##","")
        return answers
```

- 取用模型。註: 此處的 tflite 為另外下載的。vocab.txt 為給 tokenizer 映射用的檔案，一定要有。

```
m = MobileBERT('lite-model_mobilebert_1_metadata_1.tflite','vocab.txt')
answer = m.run(
"The Apollo program, also known as Project Apollo, was the third United
States human spaceflight program carried out by NASA, which succeeded in
landing the first humans on the Moon from 1969 to 1972.",
"What was the goal of the Apollo program?"
)
print(answer)
print("***運作到這邊還沒東西就是真的沒答案啦***")
```