**運用BERT執行文件自動分類**

曾元顯

2019/09/27 第一版, 2019/10/02第二版

2020/02/08第三版, 2020/08/24第四版

**底下的說明，是在Mac的環境（一種Linux作業系統）上執行。**

若是Windows環境，建議安裝Windows Subsystem for Linux（簡稱WSL，參考：<https://blog.miniasp.com/post/2019/02/01/Useful-tool-WSL-Windows-Subsystem-for-Linux>），在WSL中執行Linux的命令。安中WSL可能會用到PowerShell，其已內建於Windows 10中，搜尋PowerShell然後執行它（也許要用系統管理員身份執行）。

1. **前置作業：**
2. 安裝Anaconda：從<https://www.anaconda.com/distribution/>下載跟作業系統匹配的安裝檔。
3. 以Anaconda製作一個Python3.6版的環境（在此例名稱為py3.6），然後啟動：

$ conda create -n py3.6 anaconda python=3.6

$ conda activate py3.6

1. 安裝git：<https://www.atlassian.com/git/tutorials/install-git>。  
   不安裝也可以啦！
2. **下載BERT程式碼以及預先訓練的模型：**
3. 下載BERT程式：可用git clone 方式（用 Download 方式也可以）：

$ cd **/Users/sam/GoogleDrive**

其中sam為登入電腦的使用者名稱，可改成自己的名稱。

$ mkdir BERT

$ cd BERT

$ git clone <https://github.com/google-research/bert.git>

或是直接從<https://github.com/google-research/bert>，下載壓縮檔，並解壓縮到**/Users/sam/GoogleDrive/BERT/bert/**目錄下。

1. 下載並解壓縮預先訓練好的模型（前培訓模型），參考：<https://blog.gtwang.org/linux/linux-wget-command-download-web-pages-and-files-tutorial-examples/>：

可用Unix的wget命令下載：如果下載大型檔案中途斷線，檔案只下載了一部分，可以使用 wget 的 -c 參數從上次中斷的地方繼續下載：

$ wget -c <https://storage.googleapis.com/bert_models/2018_11_03/chinese_L-12_H-768_A-12.zip>

在Windows的話，可直接從瀏覽器網址輸入：<https://storage.googleapis.com/bert_models/2018_11_03/chinese_L-12_H-768_A-12.zip> 下載到**GoogleDrive/BERT/bert/**目錄下。

然後以下列命令解壓縮（或是在視窗上手動解壓縮）：

$ unzip chinese\_L-12\_H-768\_A-12.zip

1. 確認上述Google的bert所有的程式碼、預訓練模型目錄、以及此目錄下的程式都在同一目錄下： **GoogleDrive/BERT/bert/**。
2. **修改BERT程式碼、準備分類資料：**

參考：<https://leowood.github.io/2018/11/16/BERT-fine-tuning-01/>，步驟如下：

1. 準備一個有12GB左右RAM的GPU，**沒有也沒關係**，可以使用谷歌免費的GPU（請見後面說明）。若只在自己電腦上用CPU上執行，訓練232筆CnonC資料（計畫名稱的二元分類："001-營建類", "002-非營建類"），約需2~3小時。
2. 準備好（CnonC）分類資料的train.txt，dev.txt以及test.txt，放在如： ../Datasets目錄下。
3. 保留原始run\_classifier.py以供爾後參考，所以拷貝run\_classifier.py到run\_classifier\_sam.py，修改其程式，新建一個跟自己task\_name對應的處理器（processor），亦即大約在第206行插入下面的程式碼：

class **CnonC\_Processor**(DataProcessor):

def get\_train\_examples(self, data\_dir):

return self.\_create\_examples(

self.\_read\_tsv(os.path.join(data\_dir, "CnonC\_train.txt")), "train")

def get\_dev\_examples(self, data\_dir):

return self.\_create\_examples(

self.\_read\_tsv(os.path.join(data\_dir, "CnonC\_dev.txt")), "dev")

def get\_test\_examples(self, data\_dir):

return self.\_create\_examples(

self.\_read\_tsv(os.path.join(data\_dir, "CnonC\_test.txt")), "test")

def get\_labels(self):

#return ["0", "1"]

return ["001-營建類", "002-非營建類"] # for CnonC dataset

def \_create\_examples(self, lines, set\_type):

"""Creates examples for the training and dev sets."""

examples = []

for (i, line) in enumerate(lines):

guid = "%s-%s" % (set\_type, i)

text\_a = tokenization.convert\_to\_unicode(**line[1]**)

label = tokenization.convert\_to\_unicode(**line[0]**)

examples.append(

InputExample(guid=guid, text\_a=text\_a, text\_b=None, label=label))

return examples

然後，大約在813行，修改成：

processors = {

"cola": ColaProcessor,

"mnli": MnliProcessor,

"mrpc": MrpcProcessor,

"xnli": XnliProcessor,

"**cnonc**": **CnonC\_Processor** # for my own dataset; key should be in lower case, due to task\_name = FLAGS.task\_name.lower() below

}

1. 設定相關參數，進行訓練、預測、評估的流程：
2. 在自己的電腦上啟動Python 3.6 環境：

$ conda activate py3.6

1. To run training:

$ python run\_classifier\_sam.py \

--task\_name=**cnonc** \

**--do\_train=true** \

--do\_eval=false \

--data\_dir=**../Datasets** \

--vocab\_file=chinese\_L-12\_H-768\_A-12/vocab.txt \

--bert\_config\_file=chinese\_L-12\_H-768\_A-12/bert\_config.json \

**--init\_checkpoint=chinese\_L-12\_H-768\_A-12/bert\_model.ckpt \**

--max\_seq\_length=128 \

--train\_batch\_size=32 \

--learning\_rate=2e-5 \

--num\_train\_epochs=10.0 \

--output\_dir=Model\_CnonC

*eval\_accuracy = 0.93*

*loss = 0.000105106905*

*It takes about 60 minutes.*

1. To run prediction:

$ python run\_classifier\_sam.py \

--task\_name=**cnonc** \

**--do\_predict=true** \

--data\_dir=../Datasets \

--vocab\_file=chinese\_L-12\_H-768\_A-12/vocab.txt \

--bert\_config\_file=chinese\_L-12\_H-768\_A-12/bert\_config.json \

**--init\_checkpoint=** **Model\_CnonC** \

--max\_seq\_length=128 \

--output\_dir=Output\_CnonC

*It takes about 2 minutes*

1. MicroF1至少0.91或0.93不等，這結果太驚人了，因為之前用傳統方式，最好的SVM結果約0.90或0.91。
2. **在Colab上執行：**

參考：<https://leowood.github.io/2018/11/16/BERT-fine-tuning-with-Google-GPU/>，步驟如下：

1. 確認上述Google的bert所有的程式碼、預訓練模型目錄、以及此處的程式都在Google雲端硬碟的同一目錄下：**/Users/sam/GoogleDrive/BERT/bert/**，其中sam為登入電腦的使用者名稱，可改成自己的名稱。
2. 在瀏覽器的Google雲端硬碟畫面，按右鍵點選run\_classifier\_colab.ipynb，然後點選「選擇開啟工具」，再選擇Google Colaboratory即可。  
   （若要新增一個colab檔案，請在瀏覽器的任意位置（不要在檔案或目錄上）按右鍵，顯示彈跳視窗選單，選取「更多」，選擇「Google Colaboratory」，會出現Jupyter的ipynb檔案畫面。這是一個空白檔案，可以將程式碼拷貝進去。）
3. 在命令列上選擇「修改」->「筆記本設置」，將「硬件加速器」設置為GPU。
4. 請參考run\_classifier\_colab.ipynb裡的說明，若資料準備妥當，放在正確的路徑，即可執行無誤。需搭配run\_classifier\_sam.py, eval\_clf.py程式。