文字探勘初論期末專題

獨領風騷地走在時代尖端—預測新興網路詞彙

第17組

李昀熹

資管二

B11705046@g.ntu.edu.tw

陳庭宇

資管二

B11705053@g.ntu.edu.tw

陳奕廷

資管二

B11705051@g.ntu.edu.tw

陳泊華

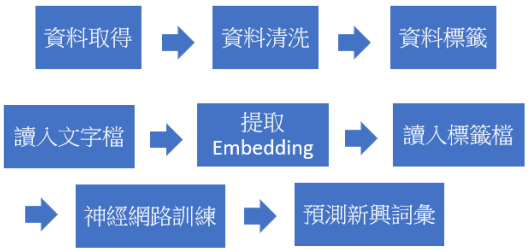
資管一

B12705014@g.ntu.edu.tw

動機與目的

身處當今網際網路世代，許多潮流在網上產生；我們好奇一個不同於傳統常用的日常對話用語的網路新興詞彙有何特色，藉由分析過往網路用語大多數被使用的語境判斷是否為網路用語，並以這些文章資料去做training，並期待訓練模型能在一篇輸入文章中找出潛在的新興網路詞彙，抑或發掘未來網路上可能產生的新興網路詞彙，更好地掌握潮流脈動。

解法架構



圖一 架構流程圖

上圖為針對新興詞彙的預測問題我們所設計的解法架構；以下將對各部分簡要闡述，後續章節會有詳細說明。

首先利用爬蟲程式來取得PTT八卦版上資料，接著將得到的文字檔做資料清洗，並且針對每份資料利用domain knowledge標記新興詞彙作為客製化訓練資料集。之後逐個讀入文字檔，使用BERT-Chinese_L-12_H-768_A-12提取文章中每個Token的Embedding，接著讀入標籤檔放入神經網路訓練，最後的模型可以用來預測一篇輸入文章中可能的新興詞彙有哪些。

資料獲取

我們認為新興網路詞彙較頻繁出現在年輕人常用的網站或評論，初步選定PTT、Dcard、FaceBook社團當作資料來源，後於實作網路爬蟲時發現PTT上的資料較容易取得且有一定討論度，於是選用PTT作為資料庫基礎；為了避免文章取向差距太大導致語意分散，進一步鎖定八卦版作為資料。

我們利用Python的網路爬蟲來取得我們所需資料，主要使用Selenium函式庫，用到的功能包括點擊頁面、回到上一頁以及獲取當前頁面所有的文字。大致上是，點一個標題進去->取得當前頁面的內文->回上一頁，如果當前頁面取完就換別頁。程式碼如下:

```
while(True):
    title_divs = driver.find_elements(By.CLASS_NAME, "title")
    metas = driver.find_elements(By.CLASS_NAME, "meta")
    for title_div, meta in zip(title_divs, metas):
        date_div = meta.find_elements(By.CLASS_NAME, "date")
        #print(date[0].text)
        i += 1
        print(str(title_div.text)[:8])
        if str(title_div.text)[:8] == "(本文已被刪除)" or str(title_div.text)[:3] == "(已被":
            continue


        date = date_div[0].text.replace("/", "_")
        path = f'text/{date}'
        # Check if the folder doesn't exist, then create it
        if not os.path.exists(path):
            os.mkdir(path)
            print(f"Folder '{path}' created successfully.")

        link = title_div.find_element(By.TAG_NAME, "a")
        href_value = link.get_attribute("href")
        #print("Link:", href_value)
        link.click()
        time.sleep(0.01)

        try:
            content = driver.find_element(By.XPATH, '//*[@id="main-content"]')
            file_path = f'text/{date}/{i}.txt'
            #print(content.text)
            with open(file_path, "w", encoding="utf-8") as file:
                file.write(content.text)
                print(f'Successfully store {i}.txt')
            # If the element is found, you can access it or perform other actions here
        except NoSuchElementException:
            print("content not found. Continuing with the script.")

        driver.back()
    click_last_page()
```

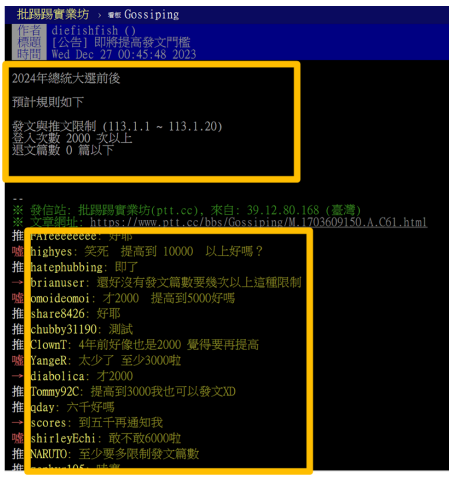
圖二 爬蟲程式碼

主要遇到的問題是，某些有標題卻沒有文章連結可以點擊，導致出錯，我們就篩選出「本文已被刪除」等等的標題就略過不取。

資料處理

文章處理和過濾

為了排除PTT本身版上的noise，我們僅選取橘色方框內的文字，又因為我們的模型僅能讀取512個字，因為BERT在超過512個tokens之後會進行壓縮，於是我們會淘汰那些大於512字的文章。



圖四 PTT頁面

雖然有些新興詞彙是英文或中英夾雜，但由於ptt八卦版上還是以中文為主，為了方便和統一性，我們僅觀察中文的新興詞彙。同時，我們過濾掉標點符號(stop words)、「推」、「噓」等共通的無意義字詞，減少模型雜訊。如圖四:

```
trainingDataSize = 180
stop_punc = [',','：','.','/','→','[',']','#','噓','推','發信站','文章網址','※',' ','?'] #self defined

#preprocessing/cleaning data
for i in range(1, trainingDataSize+1): #iterate all data
    with open ("./TrainingData2/"+str(i)+".txt",'r', encoding='utf-8') as data: #using the designated path
        text = data.read()
        #text = text.split('\n')
        text = re.sub(r'[a-zA-Z]', '', text)
        text = [x for x in text if x not in stop_punc]
        text = ' '.join(text)
        print("text", i, text)
        corpus.append(text)
        print("data",i,"read")
```

圖四 資料清洗程式碼

人工標籤

我們瀏覽PTT上爬到的文字內容，手動將文章中有出現的新興詞彙標記起來，並將文章標示為是否有新興詞彙，之後觀察程式碼預測出來的字是否符合我們當初人工標籤的那些新興詞彙。

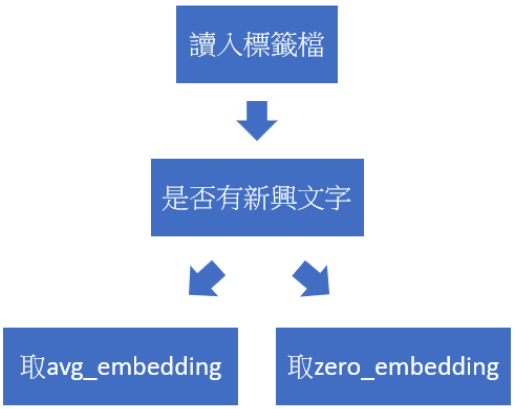
文件編號	分類(1/0)	啥詞		
example.txt	1	超派	確實	肥宅

表一 標籤格式

模型訓練

訓練邏輯

由於我們希望利用訓練資料的文章前後文所提供的資訊，試圖學習關鍵字（新興詞彙）和其前後文語意間的關係，並且可以學習到一字多義的語意（擁有不同embedding），因此我們選用BERT-Chinese_L-12_H-768_A-12預訓練模型作為生成embeddings的依據，將讀入文章的CLS embedding作為訓練特徵（X_train），接著根據讀入的標籤檔（y_train）資料判斷該篇文章是否含有關鍵字，若該資料中含有關鍵字便將關鍵字的embedding取平均，若無則取zero embedding，代表該語意下並沒有產生關鍵字（如圖六說明）。最後便可以得到一個input為文章的CLS embedding，output為對應到模型預測代表「新興詞彙」的embedding。意即當我們輸入一篇文章時，預期輸出代表的意義是對輸入文章可能的新興詞彙的embedding預測，而非此篇文章有「有」或「無」新興詞彙的標籤；至於如何利用此預測的embedding會於「預測方式」章節進一步說明。下圖是兩個文章的輸出：



圖五 X_train與y_train提取示意圖

```
print(predictions)

[[ 0.7962642  -0.04244137 -0.85815156 ...  0.6589758  -0.2806137
  -0.68983144]
 [ 0.59862083 -0.01777782 -0.870049   ...  0.7350842  -0.17350526
  -0.7062705 ]]
```

圖六 模型預測說明

程式函式說明

find_index(tokens, target): target是我們標記的新興詞彙，tokens則是在文章tokenize後的結果，我們使用函式的目的是為了取得token裡面的哪個index是我們目標的新興詞彙，以利後續取得avg_embedding當模型訓練時的y_train。

avg_embedding(emb): emb是存有跟target長度一樣的list of embeddings，方式是把emb全部加起來做平均，得出的結果即為我們認定的新興詞彙的embedding長相（可能的前後文機率分佈樣態）。

模型說明

為了能夠在有限的資料集內盡可能學習文本的語意，我們採用神經網路進行模型訓練。首先針對訓練資料進行train test split，接著將training set文章的CLS embedding配對關鍵字（新興詞彙）的embedding，丟入兩層的神經網路進行training，學習資料集文章的語意。

預測方式

承上述訓練邏輯，模型預測方式為讀入一筆測試文章資料，利用BERT Chinese 產生其CLS embedding（代表整篇文章的語意）後放入模型進行預測。由於訓練的時候是以文章CLS對應新興詞彙的embedding，所以我們預期得到的結果就會是這篇文章的新興詞彙embedding大致長相，我們認為只要差距不要差太多就可能是新興詞彙，而差距的定義為文章每個字的embedding跟預測結果embedding的Mean Squared Error，只要數值小於某個給定的門檻，便認定該字為新興詞彙，目前以MSE低於0.6當取用的標準。

另外，當找到某個符合我們門檻的字時，或許左右兩邊字的embedding

也相當接門檻但卻沒有被取到，我們也會把它加入我們的預測中，畢竟左右兩個字是有機會是跟找到的字搭配的。而接近的程度我們也會設一個門檻，目前設定為0.1，因此預測的結果可能會是一個到三個字的詞。

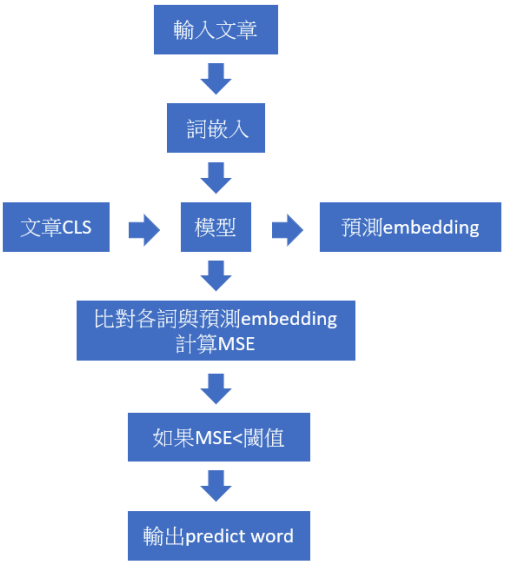
我們在下圖七中由左至右依序是，token的index、字、MSE。例如第23及24低於了0.6就採用，而左右兩個相鄰的字若差不到0.1也納入考量，因此輸出會像圖八。

```
21 認 0.7271926
22 識 0.6831098
23 帥 0.5883491
24 甲 0.59028244
25 甲 0.6168773
```

圖七 文章輸出

```
'識帥甲', '帥甲甲', '甲甲'
```

圖八 輸出



圖九 模型預測流程圖

訓練結果

我們沒有一個普遍效能評估的方式，僅是運用肉眼比對預期的字與預測出來字差別多大。其中我們有發現很多特點，例如給入的文章的可讀性能大大提升預測的機率跟純度等等。

```
'排車同', '軌書同', '同文', '國女嫪',
```

這張圖某個資料的預測字，看起來沒有很合理的字詞出現，但如果我們觀察文章的時候可以發現，其實人也看不懂，如圖下的留言。

車同軌書同文

因此我們特別挑了文字邏輯好的文章來預測，可以發現效能的確提升很多，甚至還有好幾篇預測出來是空的，符合我們認為沒有新興詞彙的結果。

```
'=偶滴', '滴媽', '人他喵', '有瓜嗎', '啊[SEP]',
```

此輸出大致上是:
偶滴媽、他喵、有瓜嗎(有卦嗎)

這篇文章是測的很準，畢竟沒有noise的出現，而部分文章雖有預測到我們需要的字，但同時也有輸出其他noise的字詞，對我們來說表現就略顯普通。

```
'批上', '有萌新', '房變這',
```

我們認為這篇的新興字是「萌新」，但「一批一批上」的「批上」也被納入。

此外，我們還有很多效能很不錯的預測結果，如圖下。

```
[CLS]安安', '大給賀', '給賀阿', '賀阿!', '!是降', '是降的', '降的啦',
```

預測結果:
安安大給賀阿!是降的啦
(安安大家好啊!是這樣的啦)
這裡是原文的開頭語，觀察預測的結果，發現我們也能預測一連串的字，另外在同一篇文章中，他也有預測出「卡稱」，符合了我們的預期。

最後，假使預測沒有新興詞彙，則僅輸出[CLS]，如圖下。

```
'[CLS]'
```

模型表現

我們透過比較特定位置(index)的嵌入向量預測值和相鄰位置的預測值的均方誤差(MSE)來評估這些嵌入向量

之間的相似性，當特定位置的嵌入向量的預測和相鄰位置的預測相似時，這代表這些位置對應的語義信息相近或相關，MSE可以幫助我們在文本中找到並合併具有相似語義的標記，進而改進模型的輸出或生成更具意義的文本序列。

```
Epoch 1/10
3/3 [==============================] - 1s 73ms/step - loss: 0.5662 - val_loss: 0.4170
Epoch 2/10
3/3 [==============================] - 0s 13ms/step - loss: 0.3732 - val_loss: 0.3303
Epoch 3/10
3/3 [==============================] - 0s 13ms/step - loss: 0.2999 - val_loss: 0.2641
Epoch 4/10
3/3 [==============================] - 0s 14ms/step - loss: 0.2439 - val_loss: 0.2248
Epoch 5/10
3/3 [==============================] - 0s 13ms/step - loss: 0.2125 - val_loss: 0.2065
Epoch 6/10
3/3 [==============================] - 0s 14ms/step - loss: 0.1959 - val_loss: 0.1979
Epoch 7/10
3/3 [==============================] - 0s 14ms/step - loss: 0.1853 - val_loss: 0.1879
Epoch 8/10
3/3 [==============================] - 0s 14ms/step - loss: 0.1750 - val_loss: 0.1801
Epoch 9/10
3/3 [==============================] - 0s 14ms/step - loss: 0.1646 - val_loss: 0.1732
Epoch 10/10
3/3 [==============================] - 0s 14ms/step - loss: 0.1555 - val_loss: 0.1673
1/1 [==============================] - 0s 36ms/step
Mean Squared Error: 0.16727345
```

檢討改進

這次專題中我們嘗試設計了一套邏輯來發現仍有許多可以改善的空間，讓我們的模型預測能過濾雜訊，變得更加準確：

1. 人工標籤有瑕疵的地方在於標記的標準主要取決於主觀想法，加上新興詞彙本身就沒有甚麼客觀的評斷標準，導致整個結果是相對模糊的。可以改進的地方是我們可以定義出屬於我們自己一套的評量標準，統一地去幫資料做標籤。
2. 有鑑於我們的文字都是一個一個做切割，使得詞和詞之間的連結性不高，或許可以嘗試用**Jieba**做**Tokenization**，雖然新興詞彙大多和正常用法不同，但尚能將文章作相對適合的切割。
3. 我們發現目前選用Bert的模型tokenization長度最高為512，若選用其他模型，或許能基於更大的輸入文章進行訓練，提高模型預測能力。
4. 在這次專題中我們是透過觀察輸入文章的embedding，調整MSE誤差容許的閾值；未來或許可以導入更精細的統計概念，系統化調整參數，讓模型可以在雜訊和關鍵字輸出間取得良好平衡。

參考資料

林昱瑋，”多來源之網路輿論關鍵字分析之研究，“ 國立台中科技大學資訊工程學系碩士論文，2022
https://hdl.handle.net/11296/693fc5

郭豪育，“以長短期記憶模型為基礎的語意理解及關鍵字生成，” 義守大學資訊工程學系學術論文，2023
https://hdl.handle.net/11296/757262