

2nd-Python Crawler期末專題(Vincent Wu)

2020/04/27

一、專題摘要 (解釋實作與說明需要解決的問題，限300~500字。)

1. 期末專題主題

爬取Cupoy熱門文章[https://www.cupoy.com/newsfeed/
topstory](https://www.cupoy.com/newsfeed/topstory)中的任一主題前 500 篇文章。

2. 期末專題基本目標

- (1) 學習如何使用selenium抓取網頁上資料，如標頭、網頁連結和分類等等。
- (2) 學習如何抓取網頁連結之內容，這裡使用newspaper。
- (3) 分析網頁分類之分佈，這裡使用圓餅圖。
- (4) 利用jieba來分析網頁詞句之分佈。
- (5) 利用TFIDF來分析網頁詞句之分佈。
- (6) 利用wordcloud來具體表現網頁詞句之分佈。

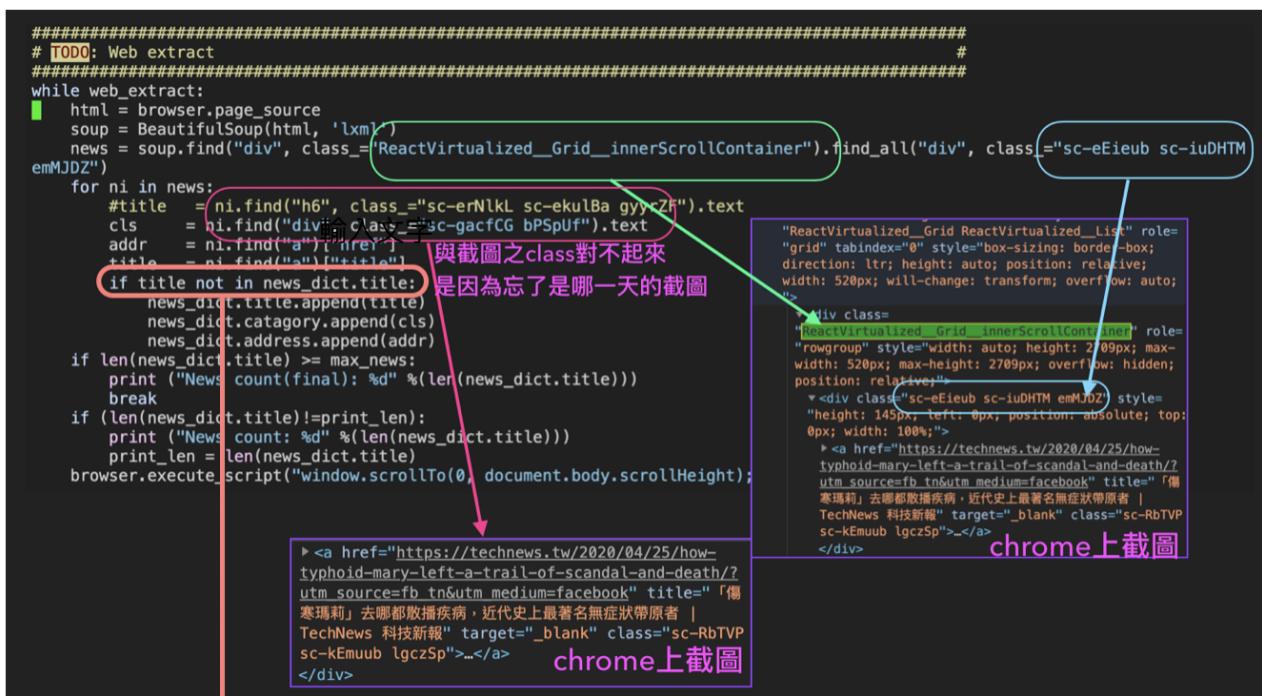
二、實作方法介紹 (介紹使用的程式碼、模組，並附上實作過程與結果的截圖，需圖文並茂。)

1. 使用的程式碼介紹

(1) 使用selenium抓取網頁上資料

```
if web_extract:  
    browser = webdriver.Chrome(executable_path='/Users/vincentwu/Documents/GitHub/2nd-PyCrawlerMarathon/homework/driver/chromedriver')  
    browser.get(cupoy_url)  
    time.sleep(15)
```

⇒`time.sleep(15)`→電腦連網頁較慢，需等待約15秒，後續的`soup.find`才不致於找回None之物件



```
if len(news_dict.title) > max_news:          Web extraction finish  
    print ("News count before slice: %d" %(len(news_dict.title)))  
    news_dict.title      = news_dict.title[:max_news]  
    news_dict.catagory  = news_dict.catagory[:max_news]  
    news_dict.address   = news_dict.address[:max_news]  
    print ("News count after slice: %d" %(len(news_dict.title)))
```

⇒其中橘線主要是用來判斷是否為同一主題，此方法可以讓我不用使用`time.sleep`，因為相同的`page_source`會讓我繼續，省時間。結束時，因為會取超過之網頁所以會先`slice`到需要之新聞數目。

→其中web_extract之參數，主要控制是否要爬文；若無則取所存之binary 檔案proj_data_web.pkl

2. 使用newspaper來取網頁連結之內文

3. 使用圓餅圖分析文章屬性

APP應用程式
12.1%
9.1%
9.1%
6.1%
3.0%
3.0%
9.1%
9.1%
9.1%
9.1%
Fliper Net
Engadget
BuzzOrange
蘋果
AI與大數據
Fintech金融科創
Google
AppWorks
20道品

```
# TODO: Pie table plot by matplotlib
# ref: https://blog.csdn.net/gmr2453929471/article/details/78655834
# ref: https://codertw.com/程式語言/359974/
# ref: https://www.itread01.com/content/1545238867.html
# ref: https://ithelp.ithome.com.tw/articles/10196410
# ref: https://matplotlib.org/3.1.1/gallery/pie_and_polar_charts/pie_and_donut_labels.html#sphx-glr-gallery-pie-and-polar-charts-pie-and-donut-labels-py
#####
news_pd = pd.DataFrame({"title": news_dict.title, "catagory": news_dict.catagory})
news_gp = news_pd.groupby("catagory")
news_count_dict = news_gp.count().to_dict()["title"]
max_pie_out = 10
max_explode = 3
news_cls_list = list(news_count_dict.keys())[:max_pie_out]
news_count_list = list(news_count_dict.values())[:max_pie_out]
ii = np.argsort(news_count_list)[-1:-max_pie_out-1:-1]
news_cls_arr = np.array(news_cls_list)[ii]
news_count_arr = np.array(news_count_list)[ii]
pie_explode = np.zeros_like(ii)
pie_explode[:max_explode] = 0.0
from matplotlib.font_manager import FontProperties
font=FontProperties(fname='/Users/vincentwu/Downloads/simheittf/simhei.ttf',size=10)
#ax1.set_xticklabels(ability_labels,fontproperties=font)
patches, l_text, p_text = plt.pie(news_count_arr, labels=news_cls_arr, autopct="%1.1f%",explode=pie_explode, radius=1)
for li in l_text:
    li.set_fontproperties(font)
plt.axis("equal")
plt.savefig("proj_wordcloud.png")
print ("[Pie chart]:\n\t!t save png to proj_pie.png")
```

4. 使用jieba分析熱門文章內文之詞句分佈

```
#####
# TODO: jieba
# ref: https://github.com/tomlinNTUB/Python-in-5-days/blob/master/10-2%20中文斷詞-移除停用詞.md #
#####
stop_words = []
file_stop_words = "jieba_stop_words.txt"
with open(file_stop_words, 'r', encoding='UTF-8') as file:
    for data in file.readlines():
        data = data.strip()
        stop_words.append(data)
##title_all = [ti+"。" for ti in news_dict.title]
##title_all = reduce(op.add, title_all)
words_all = []
words_all2 = []
print ("[Cut sentence by Jieba]")
for ti in tqdm(news_dict.article):
    if len(ti) <= 0: continue
    segments = jieba.cut(ti, cut_all=False)
    remainderWords = list(filter(lambda a: a not in stop_words and a != '\n', segments))
    words_all2 += remainderWords
    remainderWords = [wi.strip() + " " for wi in remainderWords]
    words_all += [reduce(op.add, remainderWords)]
words_pd = pd.DataFrame(words_all2, columns=["word"])
words_gp = pd.DataFrame(words_pd.groupby("word").size(), columns=["cumsum"])
words_gp = words_gp.sort_values(by="cumsum", ascending=False)

# 使用此網站之詞作stop words
# jieba之結果by terminal截圖
# 收集stop words
# 使用jieba分析哪個詞用作多
```

5. 使用TFIDF分析熱門文章內文之詞句分佈

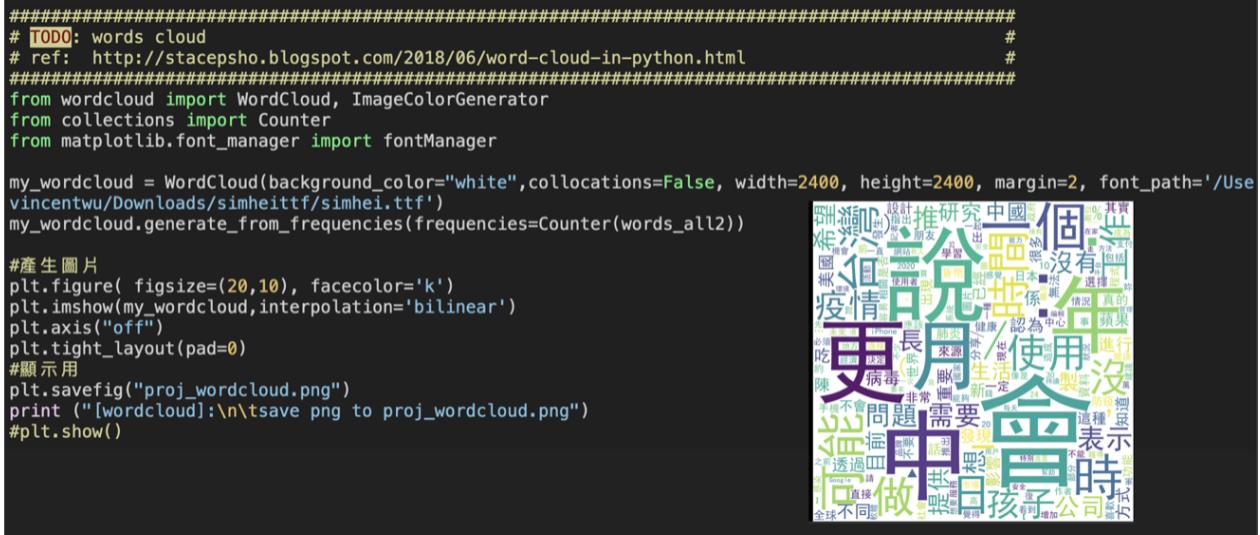
```
#####
# TODO: TFIDF
# ref: https://blog.csdn.net/Eastmount/article/details/50323063
# ref: https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfTransformer.html#sklearn.feature_extraction.text.TfidfTransformer
#####
from sklearn import feature_extraction
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.pipeline import Pipeline
pipe = Pipeline([('count', CountVectorizer(vocabulary=stop_words)), ('tfid', TfidfTransformer())]).fit(news_dict.article)
ff = open("proj_tfidf.txt", "w")
ii = np.argsort(pipe['tfid'].idf_)
for iii in ii:
    ff.write("%s: %f\n" %(stop_words[iii], pipe['tfid'].idf_[iii]))
ff.close()
import pdb; pdb.set_trace()
print ("[TFIDF]:\n\tsave to file (proj_tfidf.txt) !")

Inverse document frequency [edit]
The inverse document frequency is a measure of how much information the word provides, i.e., if it's common or rare across all documents. It is the logarithmically scaled inverse fraction of the documents that contain the word (obtained by dividing the total number of documents by the number of documents containing the term, and then taking the logarithm of that quotient):

$$idf(t, D) = \log \frac{N}{|\{d \in D : t \in d\}|}$$

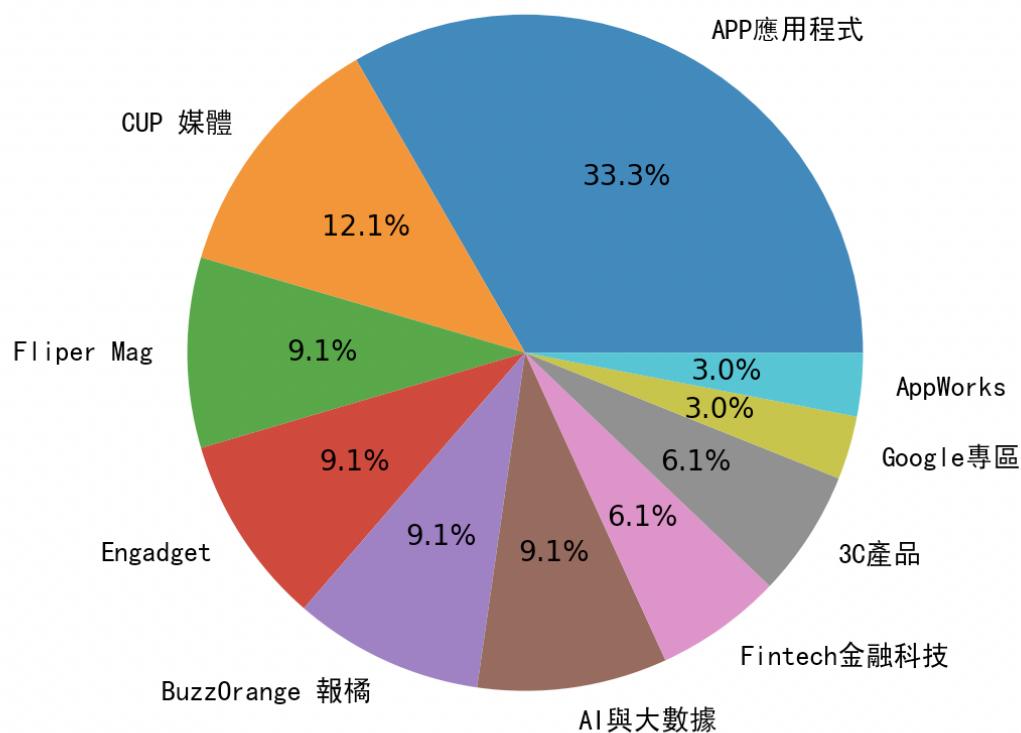
IDF ↴ , frequency ↗
# 輸出結果
```

6. 使用wordcloud具體表現文章詞句使用量



三、成果展示 (介紹成果的特點為何，並撰寫心得。)

(1) 文章分類圓餅圖



圖表現得感覺不多精緻，但大致可以表現出屬性最多的是APP應用程式，佔1 / 3。

(2) 熱門文章詞句使用量分析，jieba和TFIDF

```
Jieba  
(Pdb) words_gp.iloc[:100].to_dict()  
{'cumsum': {'': 14621, '會': 1241, '說': 945, '中': 832, '': 740, '更': 727, '。': 699, '是': 689, '年': 659, '月': 648, '一個': 648, '': 578, '可能': 577, '時': 571, '': 533, '台灣': 446, '日': 445, '時間': 444, '做': 431, '使用': 417, '沒': 416, '': 406, '孩子': 393, '工作': 387, '疫情': 383, '表示': 364, '推': 360, '需要': 355, '問題': 348, '想': 339, '長': 334, '': 333, '公司': 333, '沒有': 317, '希望': 302, '': 300, '('': 294, '提供': 293, '研究': 292, '中國': 281, '目前': 280, '生活': 272, '%': 269, '製': 269, '不同': 262, '係': 252, '吃': 250, '病': 249, '進行': 249, '認為': 244, '透過': 243, '美國': 241, '重要': 240, '/': 237, '方式': 234, '發現': 226, '影響': 225, '新': 224, '陳': 221, '蘋果': 220, '知道': 218, '已經': 217, '這種': 216, '不會': 215, '話': 214, '很多': 213, '出現': 209, '世界': 207, '設計': 207, '真的': 207, '非常': 206, '其實': 205, '資料': 201, '來源': 192, '肺炎': 190, '日本': 186, '無法': 186, '出': 184, '一定': 182, '分享': 182, '全球': 181, '選擇': 180, '不要': 178, '功能': 176, '是否': 176, '健康': 175, '防疫': 175, '▲': 175, '發生': 174, '貨幣': 172, '朋友': 170, '指出': 169, '是': 167, '中心': 164, '學習': 164, '★': 164, '程式': 163, '圖片': 163, '現在': 163, '手機': 163, '直接': 163, '相關': 162}}
```

TFIDF 輸出結果

| 1 proj_tfidf.txt |
|------------------|
| 不過: 3.1716 |
| 此外: 3.2443 |
| 因此: 3.2443 |
| 另外: 3.2634 |
| 然而: 3.6311 |
| 例如: 3.8134 |
| 所以: 4.0366 |
| 當然: 4.1701 |
| 但是: 4.2189 |
| 同時: 4.2189 |
| 首先: 4.5066 |
| 或是: 4.6497 |
| 除此之外: 4.6497 |
| 以及: 4.7297 |
| 根據: 4.7297 |
| 另一方面: 4.7297 |
| 其中: 4.8167 |
| 換句話說: 4.9120 |
| 於是: 4.9120 |
| 而且: 5.0174 |
| 等等: 5.1352 |
| 或者: 5.1352 |
| 因為: 5.2687 |
| 比如: 5.2687 |
| 只是: 5.2687 |
| 雖然: 5.2687 |

兩個stop_words的詞句不同，jieba是另外下載帶入的，但TFIDF應該是做用內建的，所以結果不盡相同。

(3) wordcloud



四、結論 (總結本次專題的問題與結果)

這次發現stop words的選擇上要再挑選，wordcloud的表現偏較單字。還有在Jieba斷詞上還有mac中文表現的能力上還需再多作一點研究，至於TFIDF的使用上也是常常碰壁。程式方面可能可以多使用function會較簡潔。

這次也因為自己的小失誤打錯 (art.test(X)應該是art.text(O))，而多學會一個newspaper的方法作網頁內文的整理，也許一開始的 500 熱門文章資訊收集上，也許可以試試用newspaper來試著作作看。

感謝每次的專題報告都讓我獲益良多～這次的進階題目學到很多。謝謝cupoy團隊～

五、期末專題作者資訊 (請附上作者資訊)

1. 個人Github連結

https://github.com/double1010x2/2nd-PyCrawlerMarathon/blob/master/homework/project_final.py