IRTM HW2 Report

R06725035 資管碩二 陳廷易

執行環境

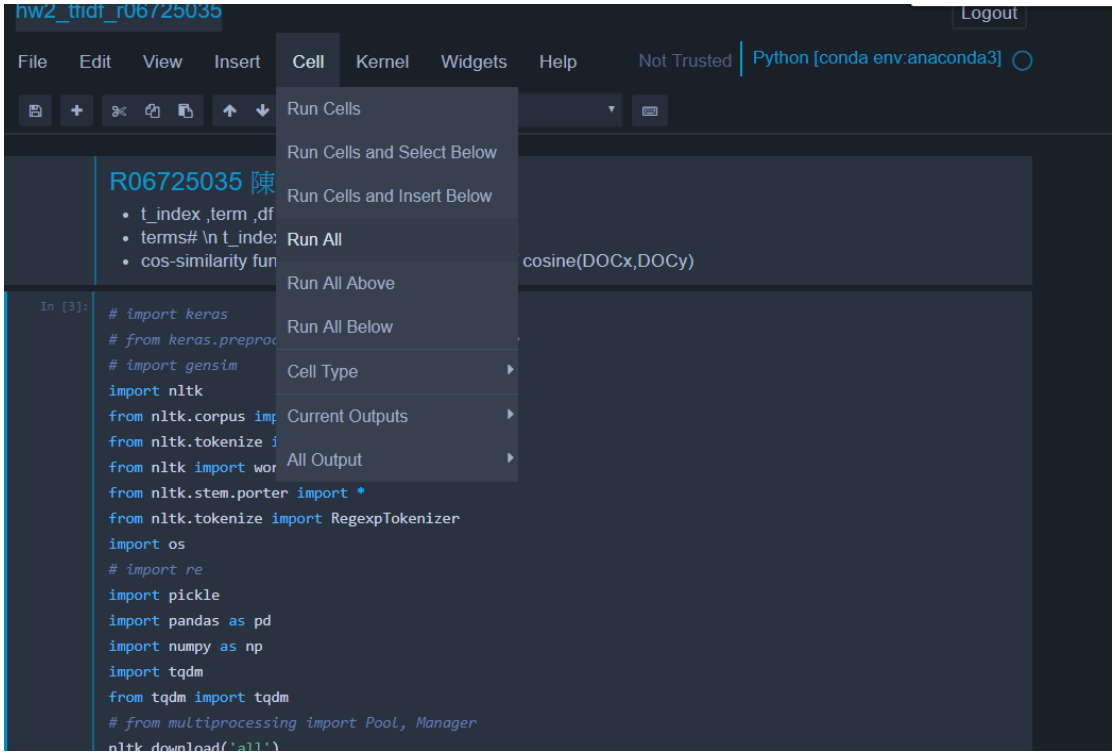
- Ubuntu 16.04
- Jupyter Notebook

程式語言

- Linux Anaconda 5 Python 3.6

執行方式

- 提供 jupyter notebook 版本以及一般 python 版: *hw2_tfidf_r06725035.ipynb*
  - 可利用 jupyter 開啟 ipynb 以後 Cell=>Run All



  - 或可利用 python3 hw2_tfidf_r06725035.py 直接執行 python 檔案

- 需 pip install:
  - nltk 並 download data
  - pickle、pandas、numpy
  - tqdm 顯示當前進度
- 確保提供的 1095 篇 txt 預設放於 data/IRTM/目錄下
- data/目錄下需要有 stop_words.txt 及 stop_list.pkl 來製作 stop words list
- dictionary 產出的結果預設放於 output/目錄下: dictionary.txt
- 各文章的 tf-idf 將會依照 data/IRTM/目錄下的檔案名稱輸出至 output/tf-idf/目錄下: X.txt

作業邏輯說明

1. 讀入所有不需要的 stop words 以及標點符號等等，並利用 nltk 套件初始化 porterstemmer

stop words and stemming prepare:

```
with open('data/stop_words.txt') as f:
    stop_words_list = f.read().splitlines() #stop_list1
stop_list2 = pickle.load(open('data/stop_list2.pkl','rb'))
ps = PorterStemmer() # Stemming
stop_words = set(stopwords.words('english')) #Stopword
short = ['.', ',', '"', "\'", '?', '!', ':', ';', '(', ')', '[', ']', '{', '}', "'at",
         "_","`","\'\'","--","``",".,","//",":","___",'_the','-',"'em",".com",
                  '\'s','\'m','\'re','\'ll','\'d','n\'t','shan\'t',"...","\'ve",'u']
stop_words_list.extend(short)
stop_words_list.extend(stop_list2)
stop_words.update(stop_words_list) # remove it if you need punctuation
```

2. 定義前處理函式：將文件轉換為小寫、濾除 stop words、stemming、進行 tokenize、濾除數字

```
def preprocess(texts):
    tokens = [i for i in word_tokenize(texts.lower()) if i not in stop_words]  # Tokenization.# Lowercasi
    token_result = ''
    token_result_ = ''
    for i,token in enumerate(tokens): #list2str
        token_result += ps.stem(token) + ' '
    token_result = ''.join([i for i in token_result if not i.isdigit()])
    token_result = [i for i in word_tokenize(token_result) if i not in stop_words]
    for i,token in enumerate(token_result):
        token_result_ += token + ' '
    return token_result_
```

3. 將所有文件讀入並依照上述函式進行前處理

crawl all documents cut out tokens

```
tokens_all = ""
for file in next(os.walk('data/IRTM/'))[2]:
    f = open('data/IRTM/'+file)
    texts = f.read()
    f.close()
    tokens_all += preprocess(texts)
```

4. 進一步移除過短的字詞(長度小於三的 token，因認為太短的字沒辦法代表意義)

remove too short and print

```
token_set = set(tokens_all.split(' '))
token_list = list(token_set)
for token in token_list: #13551=>13406
    if len(token)<3:
        token_list.remove(token)
token_list = sorted(token_list)
print(len(token_list))
token_list
```

```
13191

['abahd',
 'abandon',
 'abat',
 'abc',
 'abc.com',
 'abcnews.com',
 'abdallah',
 'abdel',
 'abdomin',
 'abduct'
```

5. 先進行第一次計算 df，為了看各 token 在此 dataset 的出現情況

```
df = pd.DataFrame(pd.Series(token_list),columns=['term'])
df['t_index'] = df.index+1
df['df'] = 0
df = df[['t_index','term','df']]
```

```
for file in tqdm(next(os.walk('data/IRTM/'))[2]):
    f = open('data/IRTM/'+file)
    texts = f.read()
    f.close()
    tokens_all = preprocess(texts)
    for term in token_list:
        if term in tokens_all:
            df.loc[df['term']==term,'df'] += 1 # build the dict
#     print(df.loc[df['term']==term,'df'])
df
```

13176	13177	zimmerman	1
13177	13178	zinc	1
13178	13179	zingic	1
13179	13180	zinni	18
13180	13181	zivko	2
13181	13182	zivkov	2
13182	13183	zogbi	3
13183	13184	zone	28
13184	13185	zoran	20
13185	13186	zorica	1
13186	13187	zubin	1
13187	13188	zuric	2
13188	13189	zurich	1
13189	13190	zutshi	1
13190	13191	zvezda	1

13191 rows × 3 columns

6. 為了進一步降低 token 數量讓分類可以更好且增加效能，濾除 token 出現過少次(一次，因難以分類)，以及出現過多次者(>85%文章都有，超過 950 次)，並將此 dictionary 儲存下來

```
df2 = df.drop(df[df.df<2].index)
df2 = df2.drop(df2[df2.df>950].index)
df2.reset_index(drop=True,inplace=True)
df2['t_index'] = df2.index + 1
df2  # 最後的terms dict
```

8897	8898	zarko	6
8898	8899	zawahri	5
8899	8900	zeal	3
8900	8901	zedek	2
8901	8902	zein	4
8902	8903	zell	3
8903	8904	zemin	7
8904	8905	zero	6
8905	8906	zinni	18
8906	8907	zivko	2
8907	8908	zivkov	2
8908	8909	zogbi	3
8909	8910	zone	28
8910	8911	zoran	20
8911	8912	zuric	2

8912 rows × 3 columns

```
df2.to_csv('output/dictionary1.txt',index=False,header=False,sep=' ')
```

7. 計算各篇文章於 dictionary 中的 token 之 tf-idf：

甲、先將文章讀入後進行前處理

乙、將 token 取出後，對照該 term 是否於 dictionary 中，若是則計數+1

丙、將各 term 出現次數除以所有 term 於該文章總出現次數再乘上所有文件數除以該 term 的 df 之 log，並取直飛 NaN 者

丁、接下來計算該文章所有有用到的 term 數目寫入 tf-idf txt 第一列

count tf-idf

```
tf_list = list(df2.term)
for file in tqdm(next(os.walk('data/IRTM/'))[2]):
    df_tf = df2[['t_index','term']]
    df_tf['tf'] = 0
#     df_tf
    f = open('data/IRTM/'+file)
    texts = f.read()
    f.close()
    tokens_all = preprocess(texts)
    for token in tokens_all.split(' '):
        if token in tf_list:
            df_tf.loc[df_tf['term']==token,'tf'] += 1
    all_term_tf = sum(df_tf.tf)
    df_tf.tf = (df_tf.tf / all_term_tf)* np.log10(1095 / df2.df)
    df_tf = df_tf[df_tf.tf > 0]
    df_tf.drop('term',axis=1,inplace=True)
    df_tf.to_csv('output/tf-idf/'+file, index=False, header=False, sep=' ')
    with open('output/tf-idf/'+file,'r') as ori: data = ori.read()
    with open('output/tf-idf/'+file, 'w') as mod: mod.write(str(len(df_tf))+'\n'+data)
```

```
df_tf
```

6903	6904	0.039898
6956	6957	0.057906
6957	6958	0.032242
6985	6986	0.033653
6999	7000	0.024448
7177	7178	0.029927
7346	7347	0.015636
7358	7359	0.037791
7404	7405	0.042614

8. 寫一個 cosine similarity function：

甲、將兩篇文章讀入後，丟棄第一列，並利用 t_index 將兩篇文章的 tfidf 值合併，若該文章無該 term 則填 0

乙、在分子部分，將兩篇文章的 tf-idf 相乘後做總和

丙、在分母部分，將兩篇文章的 tf-idf 計算平方和後開根號再相乘

丁、將分子除以分母便可得到 cosine similarity

以第一篇及第二篇文章為例可得 cosine similarity 為 0.23

```python
def cosine(DOCx, DOCy):
    '''
    input: doc1 path name(str) , doc2 path name(str)
    ouput: two doc's cosine similarity
    '''
    dfx = pd.read_csv(DOCx,sep=' ',names=['tindex','tfidf'],header=None)
    dfy = pd.read_csv(DOCy,sep=' ',names=['tindex','tfidf'],header=None)
    dfx = dfx.drop(0)
    dfy = dfy.drop(0)
    dfxy = pd.merge(dfx,dfy,on='tindex',how='outer')
    dfxy.fillna(0,inplace=True)
    up = sum(dfxy.tfidf_x * dfxy.tfidf_y)
    down = np.sqrt(sum(np.square(dfxy.tfidf_x)))* np.sqrt(sum(np.square(dfxy.tfidf_y)))
    result = up / down
    return result
```

```python
cos = cosine('output/tf-idf/1.txt','output/tf-idf/2.txt')
cos #0.25903264882120364
```

```
0.2307354688692533
```