In [ ]:
```
# encoding=utf-8
```

高等資料探勘與巨量資料分析 作業一：文字共現關聯分析 (Keyword Correlation Analysis)

Download wiki data (50,000 articles json file)

In [ ]:
```
import os
wikifilename='wiki_2021_10_05_50000.json'
download_enable = not(os.path.isfile(wikifilename))
```

In [ ]:
```
#untokenize wiki_data wiki_2021_10_05_50000
if(download_enable):
    !gdown --id 1rQnbaOiqoN40AzHVq_IrRW4ki-rFPRxZ
```

使用Jieba斷詞

下載Jieba繁體中文詞庫

In [ ]:
```
dictfilename='dict.txt.big'
download_enable_1 = not(os.path.isfile(dictfilename))
```

In [ ]:
```
# download 繁體中文詞庫
if(download_enable):
    !wget --directory-prefix='C:\Users\ken\pycode' 'https://www.dropbox.com/s/ikv3n0
    # 修改繁體中文 dict file name
    !mv /content/dict.txt.big?dl=1 /content/dict.txt.big
    # 看一下dict內容
    # !head /content/dict.txt.big
```

import library

In [ ]:
```
import jieba
import jieba.posseg as pseg
import paddle
from tqdm import tqdm #顯示進度條
import json
```

In [ ]:
```
# jieba configration
jieba.set_dictionary('dict.txt.big') # 設定使用繁體中文字典
jieba.case_sensitive = True # 可控制對於詞彙中的英文部分是否為case sensitive, 預設False
paddle.enable_static()
jieba.enable_paddle() #启动paddle模式, jieba 0.4版後支持
```

```
Paddle enabled successfully......
```

In [ ]:
```
user_dict_method_1 = False
user_dict_method_2 = False
user_dict_method_3 = True
```

In [ ]:
```
# method 1: 將所有的title做成名詞斷詞庫

if(user_dict_method_1):
    with open('userdict.txt', 'a+', encoding='utf-8') as f:
        for element in tqdm(data):
            f.write(element.get('title')+' n 3\n')
    f.close
    user_dist = 'userdict.txt'
    jieba.load_userdict(user_dist) #導入自訂詞庫
```

In [ ]:
```
# method 2: 將所有的keywords做成名詞斷詞庫
keywords = ['臺灣', '美國', '大學', '肺炎','天安門','歌手','中國','蔡英文','立法院','颱風
if(user_dict_method_2):
    with open('userdict.txt', 'w', encoding='utf-8') as f:
        for word in tqdm(keywords):
            f.write(word+' n 3\n')
    f.close
    user_dist = 'userdict.txt'
    jieba.load_userdict(user_dist) #導入自訂詞庫
```

In [ ]:
```
# method 3: use download wiki dict
if(user_dict_method_3):
    user_dist = 'wiki.dict.txt'
    jieba.load_userdict(user_dist) #導入自訂詞庫
```

```
Building prefix dict from c:\Users\ken\Documents\NCHU_Bigdata\HW_KCM\dict.txt.big
...
Loading model from cache C:\Users\ken\AppData\Local\Temp\jieba.u898abf08e0df186584d9
d0613cb0b4a9.cache
Loading model cost 1.716 seconds.
Prefix dict has been built successfully.
```

In [ ]:
```
# exapmle for 斷詞 by Jieba-tw
sentence = ['新竹的交通大學在新竹的大學路上','我愛北京天安門','台灣大學在美國與中國都很有名

for i in range(0, 3):
  words = jieba.cut(sentence[i])
  for word in words:
    print(word, end='|')
  print()

  words1 = pseg.cut(sentence[i]) # 斷詞+詞性
  for word, flag in words1:
    print(word+'('+flag+')', end='|')
  print()
  print('---------------------------------')
```

```
新竹|的|交通大學|在|新竹|的|大學路|上|
新竹(n)|的(n)|交通大學(n)|在(p)|新竹(n)|的(n)|大學路(n)|上(f)|
---------------------------------
我愛北京天安門|
我愛北京天安門(n)|
---------------------------------
台灣|大學|在|美國|與|中國|都|很|有名|,| |其|中有|一堂課|在|介紹|中國|天安門|事變|的|歷史|,
| |引發|蔡英文|總統|發表|看法|,| |並且|在|立法院|中|造成|言論|的|颶風|,|就算|在|新冠|肺炎|
的|影響|仍|具有|高度|關注度|
台灣(ns)|大學(n)|在(p)|美國(n)|與(zg)|中國(n)|都(d)|很(d)|有名(a)|,(x)| (x)|其(r)|中有
(n)|一堂課(x)|在(p)|介紹(x)|中國(n)|天安門(n)|事變(x)|的(n)|歷史(n)|,(x)| (x)|引發(x)|蔡
英文(n)|總統(n)|發表(x)|看法(v)|,(x)| (x)|並且(x)|在(p)|立法院(n)|中(n)|造成(v)|言論(x)|
的(n)|颶風(n)|,(x)|就算(v)|在(p)|新冠(n)|肺炎(n)|的(n)|影響(x)|仍(zg)|具有(v)|高度(n)|關
```

```
注度(x)|
--------------------------------
```

loading data

In [ ]:
```python
#data have been tokenized
with open('wiki_2021_10_05_50000.json', 'r', encoding='utf-8') as file:
        data = json.load(file)
```

In [ ]:
```python
# 確認一下json file內容
print('---'*10)
print('json file load to data')
print(len(data))
print(type(data))
print('---'*10)
print('the element of data[0]')
print(len(data[0]))
print(type(data[0]))
print('---'*10)
print('review dict material')
print('1. element of key')
print(data[0].keys())
print('2. get value by key \'articles\'')
print(data[0].get('articles'))
print(data[1].get('articles'))
```

```
------------------------------
json file load to data
50000
<class 'list'>
------------------------------
the element of data[0]
3
<class 'dict'>
------------------------------
review dict material
1. element of key
dict_keys(['id', 'title', 'articles'])
2. get value by key 'articles'
克拉西瓦亞梅恰河是俄羅斯的河流，位於圖拉州和利佩茨克州內，屬於頓河的右支流，河道全長244公里，流域面積6,000平方公里，在每年11月下旬開始結冰，直至翌年4月上旬，河畔城鎮有葉夫列莫夫。
蠶豆嘧啶（，也稱爲2,6-二氨基-4,5-二羥基嘧啶，）是一種在蠶豆（學名："Vicia faba"）和家山黧豆（學名："Lathyrus sativus"）中發現的生物鹼，是蠶豆嘧啶葡糖苷的糖苷配基，通常情況下是以嘧啶酮式（2,6-二氨基-1-氫-5-羥基-4-嘧啶酮）或嘧啶二酮式互變異構體（2,6-二氨基-1,3-二氫-4,5-嘧啶二酮）存在。
```

programming concept

In [ ]:
```python
# 為了確保是繁體中文, 套一個簡體 => 繁體轉換
from opencc import OpenCC
cc = OpenCC('s2t') # 初始化轉換器
```

In [ ]:
```python
## 去除輸入文字中的網址資料
def filter_url_tag(atricle):
    import re
    rule = re.compile(r'[http|https]://[a-zA-Z0-9.?/&=:]*',re.S)
    return re.sub(rule, '', atricle)

# 斷詞結果去除停用詞 (form web download list)
def filter_stopwords(words, stop_words):
    if (stop_words):
```

```
        fil_words = [w for w in words if w not in stop_words]
    return fil_words
```

In [ ]:
```
import os
filename='stopwords.txt'
with open(filename, 'r', encoding='utf-8') as f:
    stop_words = [word.strip('\n') for word in f]
print(len(stop_words))
print(type(stop_words))
print(stop_words[0:20])
print(stop_words[50:70])
print(stop_words[160:180])
```

```
200
<class 'list'>
['the', 'of', 'is', 'and', 'to', 'in', 'that', 'we', 'for', 'an', 'are', 'by', 'be',
'as', 'on', 'with', 'can', 'if', 'from', 'which']
['一', '不', '在', '人', '有', '為', '以', '於', '上', '他', '後', '之', '來', '因',
'下', '可', '到', '由', '這', '也']
['則', '怎', '曾', '至', '致', '著', '諸', '自', '．', '。', '；', '、', '」', '「',
'！', '!', ',', '[', ']', '~']
```

In [ ]:
```
for n in tqdm(range(0, len(data))):

  element = data[n]
  article = element.get('articles') #取得文章內容
  article = cc.convert(article) # 轉繁體中文
  article = filter_url_tag(article)

  words = jieba.lcut(article, cut_all=False) #斷詞
  words = filter_stopwords(words, stop_words)

  element['token'] = words
  data[n] = element
```

```
100%|██████████| 50000/50000 [03:28<00:00, 239.26it/s]
```

In [ ]:
```
print(data[0].get('articles'))
print(data[0].get('token'))
```

```
克拉西瓦亞梅恰河是俄羅斯的河流，位於圖拉州和利佩茨克州內，屬於頓河的右支流，河道全長244公
里，流域面積6,000平方公里，在每年11月下旬開始結冰，直至翌年4月上旬，河畔城鎮有葉夫列莫夫。
['克拉西瓦亞梅恰河', '俄羅斯', '河流', '位於', '圖拉州', '利佩茨克州', '內', '屬於', '頓
河', '右', '支流', '河道', '全長', '244', '公里', '流域面積', '6', '000', '平方公里',
'每年', '11月', '下旬', '開始', '結冰', '直至', '翌年', '4月', '上旬', '河畔', '城鎮',
'葉夫列莫夫']
```

In [ ]:
```
# 斷詞結果存為json file
if not(os.path.isfile('wiki_tokenize.json')):
    with open('wiki_tokenize.json', 'w', encoding='utf-8') as f:
        json.dump(data, f)
    f.close
    print('save done')
```

KCM-keyword Correlation Models from Open Corpus

load wiki tokenized data and reference QA/Ans

In [ ]:
```
# data have been tokenized
```

```
# if data already have element, don't run read file step

if ('token' not in data[0].keys()):
    with open('wiki_tokenize.json', 'r', encoding='utf-8') as f:
        data = json.load(f)
    print('loading done')

print(len(data))
print(data[0].keys())
print(data[0].get('token'))
```

```
50000
dict_keys(['id', 'title', 'articles', 'token'])
['克拉西瓦亞梅恰河', '俄羅斯', '河流', '位於', '圖拉州', '利佩茨克州', '內', '屬於', '頓河', '右', '支流', '河道', '全長', '244', '公里', '流域面積', '6', '000', '平方公里', '每年', '11月', '下旬', '開始', '結冰', '直至', '翌年', '4月', '上旬', '河畔', '城鎮', '葉夫列莫夫']
```

In [ ]:
```
# quation rule: only calculation 名詞類別 數量
flag_list = ['n','ng','nr','nrfg','nrt','ns','nt','nz'] #Part of Speech list
```

In [ ]:
```
# 取得reference QA (keywords) from file
with open("ref_qa.txt", 'r', encoding='utf-8') as file:
    keywords = file.read().split(' ')
print(keywords)
```

```
['臺灣', '美國', '大學', '肺炎', '天安門', '歌手', '中國', '蔡英文', '立法院', '颱風']
```

In [ ]:
```
# 取得reference answer for keywords
with open("ref_ans.txt", 'r', encoding='utf-8') as file:
    ref_ans = [word.strip('\n').split(' ') for word in file]
print(ref_ans)
```

```
[['日本', '香港', '中國大陸', '分佈', '中國', '中華民國', '日治', '臺北市', '名稱', '臺北'], ['非建制地區', '城市', '人口普查', '加拿大', '英國', '地區', '加利福尼亞州', '國家', '伊利諾伊州', '公司'], ['學院', '學生', '美國', '課程', '研究', '畢業', '學校', '教授', '查理', '教育'], ['病例', '報告', '冠狀病毒', '新冠', '湖南省', '疫情', '感染者', '傳染性', '當日', '感染'], ['支隊', '中隊', '母親', '北京', '中國人民武裝警察部隊北京市總隊', '警衛', '學生', '國旗', '大隊', '丁子霖'], ['歌曲', '專輯', '演員', '音樂', '流行', '香港', '日本', '單曲', '韓國', '節目'], ['日本', '特有植物', '古代', '國家', '分佈', '廣東省', '印度', '美國', '臺灣', '研究'], ['總統', '民進黨', '主席', '中華民國總統', '韓國瑜', '批評', '時任', '宋楚瑜', '總統候選人', '民主進步黨'], ['立法委員', '黨團', '行政院', '條例', '委員會', '國會', '民進黨', '質詢', '法案', '中華民國'], ['聯合颱風警報中心', '升格', '強度', '等級', '颶風', '薩菲', '辛普森', '級別', '莫拉克', '日本氣象廳']]
```

begin 統計

concept:

1. 如果keyword在token內, 再進行詞性分析(by jieba.posseg as pseg)
2. 詞性分析後, 如果是名詞類, 則進行統計

In [ ]:
```
# 依據keywords數量開始刷tokenized data
data2 = []
for n in tqdm(range(0, len(keywords))):
    qa = keywords[n]
    temp ={}

    # 開始刷
    for m in range(0, len(data)):
        token = data[m].get('token')
```

```
        article = data[m].get('articles')
        # print(' '.join(map(str, token))) # 為了增加重作斷詞的正確性, 把list轉為空格隔

        """
        (bad method, 太久了!!!!)
        if (qa in token):
            jbresult = pseg.cut(' '.join(map(str, token))) # 重新做一次jieba斷詞, 取得
            for (word, flag) in jbresult:
                if (word != ' ') and (flag in flag_list): # <= 移除空格
                    # print(word, end='|')
                    temp[word] =  temp.get(word, 0) + 1
        """

        if qa in article:
            for word in token:
                temp[word] = temp.get(word, 0) + 1

    data2.append(temp)
```

```
100%|██████████| 10/10 [00:06<00:00,  1.56it/s]
```

In [ ]:

```
print(type(data2))
print(type(data2[0]))
print(list(data2[0].keys())[0:30])

element = data2[0]
ans = {k: v for k, v in sorted(element.items(),reverse=True , key=lambda item: item[

print(list(ans.keys())[0:30])
print(list(ans.values())[0:30])
```

```
<class 'list'>
<class 'dict'>
['HMV', '源自', '英國', '跨國', '連鎖', '名稱', '牠', '主人', '聲音', '縮寫', '來自',
'小狗', '聽', '留聲機', '商標', '原本', '從事', '生產', '發行', '音樂', '唱片', '後來',
'不再', '改為', '零售', '首間', '店', '1921年', '倫敦', '開業']
['臺灣', '等', '中', '1', '會', '.', '中國', '以及', '進行', '2', '開始', '香港', '日
本', '地區', '將', '使用', '第', '主要', '3', '美國', '包括', '其中', '擔任', '表示',
':', '來自', '當時', '前', '由於', '因此']
[5941, 5524, 4750, 2286, 2248, 2210, 2106, 1998, 1976, 1972, 1966, 1928, 1919, 1795,
1754, 1711, 1640, 1583, 1579, 1573, 1516, 1508, 1455, 1437, 1428, 1413, 1383, 1382,
1354, 1346]
```

In [ ]:

```
filename='stopwords.txt'
with open(filename, 'r', encoding='utf-8') as f:
    stop_words = [word.strip('\n') for word in f]
```

In [ ]:

```
ans = []
# Top10 related word
for n in range(0, len(data2)):
    print('---  ', keywords[n], '  ---')
    element = data2[n]

    # sorted by values (word frequency)
    element = {k: v for k, v in sorted(element.items(),reverse=True , key=lambda ite
    # review sorted result
    # print(list(element.keys())[0:30])
    # print(list(element.values())[0:30])

    T10_related_words = []

    article = ' '.join(map(str,list(element.keys())))
```

```
    jbresult = pseg.cut(article)

    for (word, flag) in jbresult:
        if (flag in flag_list) and (word not in stop_words) and (word != keywords[n]
            T10_related_words.append(word)
            print(word, flag, end='|')
        if (len(T10_related_words)>10):
            break
    # print(T10_related_words)
    print('\n'+'---'*20)

    ans.append(T10_related_words)
```

```
---   臺灣   ---
中國 n|香港 n|日本 n|地區 n|美國 n|發展 n|電影 n|活動 n|公司 n|專輯 n|部分 n|
------------------------------------------------------------
---   美國   ---
中國 n|公司 n|電影 n|研究 n|發現 n|國家 n|可能 n|地區 n|日本 n|英國 n|面積 n|
------------------------------------------------------------
---   大學   ---
中國 n|美國 n|研究 n|學生 n|學校 n|畢業 n|香港 n|發展 n|臺灣 n|學院 n|國家 n|
------------------------------------------------------------
---   肺炎   ---
病例 n|病毒 n|報告 n|香港 n|美國 n|冠狀病毒 n|研究 n|出院 n|治癒 n|朝鮮 n|疫情 n|
------------------------------------------------------------
---   天安門   ---
趙紫陽 n|中國 n|周恩來 n|江澤民 n|焦裕祿 n|北京 n|學生 n|鄧小平 n|毛澤東 n|香港 n|領導 n|
------------------------------------------------------------
---   歌手   ---
專輯 n|歌曲 n|單曲 n|音樂 n|電影 n|演唱會 n|演出 n|美國 n|推出 n|香港 n|作品 n|
------------------------------------------------------------
---   中國   ---
香港 n|地區 n|臺灣 n|日本 n|美國 n|發展 n|國家 n|公司 n|研究 n|上海 n|中國大陸 n|
------------------------------------------------------------
---   蔡英文   ---
臺灣 n|總統 n|民進黨 n|中華民國 n|國民黨 n|郝柏村 n|綠黨 n|立法院 n|主席 n|媒體 n|總統府 n
|
------------------------------------------------------------
---   立法院   ---
臺灣 n|總統 n|國民黨 n|選舉 n|中華民國 n|民進黨 n|代表 n|中國 n|政府 n|委員會 n|郝柏村 n|
------------------------------------------------------------
---   颱風   ---
聯合颱風警報中心 n|香港 n|日本氣象廳 n|升格 n|熱帶氣旋 n|熱帶低氣壓 n|公里 n|臺灣 n|發展 n
|熱帶風暴 n|澳門 n|
------------------------------------------------------------
```

In [ ]:

```
# 以參考答案計算得分
score = []
score_val = 0
limit = 3

for n in tqdm(range(0, len(keywords))):
    print(keywords[n], end=':  ')

    right = 0
    for a in ans[n]:
        if a in ref_ans[n]:
            right += 1
    print(right,'/',len(ref_ans[n]))
    score.append(right)
    if right >= limit:
        score_val += 1

print('---'*15)
```

```
print('limit:', limit)
print('得分:', score_val)
```

```
100%|██████████| 10/10 [00:00<00:00, 9635.43it/s]
臺灣:  3 / 10
美國:  4 / 10
大學:  6 / 10
肺炎:  4 / 10
天安門:  2 / 10
歌手:  5 / 10
中國:  5 / 10
蔡英文:  3 / 10
立法院:  3 / 10
颱風:  3 / 10
--------------------------------------------
limit: 3
得分: 9
```

summary by ken in 2021, Oct

1. KCM計算分數主要受斷詞影響
2. 使用Jieba斷詞方便易用, 但user_dict and stopwords是關鍵
3. 統計上相關性應該可以用TF-IDF取代, 例如 stopwords 應該就是出現在很多篇文章的詞, 應該重要性要被下修(懲罰)