自然語言文字分析

Dr. Cheng-Yu Ma

Postdoctoral Research Fellow

CAIM, Chang Gung Memorial Hospital

長庚醫療人工智能核心實驗室

Center for Artificial Intelligence in Medicine,
Chang Gung Memorial Hospital

馬誠佑 博士

博士：國立清華大學資訊工程研究所

碩士：國立陽明大學生物醫學資訊所

Review

- Get html: urllib、requests
- Text Recognition: regular expression
- Html/xml parser: beautifulsoup

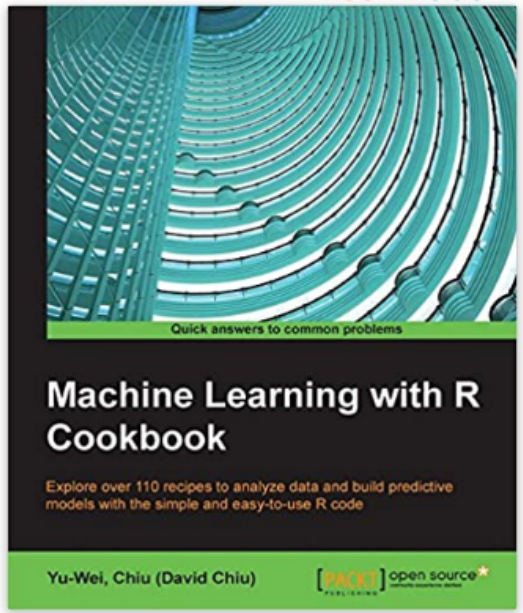
Machine Learning with R Cookbook - 110 Recipes for Building Powerful Predictive Models with R Illus

Kindle Edition

作者 Chiu (David Chiu), Yu-Wei (Author) | 格式： Kindle Edition

★★★½☆ 13等級

Look inside



Kindle
US$31.99

Paperback
US$28.17 - US$39.99

其他賣家
查看所有 3 版本

購買

USD33.59

列印價目表： ~~USD39.99~~ 省下 USD6.40 元 (16%)

電子書功能：
- 在書中醒目提示、記下筆記和搜尋
- 長度： 697 頁
- 增強輸入設定: 啟用
- 翻頁功能: 啟用
- 由於檔案較大，本書需花時間下載。

透過免費的 Kindle 應用程式（適用於 iOS、Android、PC 和 Mac）、Kindle 電子閱讀器及 Fire 平板電腦裝置閱讀。 查看所有支援的裝置

賣方： Amazon.com Services LLC

一鍵購買

發送免費樣品

傳送至您的 Kindle 或其他裝置

作為禮物贈送

輸入一個促銷代碼或者禮品卡

ISBN 13 碼: 978-1783982042
ISBN 10 碼: 1783982047
為什麼 ISBN 很重要？

此商品有更新的版本:

Machine Learning with R Cookbook - Second Edition: Analyze data and build predictive models
US$39.99
★½☆☆☆ (2)
現在可供下載。

16 個人覺得有用

有幫助 | 報告濫用

DataWookie

★★★☆☆ **Useful book to get you up and running quickly**

2015年7月6日在美國評論

“Machine Learning with R Cookbook” by Chiu Yu-Wei is nothing more or less than it purports to be: a collection of 110 recipes for applying Data Analysis and Machine Learning techniques in R. I was asked by the publishers to review this book and found it to be an interesting and informative read. It will not help you understand how Machine Learning works (that’s not the goal!) but it will help you quickly learn how to apply Machine Learning techniques to you own problems.

You can find the rest of my review at http://www.exegetic.biz/blog/2015/07/review-machine-learning-with-r-cookbook/.

有幫助 | 報告濫用

Taz

★☆☆☆☆ **Decent contents but poor book structure**

2015年8月9日在美國評論

Amazon 驗證購買

This book has pretty decent contents and well curated examples. But the serious problem is that there's no navigating bar supported for Mac OsX. I have contacted the customer service in many emails but circulating discussion, which made me send the screenshot for the diagnostic several times.

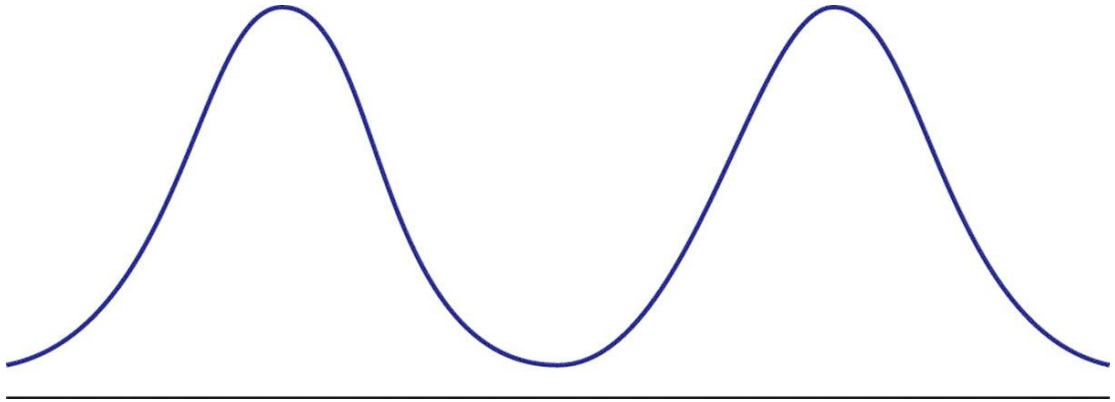
Missing the navigating bar is not tiny one as you need to find information yourself and back to the page in that you don't know where you are reading now. The publisher should fix this issue.

2 個人覺得有用

有幫助 | 報告濫用

過去如何了解使用者的反饋

- 評分(Ex:五星評分)
- 問卷調查
- 舉辦焦點團體



取樣方法或假設有錯誤時，
所得之推論也會有問題。

搜尋引擎找出跟流病的關係

- 當感冒的人多了，使用 Google 查詢「發燒」、「咳嗽」...等等，的民眾就跟著變多。

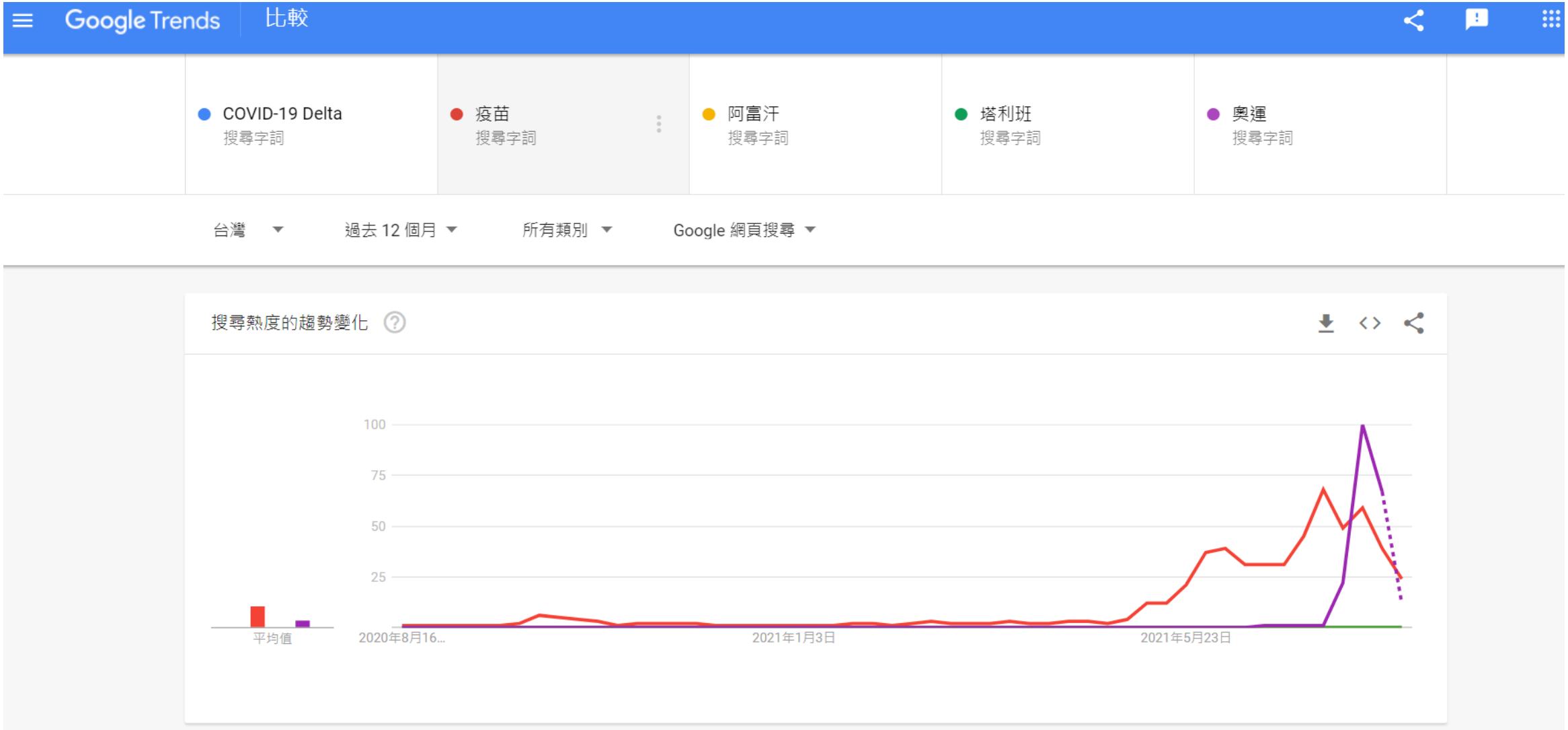
- 過去Google曾以特定關鍵字的搜尋熱門度，當作疫情變化的指標。（相關係數高達 0.85）

正相關係數 (介於 0 ~ 1 之間)	等級
≥ 0.8	超高度相關 (excellent correlation)
0.6 ~ 0.8	高度相關 (good correlation)
0.4 ~ 0.6	中度相關 (moderate correlation)
< 0.4	低度或無相關 (poor correlation)

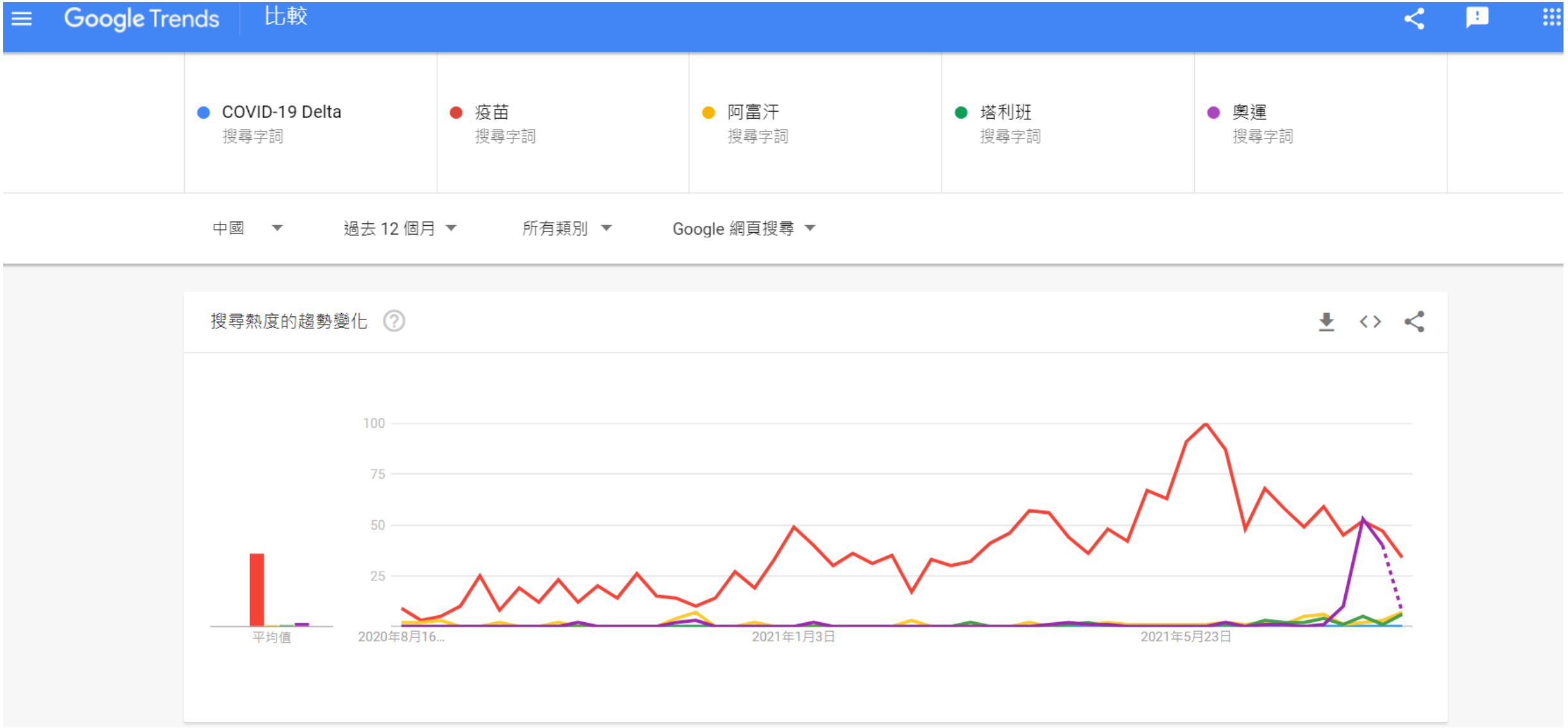
Google Trend

- https://trends.google.com.tw/trends/?geo=TW

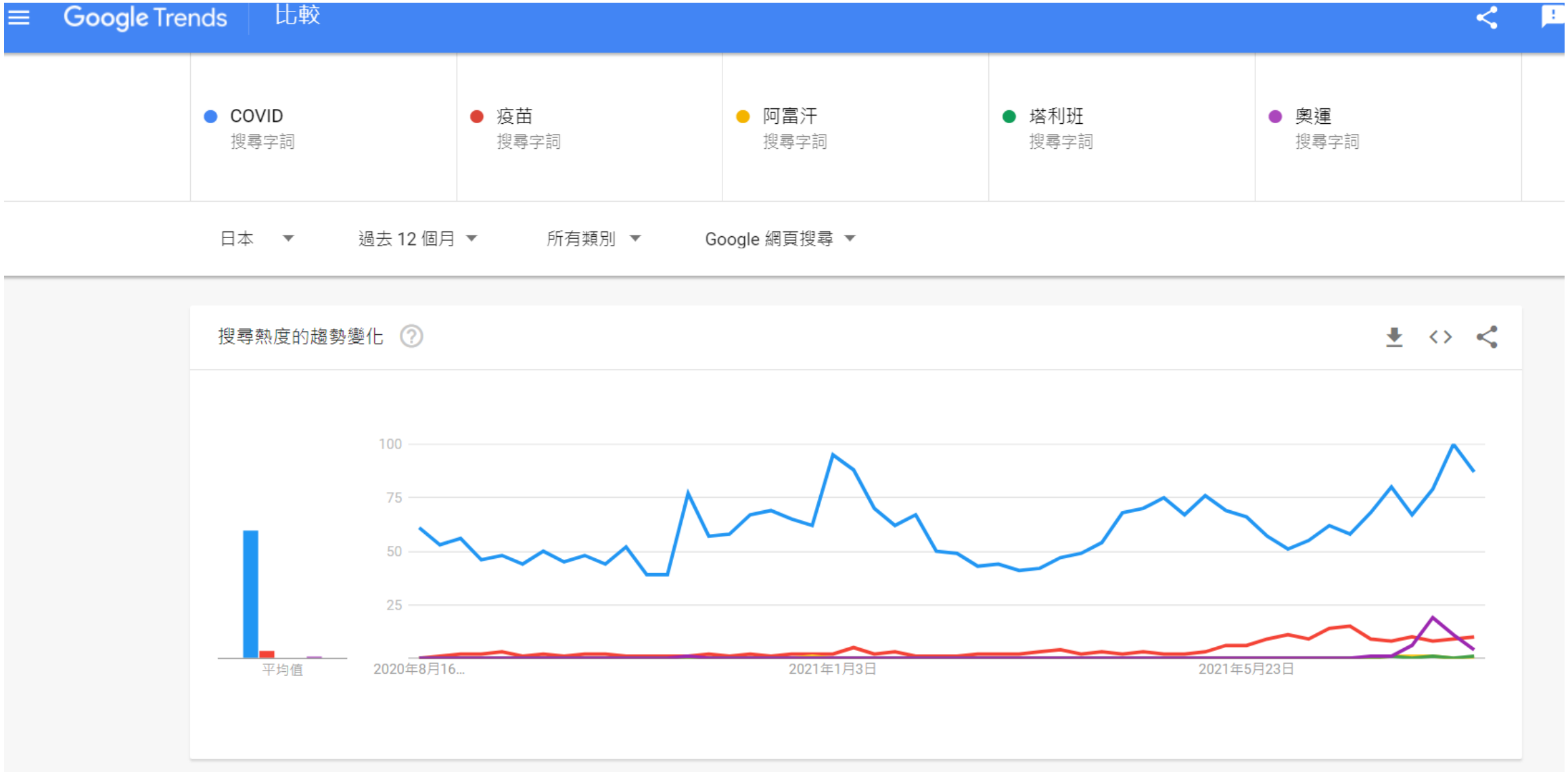
TW


Google Trends
比較
COVID-19 Delta
搜尋字詞
疫苗
搜尋字詞
阿富汗
搜尋字詞
塔利班
搜尋字詞
奧運
搜尋字詞
台灣
過去 12 個月
所有類別
Google 網頁搜尋
搜尋熱度的趨勢變化
100
75
50
25
平均值
2020年8月16...
2021年1月3日
2021年5月23日

China


Google Trends
比較
COVID-19 Delta
搜尋字詞
疫苗
搜尋字詞
阿富汗
搜尋字詞
塔利班
搜尋字詞
奧運
搜尋字詞
中國
過去 12 個月
所有類別
Google 網頁搜尋
搜尋熱度的趨勢變化
100
75
50
25
平均值
2020年8月16...
2021年1月3日
2021年5月23日

Japan


Google Trends
比較
COVID
搜尋字詞
疫苗
搜尋字詞
阿富汗
搜尋字詞
塔利班
搜尋字詞
奧運
搜尋字詞
日本
過去 12 個月
所有類別
Google 網頁搜尋
搜尋熱度的趨勢變化
100
75
50
25
平均值
2020年8月16...
2021年1月3日
2021年5月23日

Text Mining

- 傳統資料分析著重於結構化的資料
- Text mining – 從非結構化的文字中，萃取出有用的重要資訊。
- 應用:
  - 市調/民調
  - 關聯議題
  - 文章摘要
  - 文章分類
  - 聊天機器人
  - 評論分析

Text Mining: Process

文字處理
- 斷句
- 斷詞

量化
- 詞頻
- 文字矩陣
- TF-IDF

分析
- 文字雲
- 文章分群/分類
- 關聯分析

英文/中文斷詞

- Ex: I have an apple.

['I', 'have', 'an', 'apple']

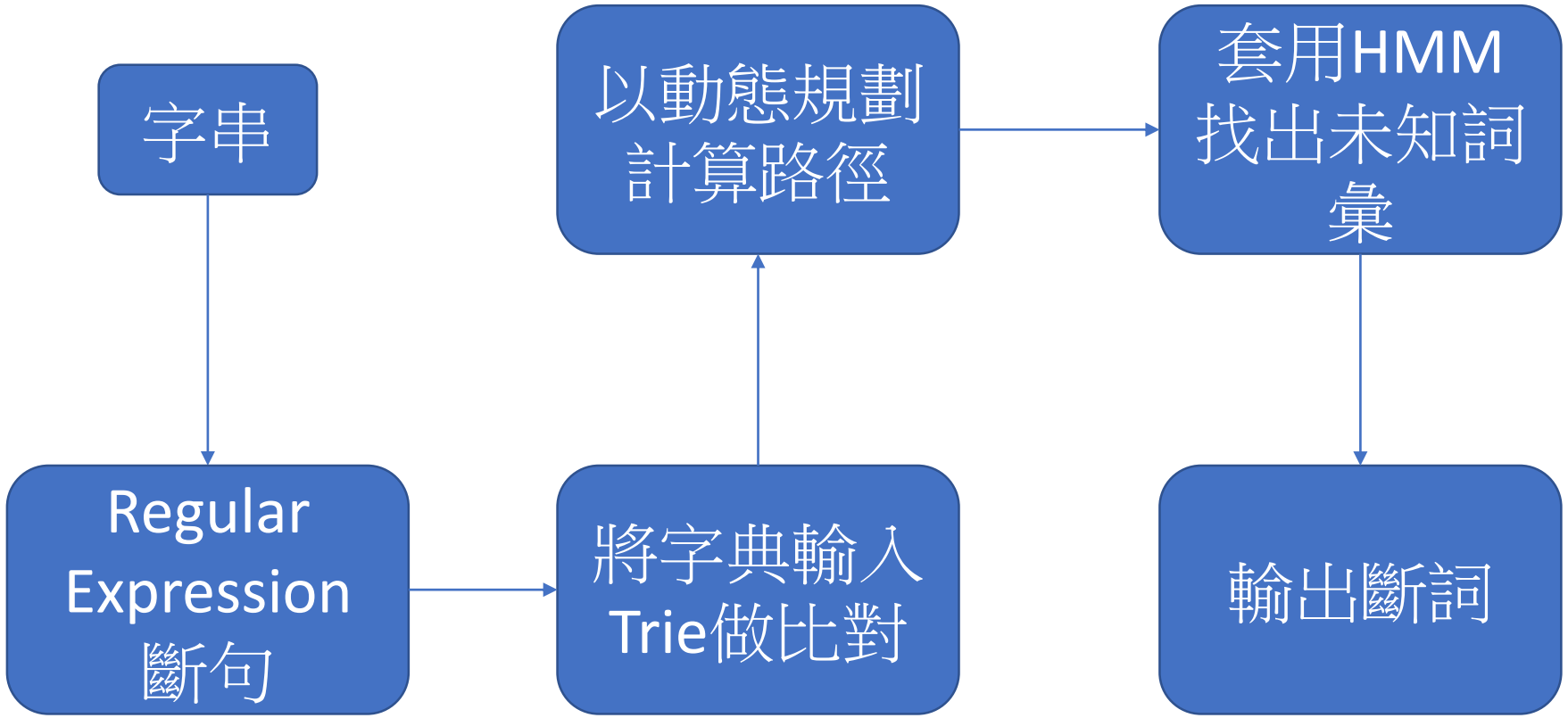
- 中文呢?

Ex: '高雄遭連日暴雨沖出天坑'

Jieba

- 生成句子中中文字所有可能成詞的情況，然後使用動態規劃（Dynamic programming）找出最大機率的路徑。
- 使用 Hidden Markov Model（HMM）及 Viterbi演算法來辨識新詞。
- https://github.com/fxsjy/jieba
- 安裝 : pip install jieba

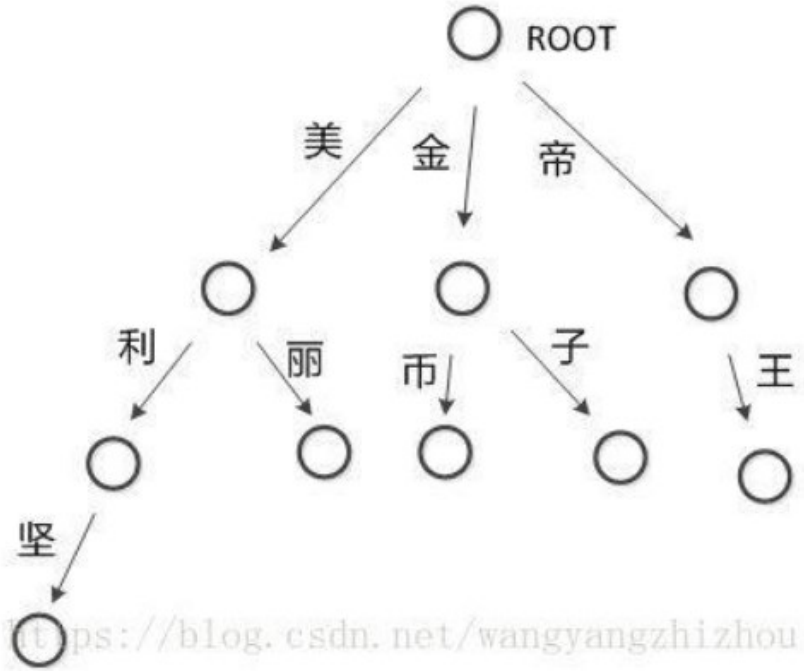
Jieba


字串
Regular Expression 斷句
將字典輸入 Trie做比對
以動態規劃 計算路徑
套用HMM 找出未知詞彙
輸出斷詞

Trie Tree (字典樹、首碼樹)

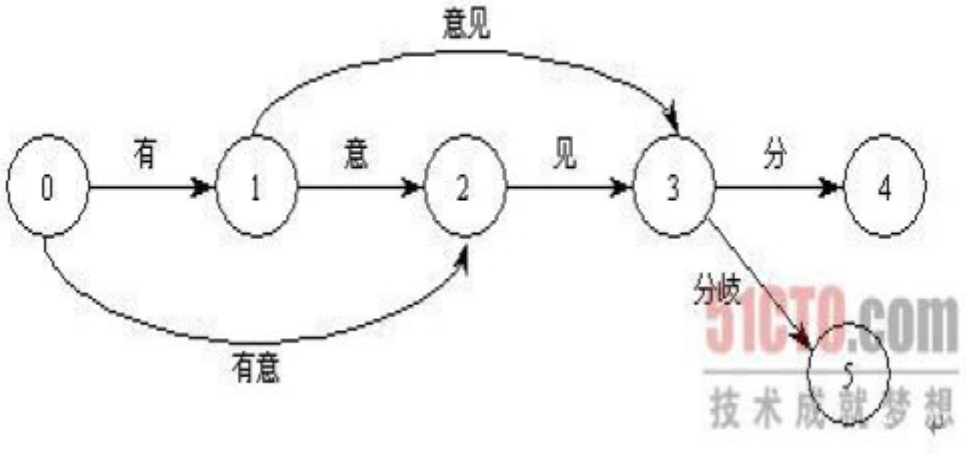
- Jieba 內建詞典 dict.txt 349,046個詞條，包含出現次數與詞性
- 繁中字典: (https://raw.githubusercontent.com/fxsjy/jieba/master/extra_dict/dict.txt.big)
- Trie Tree: 搜尋快速，耗費記憶體

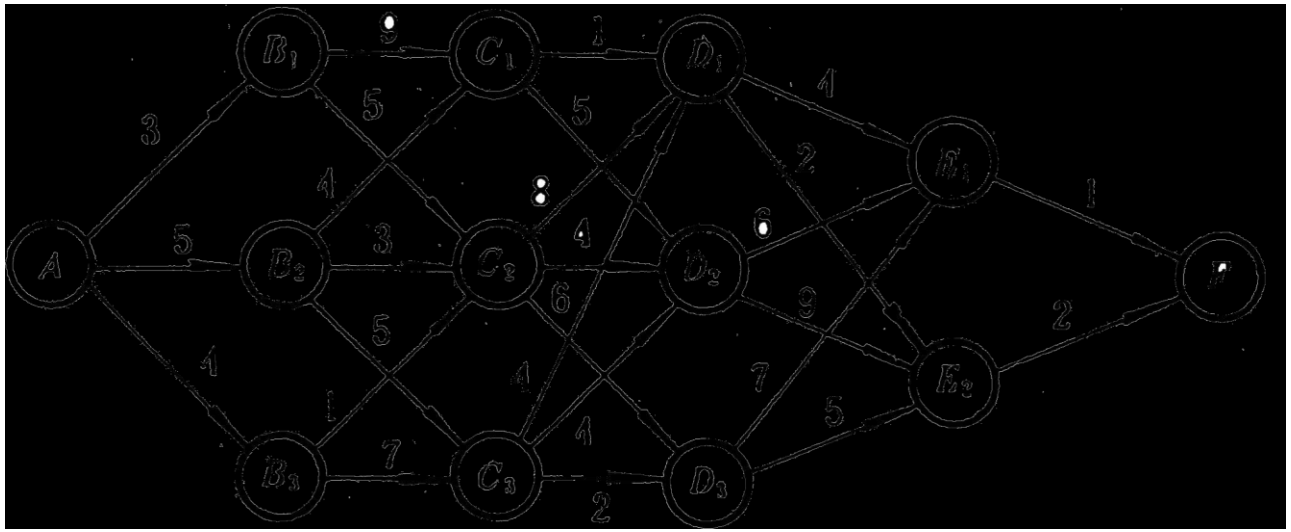
```
149496 折纸机 3 n
149497 折纹 8 n
149498 折线 13 n
149499 折线图 3 n
149500 折而族之 3 nr
149501 折耗 3 v
149502 折肉 3 n
149503 折腰 23 v
149504 折腰五斗 3 n
149505 折腾 777 v
149506 折色 5 n
149507 折节 2 n
149508 折节下士 3 n
```



Dynamic Programming

- 根據字典的trie tree 可得 DAG 並產生幾種候選斷詞路徑。
- 以動態規劃依照不同詞語出現的頻率 (次數/總數)，從句子右往左反向計算最大機率路徑的斷詞組合。





斷詞範例

```
[1]  !pip  install  jieba
```

```
Requirement already satisfied: jieba in /usr/local/lib/python3.7/dist-packages (0.42.1)
```

```
[3]  import  jieba
     seg_list  =  jieba.cut("高雄遭連日暴雨沖出天坑",  cut_all=True)
     print("Full  Mode:",  "/  ".join(seg_list))
     seg_list  =  jieba.cut("高雄遭連日暴雨沖出天坑",  cut_all=False)
     print("Default  Mode:",  "/  ".join(seg_list))
```

```
Full Mode: 高/ 雄/ 遭/ 連/ 日/ 暴雨/ 沖/ 出/ 天坑
Default Mode: 高雄/ 遭連日/ 暴雨/ 沖/ 出/ 天坑
```

- jieba.cut(text, cut_all=False)精確模式，將句子最精確的切開，適合文本分析。
- jieba.cut(text, cut_all=True)全模式，把句子中所有的可以成詞的詞語都斷出来，速度非常快。
- jieba.cut_for_search(text)搜索引擎模式，在精確模式的基礎上，對長的詞語再次切分，提高召回率，適合用於搜索引擎分詞。
- Jieba.lcut

可自己定義辭典

- jieba.load_userdict(‘selfdict.txt’)

- 辭典預設放在 yourENV\Lib\site-packages\jieba

判斷詞性

- 詞性定義: https://gist.github.com/luw2007/6016931

```
import jieba.posseg as pseg
words = pseg.cut("高雄遭連日暴雨沖出天坑")
for w in words:
    print(w.word, w.flag)
```

```
高雄 nr
遭連 v
日 m
暴雨 n
沖 yg
出 v
天坑 n
```

暫時增加新詞

```
sentence = "高雄遭連日暴雨沖出天坑"
words = jieba.cut(sentence, cut_all=False)
print("/ ".join(words))
jieba.add_word('高雄',100, 'ns')
jieba.add_word('遭',100, 'v')
jieba.add_word('沖出',100, 'v')
jieba.add_word('連日',100, 't')
words = posseg.cut("高雄遭連日暴雨沖出天坑")
for w in words:
    print(w.word, w.flag)
```

```
高雄/ 遭連日/ 暴雨/ 沖出/ 天坑
高雄 ns
遭 v
連日 t
暴雨 n
沖出 v
天坑 n
```

- add_word(word, freq=5, tag=None)
- del_word(word)

標籤	含意	標籤	含意	標籤	含意	標籤	含意
n	普通名词	f	方位名词	s	处所名词	t	时间
nr	人名	ns	地名	nt	机构名	nw	作品名
nz	其他专名	v	普通动词	vd	动副词	vn	名动词
a	形容词	ad	副形词	an	名形词	d	副词
m	数量词	q	量词	r	代词	p	介词
c	连词	u	助词	xc	其他虚词	w	标点符号
PER	人名	LOC	地名	ORG	机构名	TIME	时间

取出詞的位置

```
words = jieba.tokenize(sentence)

for tw in words:
    print(tw[0], tw[1], tw[2])
```

```
高雄 0 2
遭 2 3
連日 3 5
暴雨 5 7
沖出 7 9
天坑 9 11
```

使用網路爬蟲擴增字典

- https://zh.wikipedia.org/wiki/

可取得同義詞、翻譯詞

萌典

- https://www.moedict.tw/

新聞熱搜關鍵字

高端疫苗 | 指考 | 颱風 | 振興券 | 阿富汗

即時 | 熱門

16:19 生活 東協廣場流血鬥毆 中市議員質疑警方隔20分鐘才到 局長：...

16:18 寶島 黃金博物館礦山藝術季 巨型竹編地景

16:18 生活 聯亞傳未EUA卻已在檢驗、封緘 蔡炳坤批：不符程序難向...

16:16 體育 【新聞多益】梅西加盟新東家簽約用sign還是autograph?

16:16 財經 《國際產業》嫌168億美元太廉價 雪梨機場：再高一些可...

16:15 財經 《日股》緊急狀態恐延長 日股嚇挫1.62%

更多 即時新聞

中天新聞直播

訂閱中時新聞網

周一至五發送，每日五則庶民情報

立即訂閱

影音專區

利用Local資料建立詞典 -如何找出有意義的詞彙?

- 用N-gram做中文斷詞
- N= 2
  - 統計所有2-gram出現的次數
  - 可表示成機率: 出現次數除以總次數。

缺點: 沒有參考文法

高雄遭連日暴雨沖出天坑 => 高雄 雄遭 遭連 連日 日暴 暴雨...

生成n-gram

```
text = '高雄遭連日暴雨沖出天坑'

# 2-gram (bigram)
for i in range(0,len(text) - 2 + 1 ):
    print(text[i:i+2])

# 3-gram (trigram)
for i in range(0,len(text) - 3 + 1):
    print(text[i:i+3])
```

```
高雄
雄遭
遭連
連日
日暴
暴雨
雨沖
沖出
出天
天坑
高雄遭
雄遭連
遭連日
連日暴
日暴雨
暴雨沖
雨沖出
沖出天
出天坑
```

寫成 n-gram function

```
#n-gram  function

def  ngram(text,  n=2):
    grams  =  []
    for  i  in  range(0,len(text)  -  n  +  1):
        grams.append(text[i:i+n])
    return  grams
```

利用dictionary計算詞出現次數

```
import operator
text = ''
words = ngram(text, 2)
dict = {}
for w in words:
    if w not in dict:
        dict[w] = 1
    else:
        dict[w] = dict[w] + 1

words = sorted(dict.items(), key = operator.itemgetter(1), reverse = True)
for word, cnt in words:
    if cnt >= 4:
        print(word, cnt)
```

用Counter 統計詞頻

```
from collections import Counter
words = ngram(text,2)
c = Counter(words)
c.most_common(5)
```

```
[('疫苗', 6), ('el', 4), ('病毒', 4), ('接種', 4), ('口罩', 4)]
```

假使bi-gram 切出「大巨」、「巨蛋」
tri-gram 切出「大巨蛋」，該怎麼抉擇?

長詞優先法

- 最普遍被廣泛使用的斷詞方式

- 從句子的一端開始，取最長的詞串逐一比對辭典內的詞，若找到就把它當作斷詞的結果，再把句子中比對到的詞去除，剩下的部份再重複剛剛的動作，直到整句都斷詞完畢。

- 通常若是有夠大的辭典，長詞優先法的正確率可高達90%

長詞優先演算法

- 給定一個連續句子S
- 給定詞典D
- 從最長詞n = 4到 n=2:
  - 從左到右掃描S
    - 檢查S中是否有關鍵詞在D中
      - 如是則移除該關鍵詞
    - 回傳移除關鍵詞的句子s’
  - 用n-gram 將s’斷句
    - 將出現超過最小閥值的字加到字典D

依標點符號切開字詞

```
import  re
splitter  =  ',|。|、|;|「|」|\(|\)'
text  =  'Delta變種病毒現已成為美國主要
for  e  in  re.split(splitter,  text):
    print(e)
```

移除關鍵字

```
def removeKey(text, keywords):
    for word in keywords:
        text = text.replace(word, '')
    return text
```

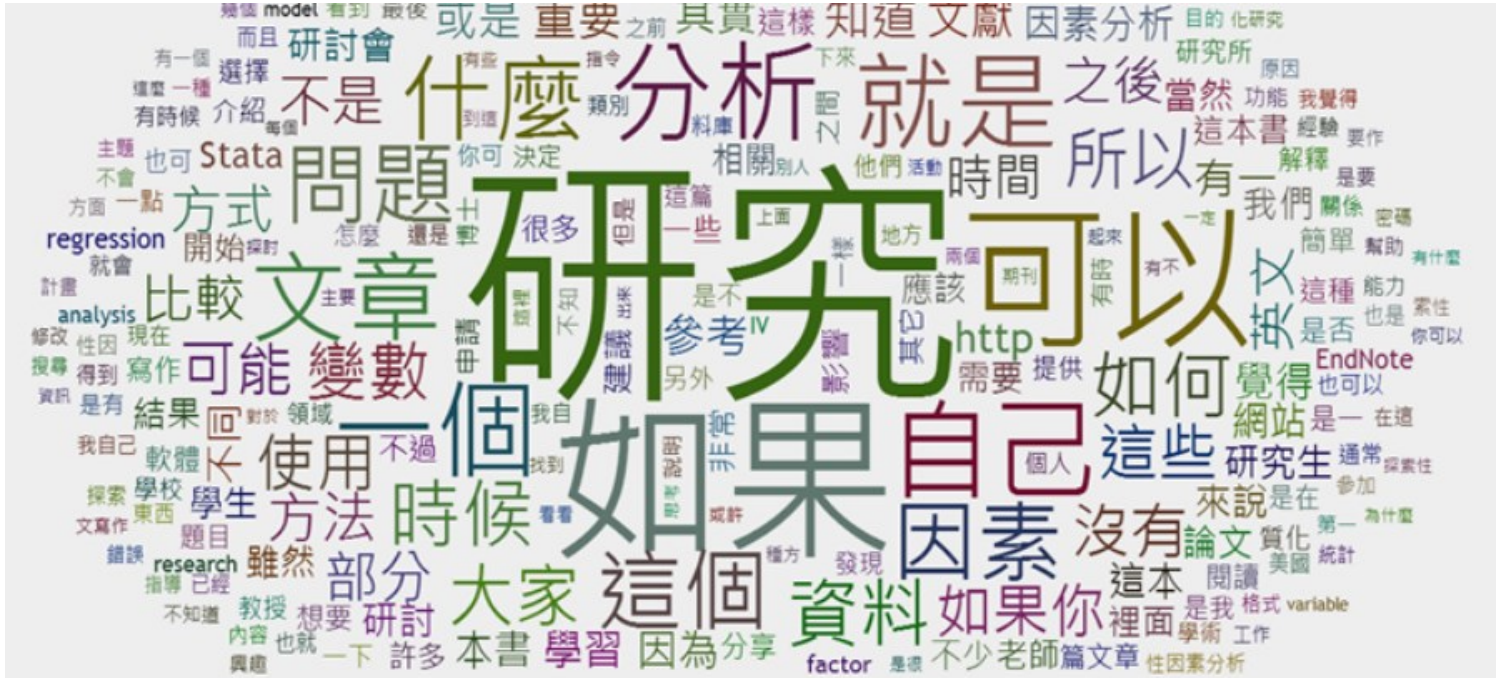
長詞優先演算法實作

```
keywords = []
for n in range(4,1, -1):
    words = []
    for sentence in sentenceArray:
        text_list = removeKey(sentence, keywords)
        words.extend(ngram(text_list, n))
    c = Counter(words)
    for word, cnt in c.items():
        if cnt >= 3:
            keywords.append(word)
```

文字量化分析

- 使用jieba 斷詞

- 統計詞頻



找出文章關鍵詞 TF-IDF (Term Frequency-Inverse Document Frequency)

- 如何判斷一個詞是不是關鍵詞?
- 如果某個詞在母群體中比較少見，但是在某篇文章中多次出現，那麼該詞很大可能反映了這篇文章的特性
- 可用來評估該詞對於該文件的重要程度
- 使用TF * IDF，假設單詞對文章的重要性越高，TF-IDF值就越大

TF & IDF

$$\mathrm{tfidf_{i,j}} = \mathrm{tf_{i,j}} \times \mathrm{idf_i}$$

- TF (Term Frequency)
  - 單詞在該文件的出現次數
  - 單詞w在文檔d中出現的次數: count(w, d)
  - 文檔d中總詞數: size(d)
  - tf(w,d) = count(w, d) / size(d)

$$\mathrm{tf_{i,j}} = \frac{n_{i,j}}{\sum_k n_{k,j}}$$

- IDF (Inverse Document Frequency)
  - 一個詞語普遍重要性的度量
  - 設文檔總數為n
  - 設詞w所出現檔數docs(w, D)
  - idf = log(n / docs(w, D))

$$\mathrm{idf_i} = \lg \frac{|D|}{|\{j : t_i \in d_j\}|}$$

實作TF-IDF

```
import scipy as sp
def tf_idf(t, d, D):
    tf = float(d.count(t)) / sum(d.count(w) for w in set(d))
    idf = sp.log(float(len(D)) / (len([doc for doc in D if t in doc])))
    return tf * idf

a, abb, abc = ["a"], ["a", "b", "b"], ["a", "b", "c"]
D = [a, abb, abc]

print(tf_idf("a", a, D))
print(tf_idf("b", abb, D))
print(tf_idf("a", abc, D))
print(tf_idf("b", abc, D))
print(tf_idf("c", abc, D))
```

```
0.0
0.27031007207210955
0.0
0.13515503603605478
0.3662040962227032
```

用Jieba 取出關鍵詞

```
import  jieba.analyse
text  =  'Delta變種病毒現已成為美國主要流行的病毒株，導致當地疫情急速升溫，近三分之二地區轉為熱區
tags  =  jieba.analyse.extract_tags(text,  20)
print(tags)
```

```
['接種', '疫苗', 'Delta', '變種', '美國', '口罩', '病毒', '疫情', '民眾', 'CDC', '配戴', '防疫', '
```

- jieba.analyse.extract_tags(sentence, topK=20, withWeight=False, allowPOS=())
  - sentence 文本
  - topK 為返回幾個 TF/IDF 權重最大的關鍵字，預設值為 20
  - withWeight 返回關鍵字權重值，預設值為 False
  - allowPOS 僅包括指定詞性的詞，預設值為空，即不篩選
- TF-IDF字典需維護: C:\ProgramData\Anaconda3\Lib\site-packages\jieba\analyse

建立詞頻矩陣

- 詞袋模型 (Bag of Words)
- 將文章斷詞以後，可以用詞向量表示文章。
- 這種表示方式如同將詞變成在袋子中零散且獨立的物件。

詞頻矩陣

- s = "柯文哲回應，關於疫苗的討論很多，他看媒體相當有計畫在進行認知作戰"
- s1 = "柯文哲質疑，AZ疫苗甚至跟日本買都可以，BNT疫苗若政府真的要買，需要拖這麼久嗎？“

Doc\單詞	柯文哲	回應	疫苗	質疑	討論	媒體	政府	買	…
S	1	1	1	0	1	1	0	0	…
S1	1	0	1	1	0	0	1	1	…

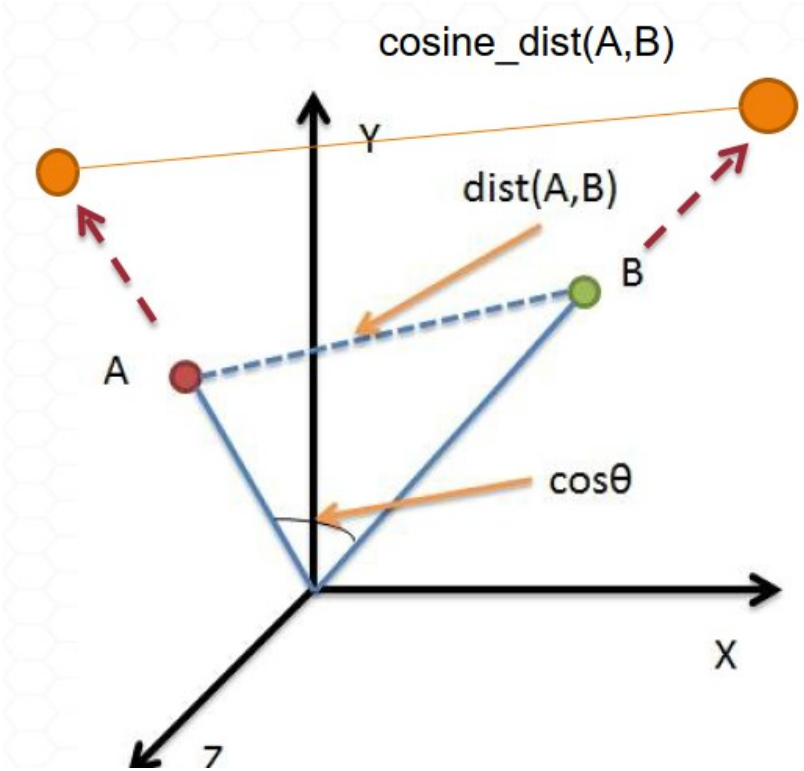
計算文章相似度

- 將文章斷詞，找出兩篇文章的單詞或關鍵詞
- 每篇文章各取出若干個單詞，合併成一個集合，計算每篇文章對於這個集合中的詞的詞頻
- 生成兩篇文章各自的詞頻向量
- 計算兩個向量的余弦相似度(Cosine Similarity)，值越大就表示越相似。

Euclidean Distance v.s. Cosine Distance

- 餘弦相似度的衡量不會受到向量大小的影響，因為在計算上會除以本身向量大小，類似標準化的動作

$$\cos\theta = \frac{\sum_{i=1}^{n}(A_i \times B_i)}{\sqrt{\sum_{i=1}^{n}(A_i)^2} \times \sqrt{\sum_{i=1}^{n}(B_i)^2}} = \frac{A \cdot B}{|A| \times |B|}$$



Euclidean Distance v.s. Cosine Distance

Cosine Distance

- 句子A：我 1，喜歡 2，看 2，蔡依林1，跳舞 1，不 1，也 0。
- 句子B：我 1，喜歡 2，看 2，蔡依林1，跳舞 1，不 2，也 1。
- 句子A：[1, 2, 2, 1, 1, 1, 0]
- 句子B：[1, 2, 2, 1, 1, 2, 1]

$$\cos\theta = \frac{\sum_{i=1}^{n}(A_i \times B_i)}{\sqrt{\sum_{i=1}^{n}(A_i)^2} \times \sqrt{\sum_{i=1}^{n}(B_i)^2}} = \frac{A \cdot B}{|A| \times |B|}$$

$$\cos\theta = \frac{1\times1 + 2\times2 + 2\times2 + 1\times1 + 1\times1 + 1\times2 + 0\times1}{\sqrt{1^2+2^2+2^2+1^2+1^2+1^2+0^2} \times \sqrt{1^2+2^2+2^2+1^2+1^2+2^2+1^2}}$$

$$= \frac{13}{\sqrt{12} \times \sqrt{16}}$$

$$= 0.938$$

Cosine Similarity

- Cosine Similarity = 1 – Cosine Distance

```
#Cosine Similarity = 1 - Cosine Distance
from sklearn.metrics.pairwise import cosine_similarity
cosine_similarities = cosine_similarity(X).flatten()
print(cosine_similarities)
```

根據內容相似度產生推薦新聞

```
import  pandas

#讀取新聞資料
news  =  pandas.read_excel('/content/gdrive/MyDrive/for_python_class/news.xlsx')

#斷詞
corpus  =  []
titles  =  []
for  article  in  news.iterrows():
    titles.append(article[1].title)
    corpus.append('  '.join(jieba.cut(article[1].content)))
#將文中詞語轉換為詞頻矩陣
from  sklearn.feature_extraction.text  import  CountVectorizer
vectorizer  =  CountVectorizer()
X  =  vectorizer.fit_transform(corpus)
print(X.shape)

#建立TF-IDF  矩陣
from  sklearn.feature_extraction.text  import  TfidfTransformer
transformer  =  TfidfTransformer()
tfidf  =  transformer.fit_transform(X)
weight  =  tfidf.toarray()
```

根據內容相似度產生推薦新聞2

```
#推薦相關新聞
from  sklearn.metrics.pairwise  import  cosine_similarity
def  getSimiliarArticle(articleid,tfidf):
    print('[查詢文章]:{}'.format(titles[articleid]))
    cosine_similarities  =  cosine_similarity(tfidf[articleid],  tfidf).flatten()
    related_docs_indices  =  cosine_similarities.argsort()[-2::-1]

    for  idx  in  related_docs_indices:
        if  cosine_similarities[idx]  >  0.05:
            print('[相關文章]:{}  {}'.format(titles[idx],  cosine_similarities[idx]))


getSimiliarArticle(1,tfidf)
```

```
(30, 3241)
[查詢文章]:半導體教父張忠謀　用30年打造台灣的驕傲
[相關文章]:張忠謀宣布明年6月退休　劉德音接台積電董事長 0.11558841346690119
```

Homework

- 寫爬蟲建立自己的詞典for jieba & corpus 內容

- 用此詞典(可加入預設詞典) & 你的corpus，測試推薦相關新聞系統，越像越好。