GOPIC FOR YOUR TRAVEL

- 報告日期: 2016.08.04
- 受訓梯次: AB101
- 報告組別: Group 2

成員介紹

組長 鄭云鈞

組員 李鈺祥

組員 王玫涵

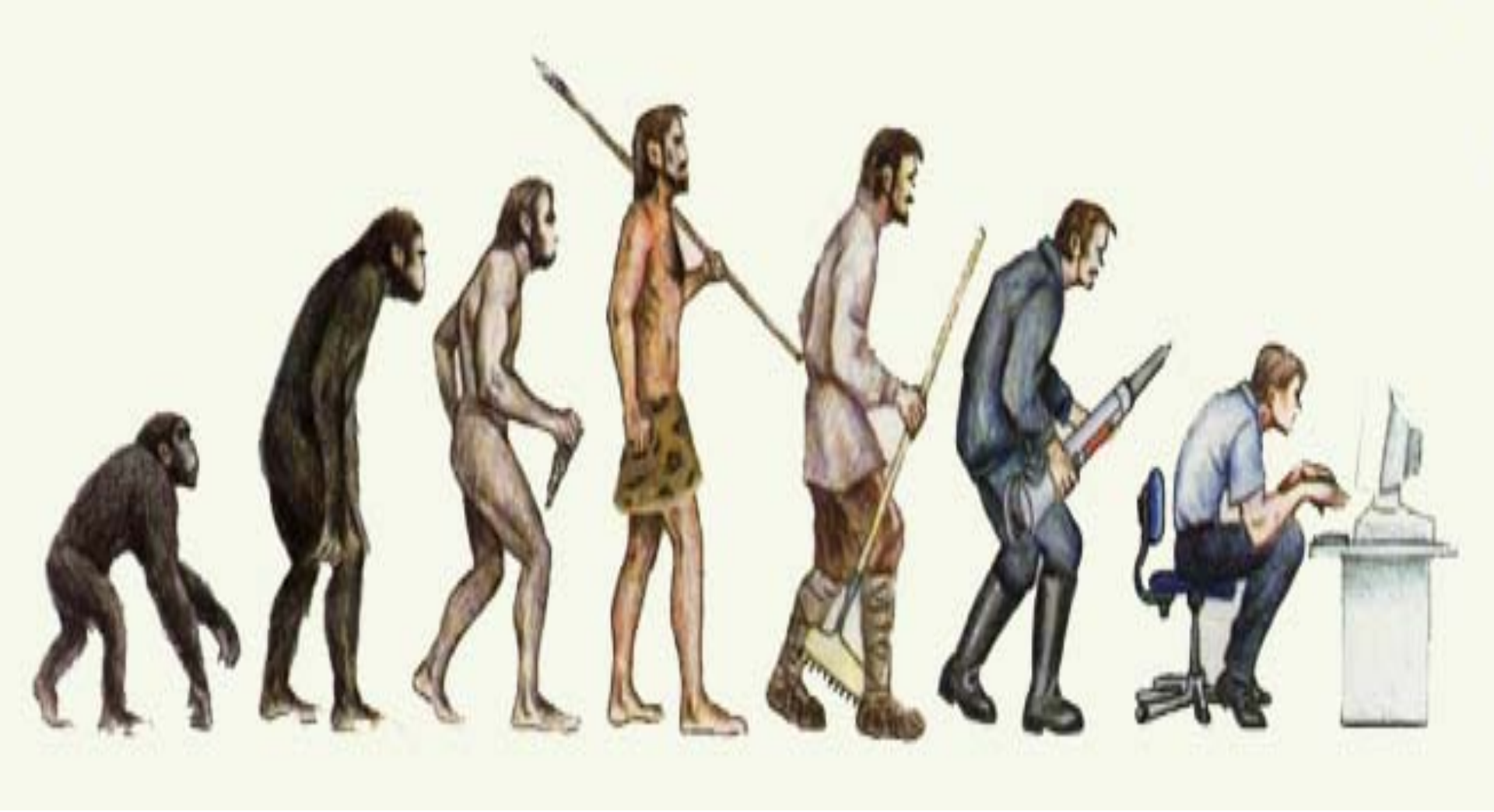
組員 倪裕程

組員 黃郁棻

組員 郭南宏

OUTLINE

- 專題源起
- 專題架構
- 資料收集
- 專題核心
- 未來展望
- Q&A



專題源起

黃郁棻

原想做旅行社行程/價格比較...

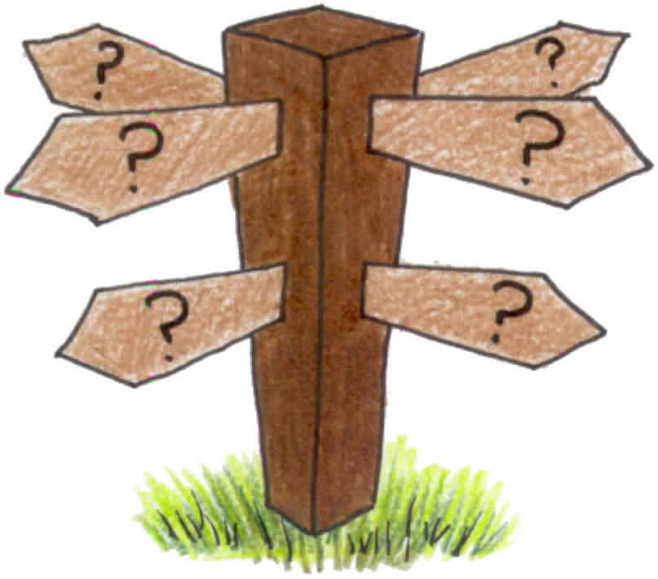
但旅遊咖和FUN TIME...

比較行程

景點

精選東京限量晴空塔展望台.橫濱紅磚倉庫.台場鋼彈.迪士尼.箱根遊船.溫泉饗宴5日	靜岡綠茶之鄉～東京迪士尼.久能東照宮.忍野八海.猿劇場.溫泉五日	《線上旅展2000》東京四大主題~迪士尼.鬼太郎.猿劇場.藤子不二雄.保證上晴空塔箱根.海盜船溫泉5日
刪除	刪除	較優 刪除
景點總數： 16處 極力推薦： 9 處 旅遊天數： 5 天	景點總數： 15處 極力推薦： 7 處 旅遊天數： 5 天	景點總數： 14處 極力推薦： 5 處 旅遊天數： 5 天
共同景點		
 富士箱根國立公園 國家公園^自然與公園	喚呀這景點我沒有	 富士箱根國立公園 國家公園^自然與公園
 箱根蘆之湖海盜船 遊船^搭船遊覽與水上運動^遊覽^戶外活動	喚呀這景點我沒有	 箱根蘆之湖海盜船 遊船^搭船遊覽與水上運動^遊覽^戶
 忍野八海 景點和地標^景點與地標	喚呀這景點我沒有	 忍野八海 景點和地標^景點與地標
 免稅店	 免稅店	 免稅店

只有圖卻不知道在哪……？



一張圖，勝過許多關鍵字！

V.S

該如何達成？

保證成團

【時尚東京】迪士尼樂園・晴空塔・杜莎夫人蠟像館・明治神宮・淺草觀
出發日期：8/04E、8/15E、8/21E

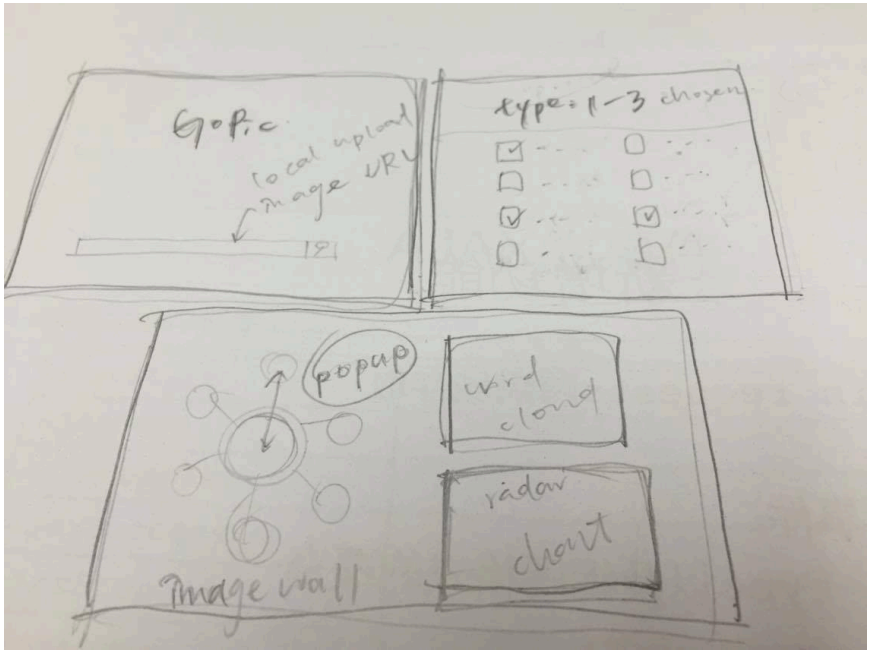
【東京雙樂園】迪士尼・KITTY樂園・海盜船・忍野八海・OUTLET・溫
出發日期：8/22G、8/25C

【富士鐵道・靜岡+東京】天上山纜車・淺間大社・迪士尼・湯瑪士列車
出發日期：8/11

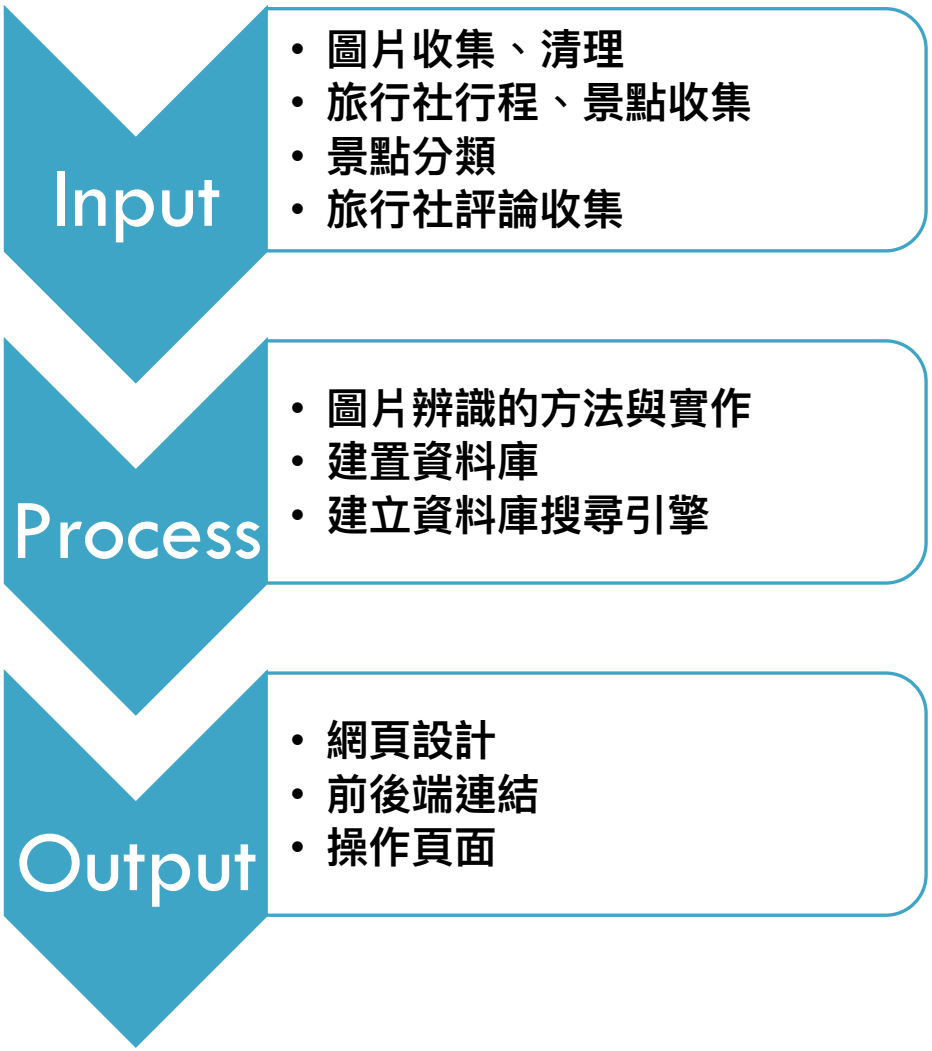
【FUN暑假。東京雙樂園】迪士尼・KITTY樂園・箱根遊船・淺草寺・O
5天

初步構想

- 以圖推薦旅行社行程
- 考量使用者偏好的景點種類
- 提供PTT/背包客棧對於旅行社的評價



專題架構


Input
• 圖片收集、清理
• 旅行社行程、景點收集
• 景點分類
• 旅行社評論收集
Process
• 圖片辨識的方法與實作
• 建置資料庫
• 建立資料庫搜尋引擎
Output
• 網頁設計
• 前後端連結
• 操作頁面

專題過程—工作配置

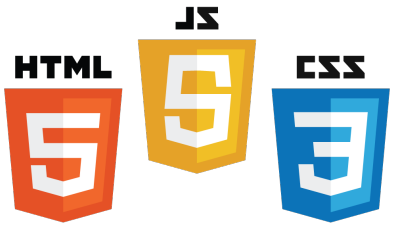
項目 / 成員	專題綜合處理	圖像辨識 & Hadoop	文字探勘	資料庫 & 前後端連接	前端網頁開發	機動協助
鄭云鈞	✓	✓		✓		
倪裕程	✓	✓		✓		
李鈺祥	✓		✓	✓		
王玫涵	✓	✓		✓		✓

WEB DESIGN

黃郁棻

使用工具

前端網頁



視覺呈現

jQuery
write less, do more.

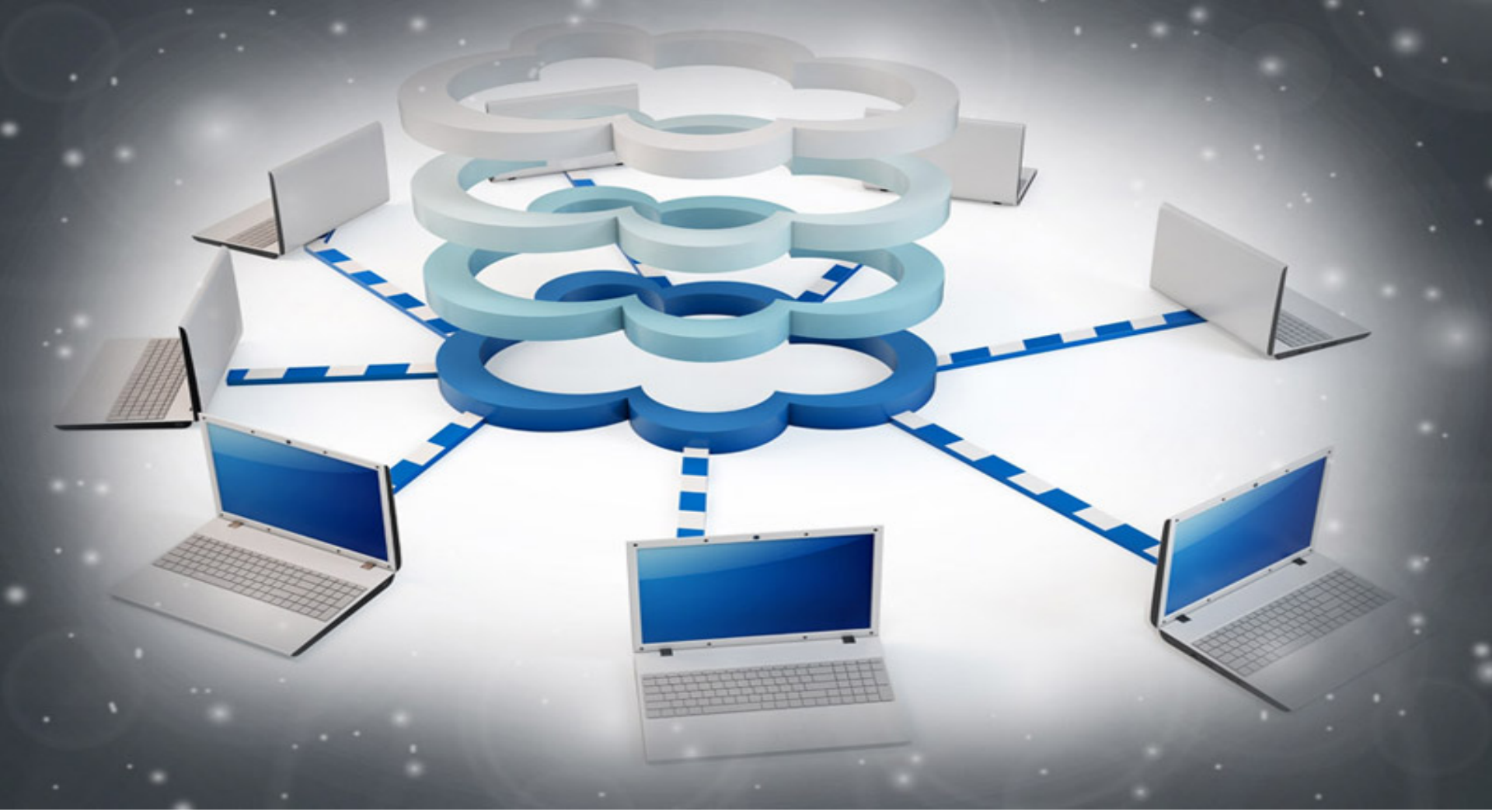
amCharts

後端連線

django

DEMO

黃郁棻



前後端連結與資料庫

王玫涵

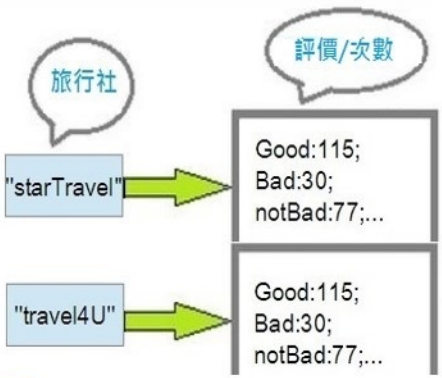
資料庫

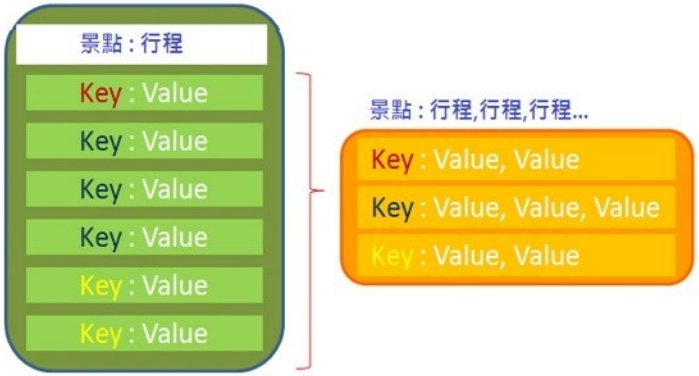
資料型態?

資料操作?
送出圖片,類型

資料庫

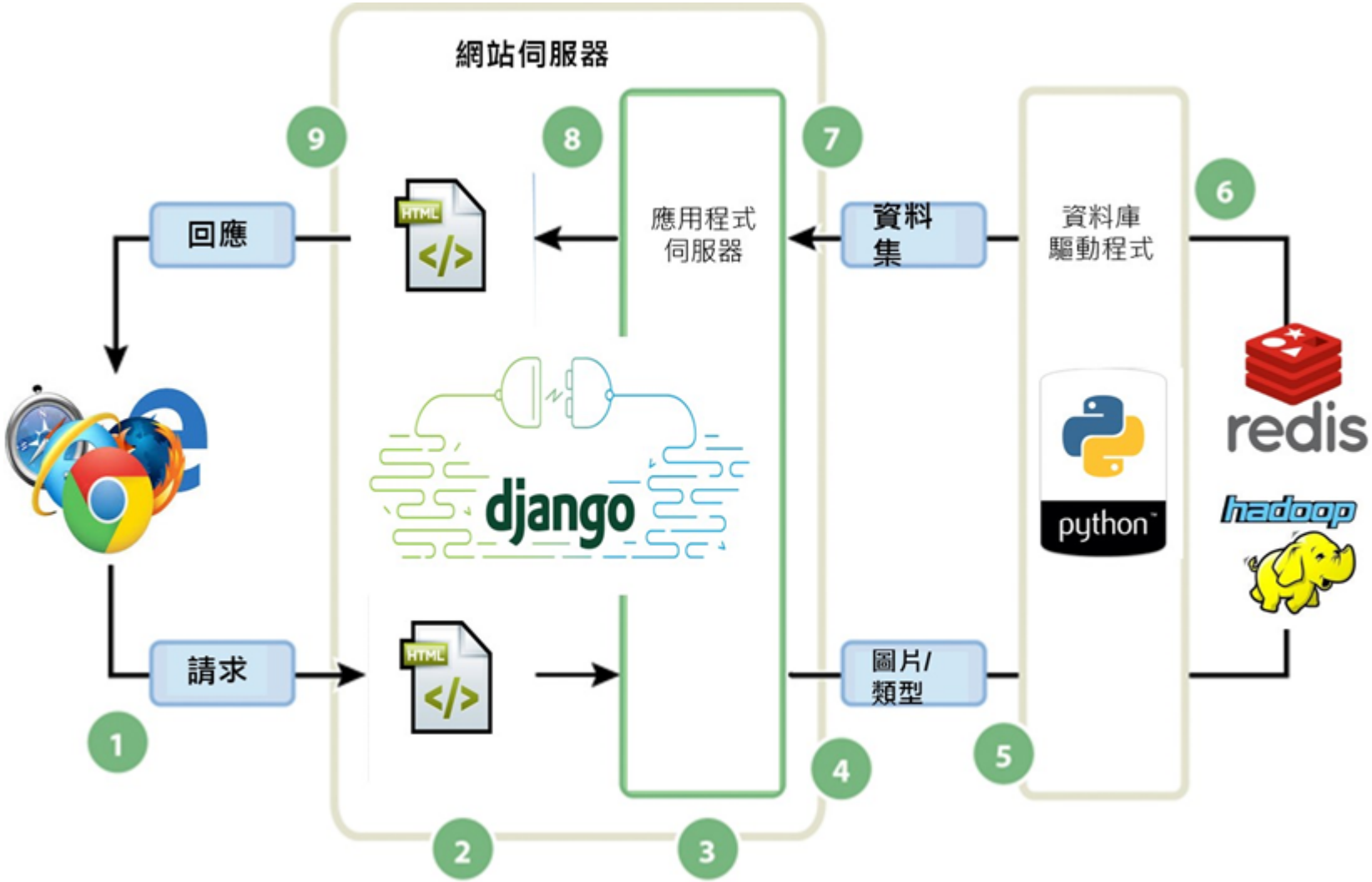
色塊特徵值 .csv
Hough line 特徵值 .txt





redis

前後端連結


網站伺服器
9
8
7
6
回應
應用程式
伺服器
資料
集
資料庫
驅動程式
HTML
</>
redis
django
python
hadoop
請求
圖片/
類型
1
2
3
4
5

資料收集與清理

行程收集與景點分類

郭南宏、鄭云鈞

旅行行程—行程爬取

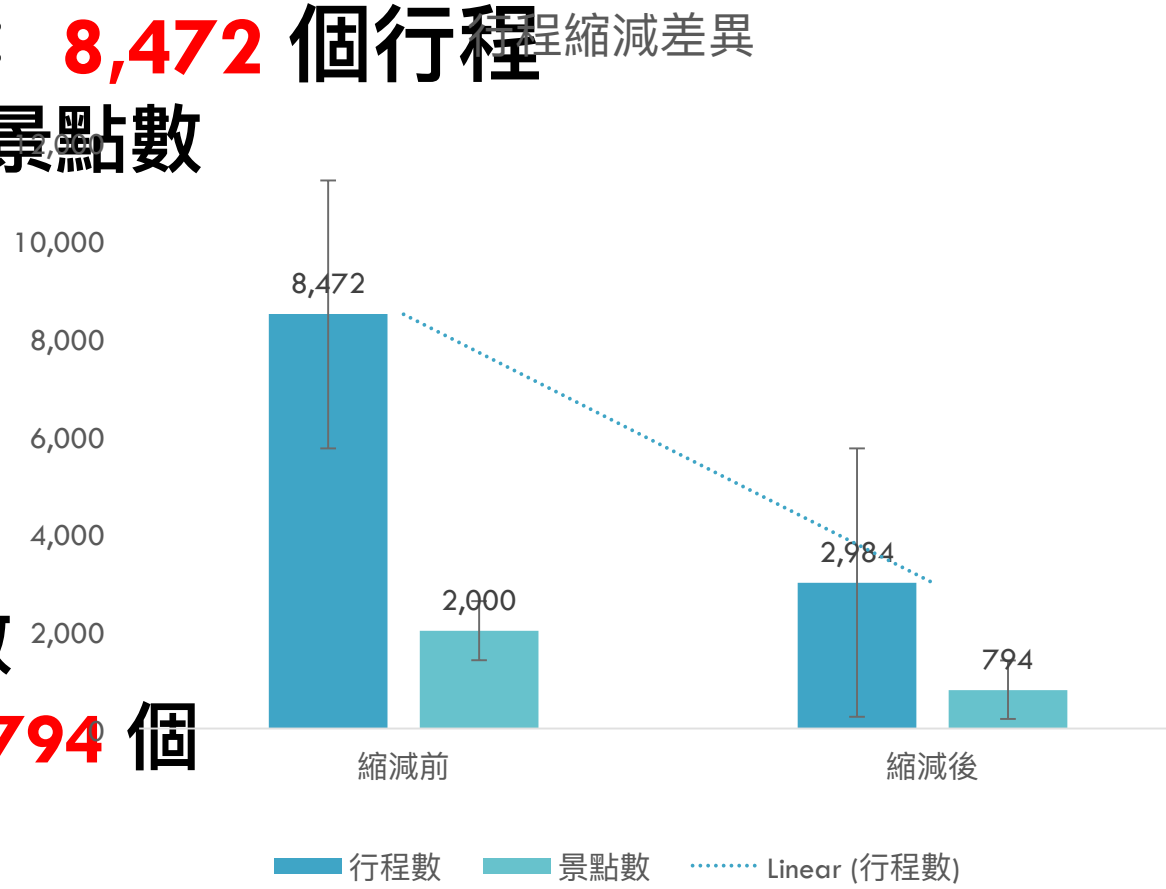
- BeautifulSoup
- Selenium

area 北海道、星野滑雪園、Club Med
href http://www.colatour.com.tw//C10A_TourSell/C10A06_TourItinerary.aspx?PatternNo=50331&GASource=國外旅遊E點通
price 35,900 起
prodNo 50331
subtitle 北海道好個秋(函館進千歲出)~函館小樽運、海洋尼克斯、札幌摩天輪、紅葉溫泉香五日(含稅) - colatour 國外旅遊
title 北海道好個秋(函館進千歲出)~函館小樽運、海洋尼克斯、札幌摩天輪、紅葉溫泉香五日(含稅)
第01天 桃園／函館~小丑幸運漢堡(車上享用)~金森倉庫群~函館百萬夜景(世界三大之夜景+纜車來回)~函館(或大沼)
第01天住宿 函館平成館溫泉旅館 或鹿部皇家溫泉 或 大沼王子溫泉或大沼greenpia溫泉 或 函館國際 或 函館溫泉IMAGINE RESORT 或 函館啄木亭溫泉旅館+自助晚餐 或同級旅館
第01天餐食 （早餐）溫暖的家 （午餐）機上輕食 （晚餐）飯店內用自助晚餐 或 日式御善海鮮料理
第02天 飯店~朝市~女子修道院~五稜郭城跡~大、小沼國立公園(紅葉)~洞爺湖展望台~昭和新山~熊牧場(每人發送兩顆蘋果或一袋餅乾可供餵食熊)~洞爺湖或登別溫泉(飯店內自助餐)
第02天住宿 洞爺湖觀光溫泉 或 登別PARK 雅亭溫泉飯店(大臨房間皆無WIFI) 或 登別萬世閣(大臨房間皆無WIFI) 或 登別石水亭 或同級旅館
第02天餐食 （早餐）飯店早餐 （午餐）日式海鮮定食 （晚餐）飯店內用自助晚餐
第03天 飯店~登別地獄谷(紅葉)~尼克斯海洋公園(企鵝遊行)~支笏湖(紅葉)~定山溪溫泉(或KIRORO渡假村)
第03天住宿 定山溪鹿之湯(大臨有WIFI/房間無) 或 定山溪HOTEL(大臨有WIFI/房間無) 或 定山溪GRAND 或 KIRORO渡假村 或同級旅館
第03天餐食 （早餐）飯店早餐 （午餐）日式壽喜鍋 （晚餐）飯店內自助晚餐 或 飯店內會席料理
第04天 飯店~豐平峽(紅葉)~免稅商店~小樽運河旅情(運河工藝館、音樂鐘館、北一硝子館)~銀之鐘咖啡座(贈送小樽風味咖啡杯及點心)~NORBESA摩天輪(眺望札幌風情)~豬肉涮涮鍋+三大蟹(吃到飽)+創意自助餐+軟飲無限~札幌
第04天住宿 RESOL 札幌(大臨房間皆無WIFI)或 札幌城市飯店 或 東急INN(大臨房間有WIFI) 或 札幌 ASPEN(大臨房間有WIFI) 或 札幌ART 或同級旅館
第04天餐食 （早餐）飯店早餐 （午餐）日式石狩鍋+烤花魚料理 （晚餐）豬肉涮涮鍋+三大蟹(吃到飽)+創意自助餐+軟飲無限
第05天 飯店~惠庭溪谷(紅葉)~OUTLET購物實場~千歲／桃園
第05天住宿 甜蜜的家
第05天餐食 （早餐）飯店早餐 （午餐）方便逛街~敬請自理 （晚餐）機上輕食
PatternNo 50307
area 北海道、星野滑雪園、Club Med
href http://www.colatour.com.tw//C10A_TourSell/C10A06_TourItinerary.aspx?PatternNo=50307&GASource=國外旅遊E點通
price 35,900 起
prodNo 50331
subtitle 北海道好個秋(千歲進函館出)~函館小樽運、海洋尼克斯、札幌摩天輪、紅葉溫泉香五日(含稅) - colatour 國外旅遊
title 北海道好個秋(千歲進函館出)~函館小樽運、海洋尼克斯、札幌摩天輪、紅葉溫泉香五日(含稅)
第01天 桃園／千歲~惠庭溪谷(紅葉)~支笏湖(紅葉)~定山溪溫泉(或KIRORO渡假村)
第01天住宿 定山溪鹿之湯(大臨有WIFI/房間無) 或 定山溪HOTEL(大臨有WIFI/房間無) 或 定山溪GRAND 或 KIRORO渡假村 或同級旅館
第01天餐食 （早餐）溫暖的家 （午餐）機上輕食 （晚餐）飯店內自助晚餐 或 飯店內會席料理

旅行行程—行程爬取

- 原行程數量： 8,472 個行程
  - 超過 2,000 個景點數

- 關西、關東：
  - 2,984 個行程數
  - 景點數縮減至 794 個



旅行行程—資料清理

- 遇到問題

桃園機場或松山機場／東京成田空港或東京羽田空港 → 飯店 → 成田(購物廣場~自由購物) 或 東京或千葉(自由活動~逛街購物)
(限航班為午前出發)箱根蘆之湖復古海盜船．遊覽富士山湖光山色 → 關東古老神社之一．箱根神社 → 御殿場OUTLET．
人氣品牌匯集~可眺望富士山 → 富士五湖溫泉鄉或石和溫泉鄉日本動畫大師．宮崎駿．三鷹之森吉卜力美術館 → 淺草觀音寺．
江戶繁華老街(仲見世通)．可遠眺東京晴空塔 → 免稅店 → 台場繽紛城．DiverCity Tokyo Plaza~巨大鋼彈戰士模型全日迪士尼樂園
或 海洋迪士尼(一票到底)~搭乘電車體驗（選擇樂園或海洋，請於出發前七個工作日告知客服人員，未事先告知的貴賓，
視同以樂園為主）自由活動~飯店→東京成田空港或東京羽田空港／桃園機場或松山機場

- 每個行程區分景點、說明的表點符號不一定一樣，本專題只要留景點名稱

旅行行程—資料清理

原始行程資料

台北松山機場東京羽田空港→東京自由活動・
建議您可自費搭乘電車前往(1)有樂町、銀座時尚百貨名店街
(2)新宿 大型知名日系百貨、美食餐飲店以及歌舞伎町 (3)橫濱未來21～紅磚赤瓦倉庫群等
淺草觀音寺·雷門～仲見世通→東京晴空塔～
晴空街道商店街→★特別安排【宮崎駿·三鷹之森吉普力美術館】→富士五湖區
日本百選名水～國家天然紀念物〈忍野八海〉→藝猴雜技表演～河口湖猿劇場→
路經彩虹大橋→免稅店→台場DiverCity Tokyo Plaza～超大鋼彈模型立像
全日 東京迪士尼樂園 或 迪士尼海洋(盡情暢遊一票到底) ～搭捷運電車、地下鐵或專車接送
自由活動・建議您可自費搭乘電車前往(1)築地場外市場
(2)上野阿美橫町傳統商店街→東京羽田空港台北松山機場

行程資料前處理

台北松山機場東京羽田空港
東京自由活動・建議您可自費搭乘電車前往(1)有樂町、銀座時尚百貨名店街
自由活動(2)新宿 大型知名日系百貨、美食餐飲店以及歌舞伎町 (3)橫濱未來21～紅磚赤瓦倉庫群等
淺草觀音寺·雷門～
仲見世通
東京晴空塔～
晴空街道商店街
★特別安排【宮崎駿·三鷹之森吉普力美術館】
富士五湖區
日本百選名水～國家天然紀念物〈忍野八海〉
藝猴雜技表演～河口湖猿劇場
路經彩虹大橋
免稅店
台場DiverCity Tokyo Plaza～超大鋼彈模型立像
全日 東京迪士尼樂園 或 迪士尼海洋(盡情暢遊一票到底) ～搭捷運電車、地下鐵或專車接送
自由活動・建議您可自費搭乘電車前往(1)築地場外市場 (2)上野阿美橫町傳統商店街
東京羽田空港台北松山機場

旅行行程—資料清理

利用Python進行去除不必要資料及景點名字轉換

```
import os

for root, dirs, files in os.walk('attraction\\'):
    for f in files:

        a = os.path.join(root, f)
        title = a.split('\\')[1]
        no = a.split('\\')[2]

        with open (a, 'r') as r:
            dir_name = "tourname\\" + title
            if not os.path.exists(dir_name):    #先確認資料夾是否存在
                os.makedirs(dir_name)
            file = open(dir_name + "\\" + no , "w")
            try:
                for i in r:
                    ele = i.decode('utf-8')
                    if u'機場' in ele:
                        continue
                    elif u'自由活動' in ele:
                        continue

                    else:
                        m = changename(ele)
                    file.write(m.encode('utf-8'))
            except:
                print a
                continue
```

```
def changename(i):
    dictour = {
            u'淺草觀音寺':u'Senso_ji_Temple',
            u'仲見世':u'Shin_Nakamise_Shopping_Street',
            u'晴空街道商店街':u'Tokyo_Solamachi',
            u'晴空塔':u'Tokyo_Skytree',
            u'吉普力美術館':u'Ghibli_Museum',
            u'富士五湖':u'Notenburo_Tensui',
            u'忍野八海':u'Oshino_Hakkai',
            u'河口湖':u'Kawaguchiko_Lake',
            u'彩虹大橋':u'Rainbow_Bridge',
            u'免稅':u'Aeon_Mall_Ayagawa',
            u'DiverCity':u'Diver_City_Tokyo_Plaza',
            u'迪士尼':u'Tokyo_Disneyland'
        }

    for ele in dictour:
        if ele in i:
            return dictour[ele] + '\n'
            break
    return i
```

旅行行程—資料清理

已整理資料

Senso_ji_Temple
Shin_Nakamise_Shopping_Street
Tokyo_Skytree
Tokyo_Solamachi
Ghibli_Museum
Notenburo_Tensui
Oshino_Hakkai
Kawaguchiko_Lake
Rainbow_Bridge
Aeon_Mall_Ayagawa
Diver_City_Tokyo_Plaza
Tokyo_Disneyland

旅行行程—分數制定

- 拉手分數
- 計算每個景點在旅行社行程出現的熱門度

```
import os
score = {}
dicscore = {}
num = 0
for root, dirs, files in os.walk('tourname\\'):
    for f in files:

        a = os.path.join(root, f)
        title = a.split('\\')[1]
        no = a.split('\\')[2]

        with open (a, 'r') as r:
            num += 1
            for ele in r:
                b = repr(ele).strip("'").split('\\')[0]
                b = b.decode('utf-8')
                if b not in score:
                    score[b] = 1
                else:
                    score[b] += 1
```

```
[(u'Aeon_Mall_Ayagawa', 1932), (u'Tokyo_Disneyland', 497), (u'Sannenzaka_Ninenzaka', 494), (u'Kiyomizu_dera_Temple', 486), (u'Togetsukyo', 441), (u'Shinsaibashi', 429), (u'Senso_ji_Temple', 422), (u'Tokyo_Skytree', 403), (u'Osaka_Castle', 386), (u'Dotonbori', 360), (u'Diver_City_Tokyo_Plaza', 353), (u'Universal_Studios_Japan', 346), (u'Hakone_Pirate_Ship', 342), (u'Todai_ji_Temple', 321), (u'Fushimi_Inari_taisha_Shrine', 318), (u'Shin_Nakamise_Shopping_Street', 316), (u'Sagano', 305), (u'Nara_Park', 292), (u'Hakone_Shrine_Kuzuryu_Shrine_Singu', 254), (u'Otonashi_Falls', 227), (u'Fuji_Hakone_Izu_National_Park', 223), (u'Nonomiya_Shrine', 215), (u'Rokuon_ji_Temple', 210), (u'Akashi_Kaikyo_Bridge', 204), (u'Oshino_Hakkai', 200), (u'Machikado_Amusement_Park_Mosaic_Garden', 195), (u'Rainbow_Bridge', 192), (u'Hanamikoji_Street', 186), (u'Jisyu_Shrine', 184), (u'Maiko_Park', 168), (u'Meiji_Jingu', 156), (u'Gion', 151), (u'Sagano_Romantic_Train', 149), (u'Isawa_Onsen', 145), (u'Kobe_Kitano_Museum_Ijinkan_gai', 138), (u'Yasaka_Shrine', 119), (u'Arima_Onsen', 117), (u'Notenburo_Tensui', 114), (u'Tokyo_Solamachi', 111), (u'Naritasan_Shinshoji_Temple', 98), (u'Yokohama_Red_Brick_Warehouse', 94), (u'St_Paul_s_Catholic_Church_Karuizawa', 90), (u'Byodoin_Temple', 89),
```

- 計算每個景點在所有行程出現的次數

```
import os
dicnum = {}
num = 0
for root, dirs, files in os.walk('tourname\\'):
    for f in files:
        dic = {}
        a = os.path.join(root, f)
        title = a.split('\\')[1]
        no = a.split('\\')[2].split('.txt')[0]
        no = no.decode('utf-8')
        dic[u'agency'] = title
        with open (a, 'r') as r:
            list = []
            num += 1
            for ele in r:
                for i in score:
                    b = repr(ele).strip("'").split('\\')[0]
                    b = b.decode('utf-8')
                    if b  == i:
                        dic[b] = score[i]
            dicnum[no] = dic
```

```
{"16JT731BR": {"Isawa_Onsen": 145, "Aeon_Mall_Ayagawa": 1932, "Shin_Nakamise_Shopping_Street": 316, "agency": "lionTravel", "Oshino_Hakkai": 200, "Tokyo_Solamachi": 111, "Meiji_Jingu": 156, "Hakone_Pirate_Ship": 342, "Tokyo_Skytree": 403, "Senso_ji_Temple": 422, "Rainbow_Bridge": 192, "Hakone_Shrine_Kuzuryu_Shrine_Singu": 254, "Tokyo_Disneyland": 497}, "PACTT-260 CX-TT160724A": {"Hakone_Kowakien_Yunessun": 17, "Shuzenji_Onsen": 25, "Enoden": 25, "Minato_Mirai_21": 63, "agency": "pacificTour", "Byodoin_Omotesando": 32, "Tsurugaoka_Hachimangu_Shrine": 29, "Enoshima_Island": 13, "The_Hakone_Open_Air_Museum": 10, "Izu_Kyoko_Line": 25, "Aeon_Mall_Ayagawa": 1932}, "VDR0000001943_OSA05CI0611B": {"Daiganbo": 17, "Kanazawa_Castle": 2, "World_Heritage_Shirakawago_Gassho_Frame_Housing_Community": 21, "Midosuji_Street": 78, "Mikurigaike_Pond": 17, "Sanmachi_Suji": 16, "Kappa_Bashi": 12, "Yuki_no_Otani": 1, "Tsuzumi_Gate": 1, "Kenrokuen_Garden": 14, "Kurobe_Dam": 17, "Aeon_Mall_Ayagawa": 1932, "Kurokabe_Square": 1, "agency": "eztravel", "Bijodaira": 12, "Rinku_Premium_Outlets": 18, "Pond_Taisho": 10, "Higashichaya_Old_Town": 7, "Yamanaka_Onsen": 3, "Dotonbori": 360, "Kamikochi": 17, "Shinsaibashi": 429, "Tojinbo_Cliff": 4, "Miho_Museum": 10}, "TYO05CI160724G": {"Isawa_Onsen": 145, "Aeon_Mall_Ayagawa": 1932, "Shin_Nakamise_Shopping_Street": 316, "agency": "bestTour", "Fuji_Hakone_Izu_National_Park": 223, "Oshino_Hakkai": 200, "Yokohama_Hakkeijima_Sea_Paradise": 42, "Meiji_Jingu": 156, "Hakone_Pirate_Ship": 342, "Tokyo_Skytree": 403, "Senso_ji_Temple": 422, "Hakone_Shrine_Kuzuryu_Shrine_Singu": 254, "Tokyo_Disneyland": 497}, "OSA04CI6830
```

旅行行程—分數制定

- 拉手分數
  - 行程中每個景點的分數加總當作拉手分數

```
import os
dictotal = {}
num = 0
for root, dirs, files in os.walk('tourname\\'):
    for f in files:
        dic = {}
        a = os.path.join(root, f)
        title = a.split('\\')[1]
        no = a.split('\\')[2].split('.txt')[0]
        no = no.decode('utf-8')
        dic[u'agency'] = title
        with open (a, 'r') as r:
            num = 0
            for ele in r:
                for i in score:
                    b = repr(ele).strip("'").split('\\')[0]
                    b = b.decode('utf=8')
                    if b  == i:
                        num += score[i]
            dictotal[no] = num
```

{"16JT731BR": 6902, "PACTT-260 CX-TT160724A": 2171, "VDR0000001943_OSA05CI0611B": 2999, "TYO05CI160724G": 4932, "OSA04CI6830Y": 3599, "FRN0000010835": 4686, "0BA3FED93711B971781D8FFBDEF22268": 3770, "OSA05BR160802HN": 11714, "16JX725BRR": 4987, "VDR0000001843_OSAGE16060305A": 5804, "16JX725BRB": 4800, "OSA05160828TB": 6472, "48893": 6918, "16JX725BR4": 5502, "VDR0000007614_JTP100160603IT": 6359, "GFG0000004368": 7165, "GFG0000003409": 6070, "PACTT-260 NH-TT160807A": 2112, "FRN0000014116": 8885, "PACTT-222

旅行行程—分數制定

- 類型分數
  - 使用TripAdvisor的景點所屬類型

```
url = 'https://www.tripadvisor.com.tw/Attractions-g294232-Activities-Japan.html'
import requests
from bs4 import BeautifulSoup as bs
import time, re
num = 0
rs = requests.session()
res = rs.get(url)
soup = bs(res.text, 'lxml')
for k in soup.select('div#CHILD_GEO_FILTER div.filter_list.al_border div.filter')[0:-1]:
    a = 'https://www.tripadvisor.com.tw' + k.select('a')[0]['href']
    res3 = rs.get(a)
    soup3 = bs(res3.text, 'lxml')
    if len(soup3.select('div.pageNumbers a')) > 0:
        if re.search(u'Activities', a) is not None:
            ccc = re.sub(u'Activities', u'Activities-oa{}', a)
            pagenum = int(soup3.select('div.pageNumbers a')[-1]['data-page-number'])

            for b in range(0, pagenum):
                res4 = rs.get(ccc.format(b*30))
                soup4 = bs(res4.text, 'lxml')
                try:
                    for ele in soup4.select('div#FILTERED_LIST'):
                        for i in ele.select('div.element_wrap'):
                            if len(i.select('div.p13n_reasoning_v2')) > 0:
                                for m in i.select('div.p13n_reasoning_v2 a'):
                                    dicKobe.setdefault(i.select('div.property_title a')[0]['href'].split('-')[4], []).append(m.text)
                                num += 1
                            else:
                                f = 'https://www.tripadvisor.com.tw' + i.select('div.property_title a')[0]['href']
                                res5 = rs.get(f)
                                soup5 = bs(res5.text, 'lxml')
                                if len(soup5.select('div.pageNumbers a')) == 0:
                                    for g in soup5.select('div#FILTERED_LIST'):
                                        for h in g.select('div.element_wrap'):
                                            for j in h.select('div.p13n_reasoning_v2'):
                                                for p in j.select('span.matchedTag.noTagImg'):
                                                    dicKobe.setdefault(h.select('div.property_title a')[0]['href'].split('-')[4], []).append(p.text)
                                else:
                                    aa = f.split('-')
                                    if re.search(aa[4], f) is not None:
                                        ddd = re.sub(aa[4], aa[4] + u'-oa{}', f)
                                        pagenum1 = int(soup5.select('div.pageNumbers a')[-1]['data-page-number'])

                                        for q in range(0, pagenum1):
                                            res6 = rs.get(ddd.format(q*30))
                                            soup6 = bs(res6.text, 'lxml')
                                            for r in soup6.select('div#FILTERED_LIST'):
                                                for s in r.select('div.element_wrap'):
                                                    for t in s.select('div.p13n_reasoning_v2'):
                                                        for u in t.select('span.matchedTag.noTagImg'):
                                                            dicKobe.setdefault(s.select('div.property_title a')[0]['href'].split('-')[4], []).append(u.text
                    time.sleep(2)
                except:
                    print ccc.format(b*30)
    else:
        for c in soup3.select('div#FILTERED_LIST'):
            for d in c.select('div.element_wrap'):
                if len(d.select('div.p13n_reasoning_v2')) > 0:
                    for e in d.select('div.p13n_reasoning_v2 a'):
                        dicKobe.setdefault(d.select('div.property_title a')[0]['href'].split('-')[4], []).append(e.text)
        time.sleep(2)
time_end= time.time()
```

旅行行程—分數制定

- 類型分數
  - 將每個行程景點轉換為TripAdvisor的景點類型

```
import os
dictype = {}
num = 0
for root, dirs, files in os.walk('tourname\\'):
    for f in files:
        dic = {}
        a = os.path.join(root, f)
        title = a.split('\\')[1]
        no = a.split('\\')[2].split('.txt')[0]
        no = no.decode('utf-8')
        dic[u'agency'] = title
        with open (a, 'r') as r:
            num = 0
            add = {}
            type = []
            for ele in r:
                for i in dicname:
                    b = repr(ele).strip("'").split('\\')[0]
                    b = b.decode('utf=8')
                    if b  == i:
                        add[b] = dicname[i]
            dictype[no] = add
```

{"16JT731BR": {"Isawa_Onsen": ["Spa 與養生"], "Aeon_Mall_Ayagawa": ["購物"], "Shin_Nakamise_Shopping_Street": ["購物"], "Oshino_H
akkai": ["景點與地標"], "Tokyo_Solamachi": ["購物"], "Meiji_Jingu": ["景點與地標"], "Hakone_Pirate_Ship": ["遊覽", "戶外活動", "搭
船遊覽與水上運動"], "Tokyo_Skytree": ["景點與地標"], "Senso_ji_Temple": ["景點與地標"], "Rainbow_Bridge": ["景點與地標"], "Hakone_Sh
rine_Kuzuryu_Shrine_Singu": ["景點與地標"], "Tokyo_Disneyland": ["水上樂園和遊樂場"]}, "PACTT-260 CX-TT160724A": {"Hakone_Kowakien

旅行行程—分數制定

- 類型分數
  - 計算每個行程各類型的個數

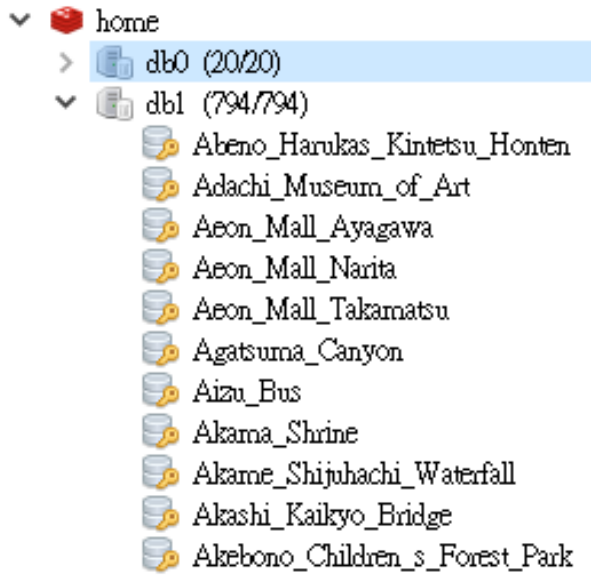
```
import os
dicscoresum = {}
for root, dirs, files in os.walk('tourname\\'):
    for f in files:
        dic = {}
        num = 0
        a = os.path.join(root, f)
        title = a.split('\\')[1]
        no = a.split('\\')[2].split('.txt')[0]
        no = no.decode('utf-8')
        with open (a, 'r') as r:
            for ele in r:
                b = repr(ele).strip("'").split('\\')[0]
                b = b.decode('utf=8')
                for i in dicname:
                    if b  == i:
                        for m in range(0, len(dicname[i])):
                            if dicname[i][m] not in dic:
                                dic[dicname[i][m]] = 1
                            else:
                                dic[dicname[i][m]] += 1
                for n in score:
                    if b == n:
                        num += score[n]
            dic[u'score'] = num


        dicscoresum[no] = dic
```

{"16JT731BR": {"購物": 4, "遊覽": 1, "搭船遊覽與水上運動": 1, "景點與地標": 6, "Spa 與養生": 1, "水上樂園和遊樂場": 1, "戶外活動": 1, "score": 6902}, "PACTT-260 CX-TT160724A": {"購物": 1, "自然與公園": 1, "博物館": 1, "景點與地標": 3, "交通": 2, "Spa 與養生": 2, "score": 2171, "旅客資源": 1, "遊覽": 1}, "VDR0000001943_OSA05CI0611B": {"購物": 3, "自然與公園": 7, "博物館": 2, "景點與地標": 13, "Spa 與養生": 1, "score": 2999, "戶外活動": 2}, "TYO05CI160724G": {"購物": 2, "自然與公園": 1, "動物園和水族館": 1, "搭船遊覽與水上運動": 1, "景點與地標": 5, "Spa 與養生": 1, "水上樂園和遊樂場": 1, "戶外活動": 1, "score": 4932, "遊覽": 1}, "OSA04CI6830Y": {"購

旅行行程—匯入資料庫

- 特定景點在那些行程出現

```
import os
dicattr = {}
listattr = []
num = 0
for root, dirs, files in os.walk('tourname\\'):
    for f in files:
        dic = {}
        a = os.path.join(root, f)
        title = a.split('\\')[1]
        no = a.split('\\')[2].split('.txt')[0]
        no = no.decode('utf-8')
        dic[u'agency'] = title
        with open (a, 'r') as r:
            num = 0
            add = {}
            type = []
            for ele in r:
                for i in dicname:
                    b = repr(ele).strip("'").split('\\')[0]
                    b = b.decode('utf=8')
                    if b  == i:
                        dicattr.setdefault(i, []).append(no)
```

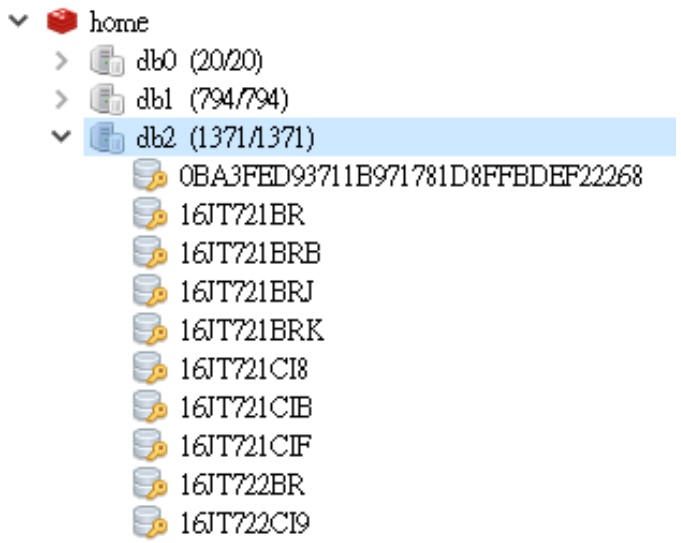



```
{"Korakukan": ["VDR0000001888_SDJ05160702A"], "Marukin_Soysouce_Memorial": ["OSA05CI160726BP", "OSA05CI160802DP", "OSA05CI160814
AP", "OSA05CI160816BP", "OSA05CI160820BP", "OSA05CI160904AP", "VDR0000001838_NPD061705CI6E", "VDR0000001888_HIJ05160614C", "VDR0
00001888_HIJ06160612B", "VDR0000001888_OSA05160604Y", "VDR0000001888_OSA05160614H", "VDR0000001888_OSA05160701N", "VDR0000001975_OSA05160628TS", "JHIJ5-CIK3", "GFG0000009417", "GFG0000009418"], "Hozugawa_River_Boat_Ride": ["VDR0000001846_16JX604CIH", "VDR0000001846_16JX629BRC", "VDR0000001943_OSA05BR6721J", "JOSBA5", "OSA05B6828J", "OSA05BR6725J", "OSA05BR6822J", "OSA05BR6904E", "OSA05BR6909E", "16JX724CIC", "16JX727BRC", "16JX731CIC", "16JX801BRH"], "Kotobukijo": ["VDR0000001888_HIJ06160609L"], "Kasamatsu_Park": ["VDR0000001843_OSAGE16060305A", "VDR0000001888_OSA05160616V", "VDR0000001943_OSA05BR6721J", "VDR0000001975_OSA05160703TC", "534", "OSA05B6828J", "OSA05BR6725J", "OSA05BR6822J", "OSA05BR6904E", "OSA05BR6909E", "OSA05CI6907J", "16JX726CI9", "GFG000002670", "GFG0000007050", "OSA05CI132", "OSA05CI1406", "OSA05CI16D"], "Goryokaku_Park": ["VDR0000001888_SDJ05160812B"], "SHIS
```

旅行行程—匯入資料庫

- 行程所有類型、拉手分數、旅行社、旅行社團號

```
import os
dictotalsum = {}
for root, dirs, files in os.walk('tourname\\'):
    for f in files:
        dic = {}
        list = []
        num = 0
        a = os.path.join(root, f)
        title = a.split('\\')[1]
        no = a.split('\\')[2].split('.txt')[0]
        no = no.decode('utf-8')
        dic[u'tourno'] = no
        dic[u'agency'] = title
        with open (a, 'r') as r:
                b = repr(ele).strip("'").split('\\')[0]
                b = b.decode('utf=8')
                list.append(b)
                for i in dicname:
                    if b  == i:
                        for m in range(0, len(dicname[i])):
                            if dicname[i][m] not in dic:
                                dic[dicname[i][m]] = 1
                            else:
                                dic[dicname[i][m]] += 1
                for p in name:
                    if p not in dic:
                        dic[p] = 0
                for n in score:
                    if b == n:
                        num += score[n]
            dic[u'score'] = num
        dictotalsum[no] = dic
```



{"16JT731BR": {"課程活動及工作坊": 0, "搭船遊覽與水上運動": 1, "購物": 4, "自然與公園": 0, "score": 6902, "博物館": 0, "動物園和水族館": 0, "agency": "lionTravel", "景點與地標": 6, "tourno": "16JT731BR", "演唱會和表演": 0, "交通": 0, "Spa 與養生": 1, "tour": ["Aeon_Mall_Ayagawa", "Oshino_Hakkai", "Hakone_Pirate_Ship", "Hakone_Shrine_Kuzuryu_Shrine_Singu", "Isawa_Onsen", "Meiji_Jingu", "Rainbow_Bridge", "Aeon_Mall_Ayagawa", "Senso_ji_Temple", "Shin_Nakamise_Shopping_Street", "Tokyo_Skytree", "Tokyo_Solamachi", "Tokyo_Disneyland"], "水上樂園和遊樂場": 1, "戶外活動": 1, "旅客資源": 0, "飲食": 0, "休閒與娛樂": 0, "遊覽": 1}, "PACTT-260 CX-TT1607

資料收集與清理

圖片收集與清理

黃郁棻

圖片來源：取得圖片URL

- 東京、京都、大阪、神戶共794個景點
- Google 圖片搜尋： 圖檔大小共約77G

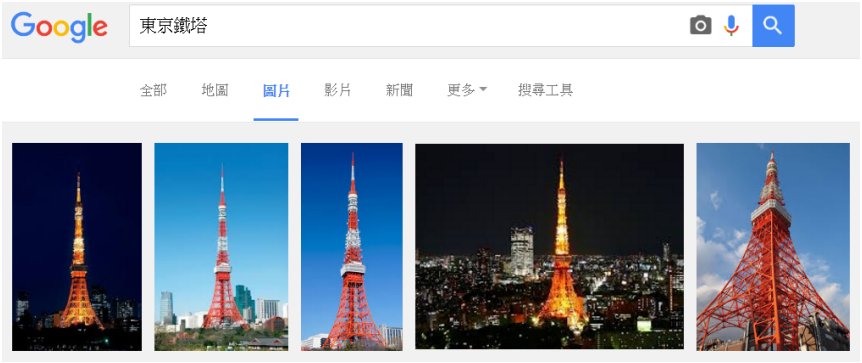


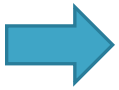
Image URL List

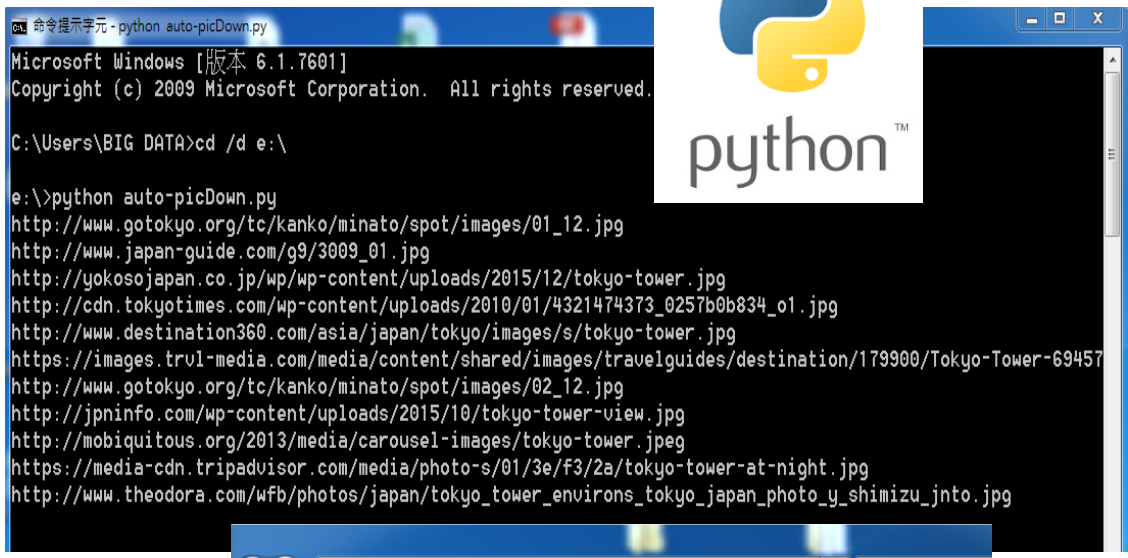
http://www.gotokyo.org/tc/kanko/minato/spot/images/02_12.jpg
http://www.gotokyo.org/tc/kanko/minato/spot/images/01_12.jpg
https://upload.wikimedia.org/wikipedia/commons/e/ed/TaroTokyo20110213-TokyoTower
http://travelheadline.com/wp-content/uploads/2014/01/241460_12984258180chX.jpg
http://travelheadline.com/wp-content/uploads/2014/01/1_edited1.jpg
http://pic.pimg.tw/handazyuku/1382187426-2482854366.jpg
https://fastjapan.com/en/wp-content/uploads/2016/01/shutterstock_129611225.jpg
http://b.share.photo.xuite.net/twgogo7/1be6c1b/9757958/432214805_m.jpg
http://www.wall001.com/human/tokyo-tower_01/mxxx01/t-tower-228.jpg
http://farm5.static.flickr.com/4048/4646822023_4804fd47ea_o.jpg
http://sib.tw/article_vw/2015042901/007.jpg
http://farm5.static.flickr.com/4017/4651115794_689bd15455_o.jpg
http://www.backpackers.com.tw/forum/gallery/images/97962/1_DSC05922.JPG
http://3.share.photo.xuite.net/yoimg/13d7ece/4671419/194688019_m.jpg
http://pic.pimg.tw/handazyuku/1382187427-699339213.jpg
http://farm9.static.flickr.com/8037/8032928423_ac7e4deee2_o.jpg
http://images2.gamme.com.tw/news2/2012/52/23/p5_Wn6Scj5_Z.jpg
http://farm9.static.flickr.com/8458/8035512745_f2e9fbb2f3_o.jpg
http://cdn2.ettoday.net/images/592/d592675.jpg
https://farm2.staticflickr.com/1632/25761356326_cfa49d2d19_b.jpg
http://pic.pimg.tw/joella/1338436610-3571660953.jpg
http://farm6.static.flickr.com/5018/5428439899_0f9fda1ecc_o.jpg
http://img103.mypsd.com.cn/20130108/1/Mypsd_22847_20130108164501000 4B.jpg
http://pic.pimg.tw/ra50130/1430743787-2385792713_n.jpg
http://pic.pimg.tw/uniquevera/4bd048f6d4c0d.jpg
https://i.ytimg.com/vi/apXtvJvF9Lo/maxresdefault.jpg
http://pic.pimg.tw/barbara0726/1366633624-2309163822.jpg
http://www.backpackers.com.tw/forum/gallery/images/217553/1_東京鐵塔S130.jpg

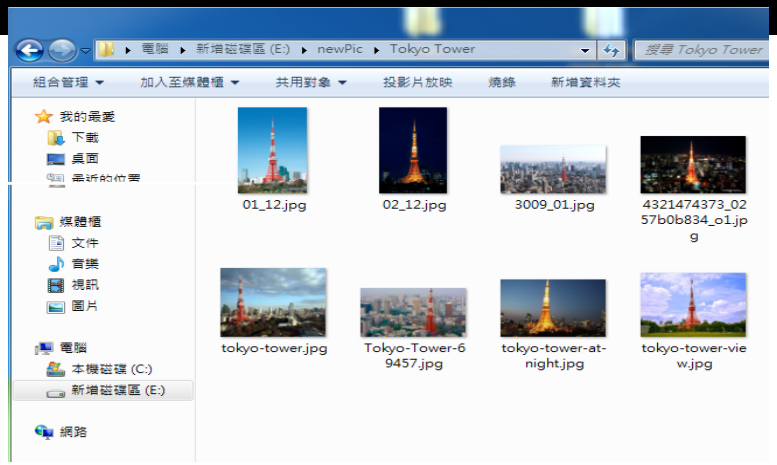
圖片下載與清理

Image URL List

http://www.gotokyo.org/tc/kanko/minato/spot/images/02_12.jpg
http://www.gotokyo.org/tc/kanko/minato/spot/images/01_12.jpg
https://upload.wikimedia.org/wikipedia/commons/e/ed/TaroTokyo20110213-TokyoTower
http://travelheadline.com/wp-content/uploads/2014/01/241460_12984258180chX.jpg
http://travelheadline.com/wp-content/uploads/2014/01/1_edited1.jpg
http://pic.pimg.tw/handazyuku/1382187426-2482854366.jpg
https://fastjapan.com/en/wp-content/uploads/2016/01/shutterstock_129611225.jpg
http://b.share.photo.xuite.net/twgogo7/1be6c1b/9757958/432214805_m.jpg
http://www.wall001.com/human/tokyo-tower_01/mxxx01/t-tower-228.jpg
http://farm5.static.flickr.com/4048/4646822023_4804fd47ea_o.jpg
http://sib.tw/article_vw/2015042901/007.jpg
http://farm5.static.flickr.com/4017/4651115794_689bd15455_o.jpg
http://www.backpackers.com.tw/forum/gallery/images/97962/1_DSC05922.JPG
http://3.share.photo.xuite.net/yoimg/13d7ece/4671419/194688019_m.jpg
http://pic.pimg.tw/handazyuku/1382187427-699339213.jpg
http://farm9.static.flickr.com/8037/8032928423_ac7e4deee2_o.jpg
http://images2.gamme.com.tw/news2/2012/52/23/p5_Wn6Scj5_Z.jpg
http://farm9.static.flickr.com/8458/8035512745_f2e9fbb2f3_o.jpg
http://cdn2.ettoday.net/images/592/d592675.jpg
https://farm2.staticflickr.com/1632/25761356326_cfa49d2d19_b.jpg
http://pic.pimg.tw/joella/1338436610-3571660953.jpg
http://farm6.static.flickr.com/5018/5428439899_0f9fda1ecc_o.jpg
http://img103.mypsd.com.cn/20130108/1/Mypsd_22847_20130108164501004B.jpg
http://pic.pimg.tw/ra50130/1430743787-2385792713_n.jpg
http://pic.pimg.tw/uniquevera/4bd048f6d4c0d.jpg
https://i.ytimg.com/vi/apXtvJvF9Lo/maxresdefault.jpg
http://pic.pimg.tw/barbara0726/1366633624-2309163822.jpg
http://www.backpackers.com.tw/forum/gallery/images/217553/1_東京鐵塔S130.jpg







- 可突破Google一次下載100張圖片的限制

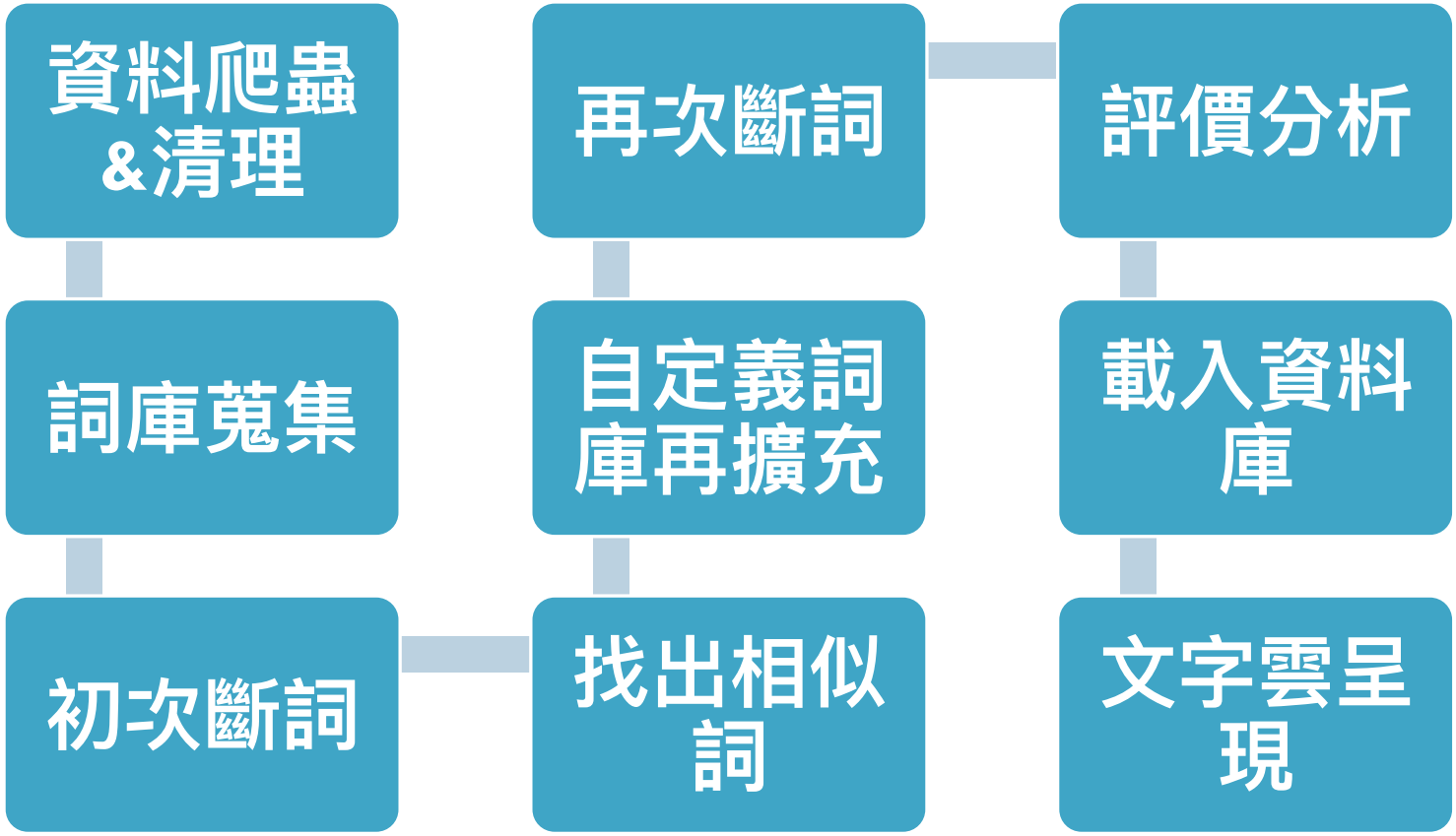
旅行社評論與文字探勘

李鈺祥

文字探勘—功能簡介

- 負責功能：旅行社評價分析
- 用途：
  - 選擇旅行社的評估參考
  - 行程規劃之參考指標

文字探勘—處理流程



文字探勘—資料爬蟲

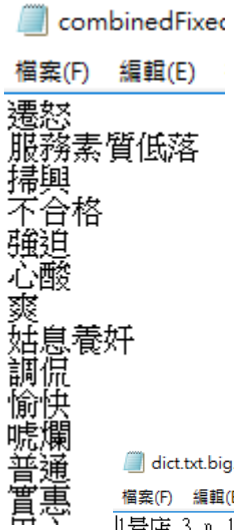
- 資料來源：
  - PTT(旅行社板)
  - 背包客棧
- 資料數量：
  - 共 13,410 篇
- 資料年份：
  - 自2011 ~ 至今

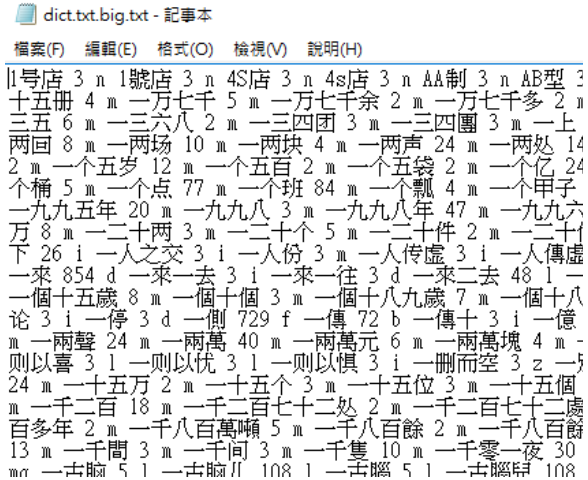
文字探勘—資料清理

- 使用語言：Python
- 去除標的物：
  - 去除不要的html標籤
  - 去除不要的文字內容
- 透過：
  - Python中的函式庫
  - 正規表示法

文字探勘—詞庫蒐集

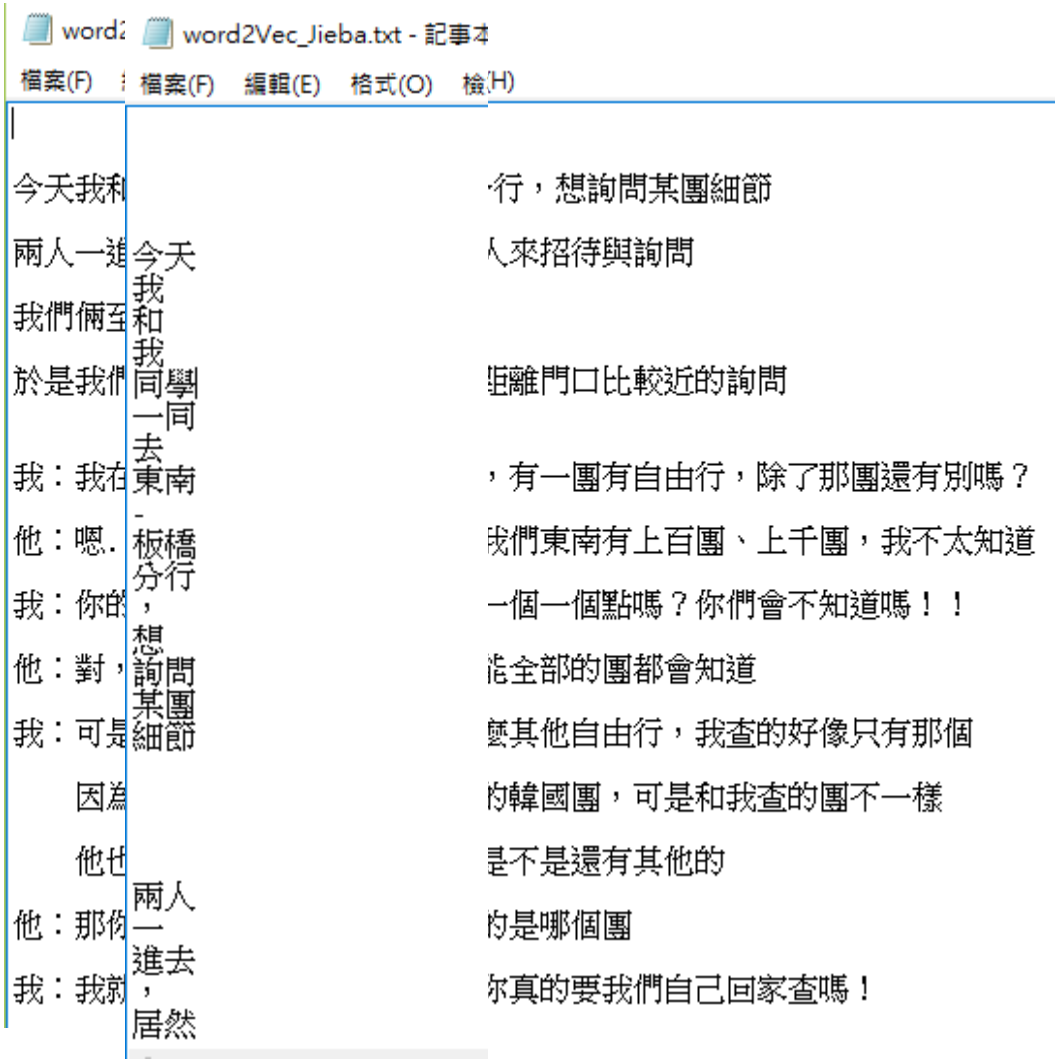
- 詞庫來源：
  - Jieba內建繁體詞庫
  - 萌典詞庫
  - 自定義詞庫
- 用途：
  - 初次斷詞之前處理
  - 提升斷詞精準度
  - 提升Word2Vec之訓練成效





文字探勘—初次斷詞

- 使用工具：Jieba
- 用途：
  - Word2Vec之前處理
- 斷詞範疇：
  - 涵蓋PTT與背包客棧
  - 不分各家旅行社
  - 共 13,410 篇文章



文字探勘—找出相似詞

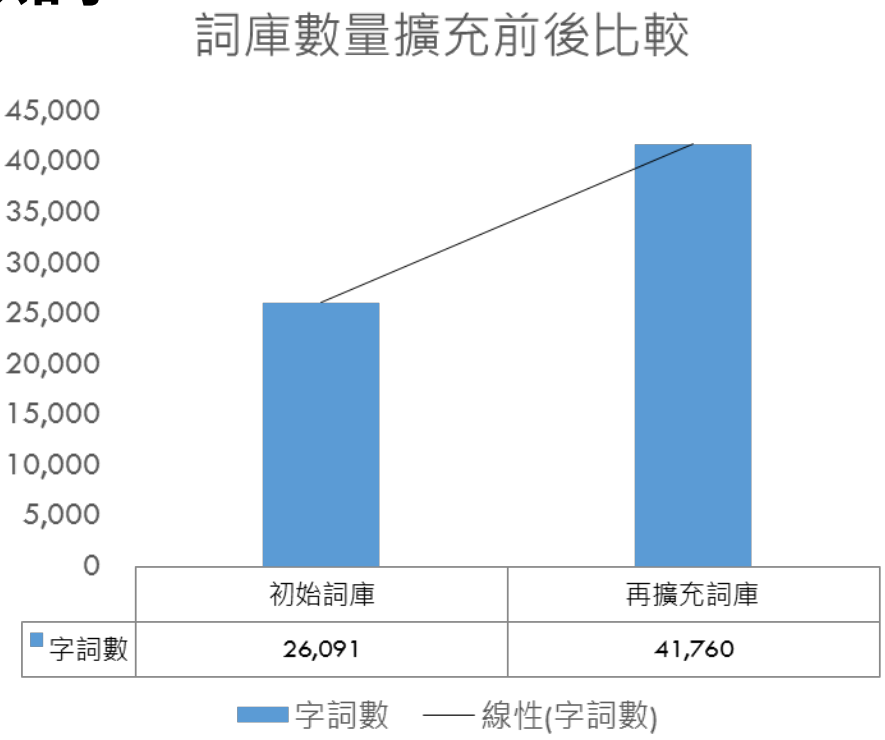
- 使用工具：Word2Vec
- 用途：
  - 找出相似並且可能遺漏的字詞
- 投入訓練量：
  - 751,490 句 / 訓練單位

```
'''
這邊是讀取訓練結果，可以不用再重新訓練。
使用most_similar這個方法做字詞的查詢，並且顯示前N個相
'''
load_model = Word2Vec.load(r'D:\BigData\Python\Works
for name, weight in load_model.most_similar(u'不爽',
    print name, weight
```

```
不高興 0.823772668839
貪小便宜 0.813368439674
嘸爛 0.808220982552
不划算 0.806670665741
被騙 0.805783569813
活該 0.805532753468
笨蛋 0.803521513939
很慘 0.800624847412
想想 0.800486445427
得了 0.798668861389
難看 0.796682596207
很煩 0.839768919613
好笑 0.837970972061
```

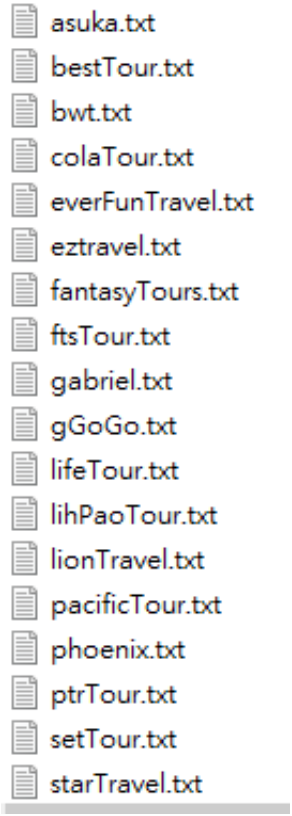
文字探勘—自定義詞庫再擴充

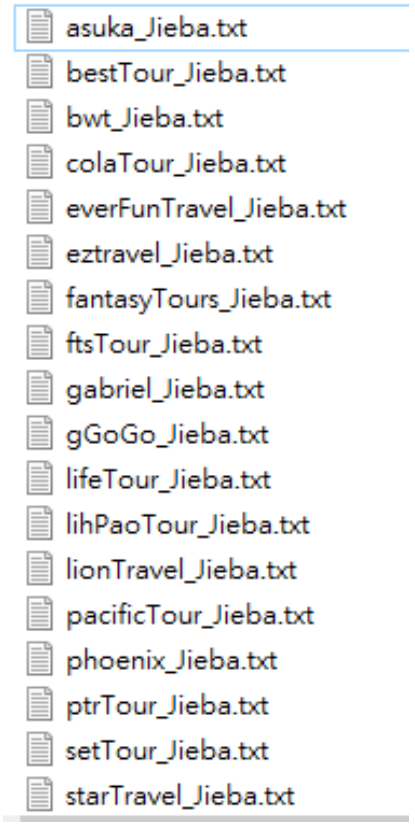
- 遺漏詞來源：
  - Word2Vec訓練出的相似詞
- 作法：
  - 篩選出符合實際需求之遺漏詞
  - 並建立於原有詞庫中
- 數量差異：
  - 擴充前詞量 **26,091**字
  - 擴充詞後量 **41,760**字



文字探勘—再次斷詞

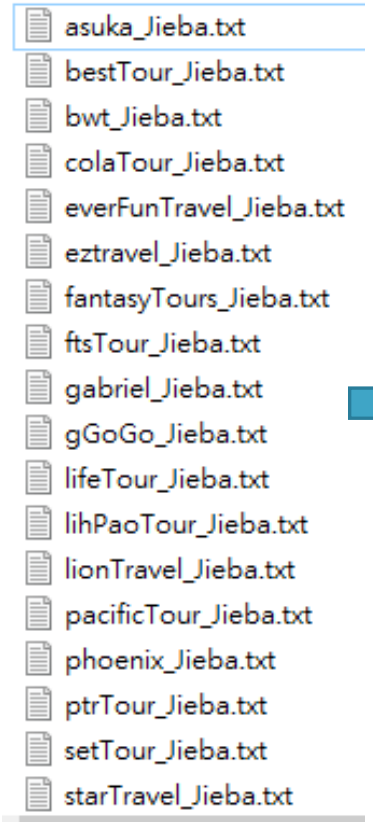
- 使用工具：Jieba
- 用途：
  - 評價分析前處理
- 斷詞範疇：
  - 涵蓋PTT與背包客棧
  - 分為20家旅行社
  - 共 13,410 篇文章

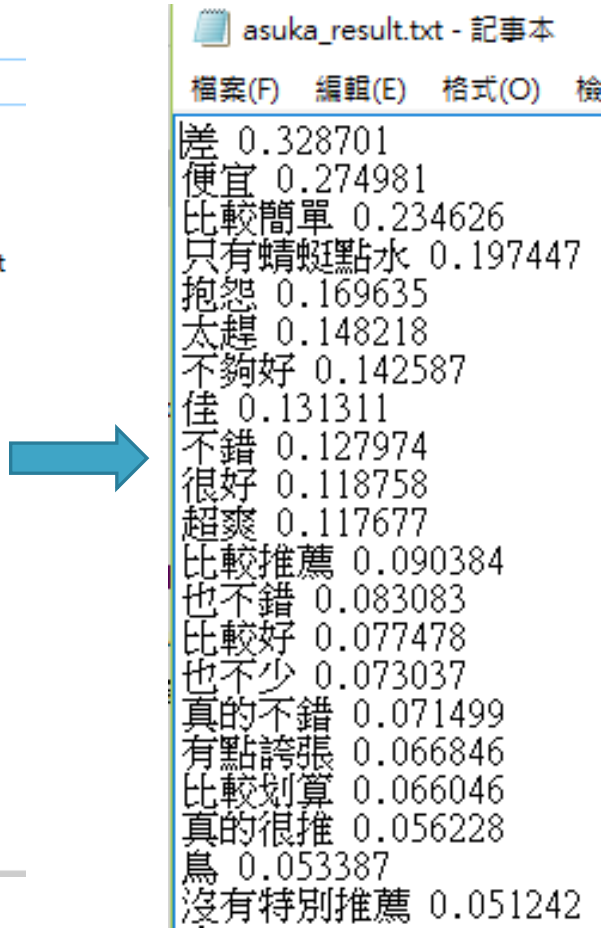




文字探勘—評價分析

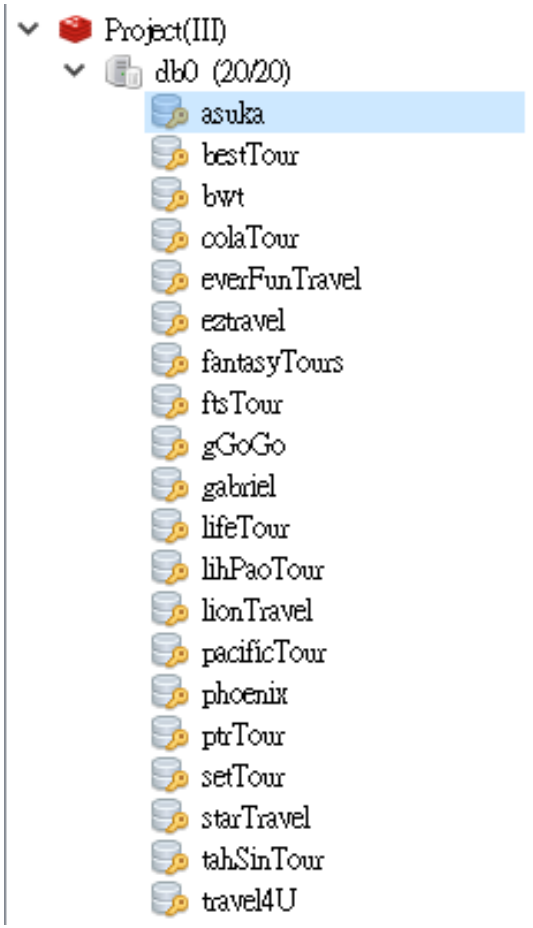
- 使用方法：
  - 共現詞頻分析
- 用途：
  - 找出符合該旅行社之真實評價
  - 提供使用者在挑選旅遊方案的參考





文字探勘—載入資料庫

- 使用資料庫：Redis
- 儲存型態：
  - Hash (Key/Value of 2 layers)
- 儲存資料：
  - 20家旅行社之評價分析
- 用途：
  - 為前端網頁文字雲之呈現資料



圖片辨識

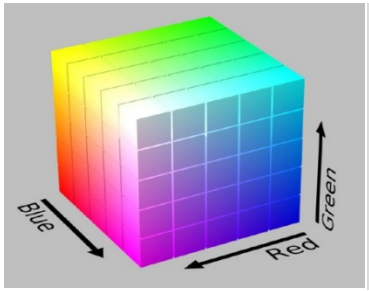
倪裕程、鄭云鈞

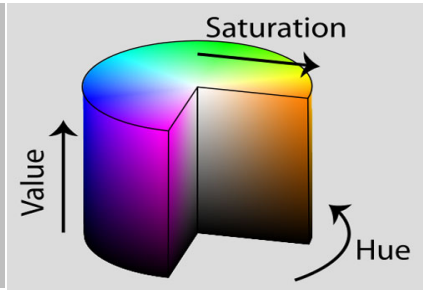
圖片辨識—圖片資料狀態

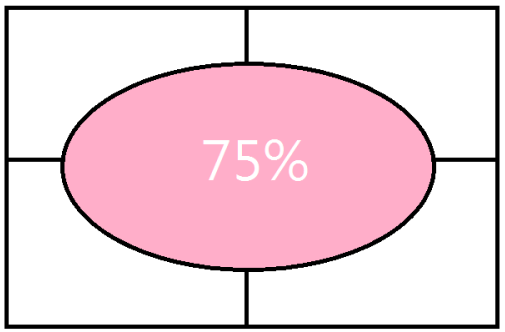
- 景點圖片比對遇到的問題
  - 日夜
  - 季節
  - 建築物

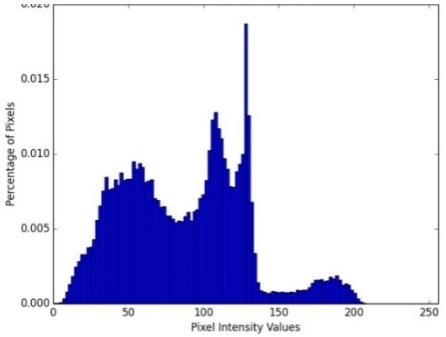
圖片辨識—顏色辨識

- 色塊直方
  - 將圖片從RGB色彩轉為
    - H (色相)
    - S (飽和度)
    - V (明度)
- 將圖片分成五大區域
  - 中間區域佔75%
- 分別針對各區域計算
  - 顏色直方圖









圖片辨識—特徵值選取

- 計算特徵值
  - 每個區塊都會分成H(8)S(12)V(3)
  - 分別有288個每個顏色出現的累積值
  - 有5個區塊，共1440個累積值做為特徵值

dataset\100000.jpg	7.15E-06	4.17E-05	3.57E-06	1.19E-06	0	0	0	0.995165	0	0	0	0	0	0	0	0
dataset\100001.jpg	1.72E-05	3.51E-05	1.24E-05	0	0	0	0	0.95551	0	2.06E-06	0	1.24E-05	0	0	0	0

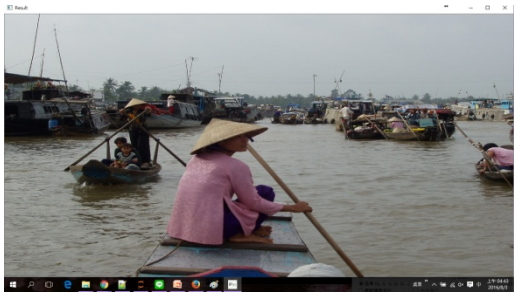
圖片檔名

共儲存有1440個累積值

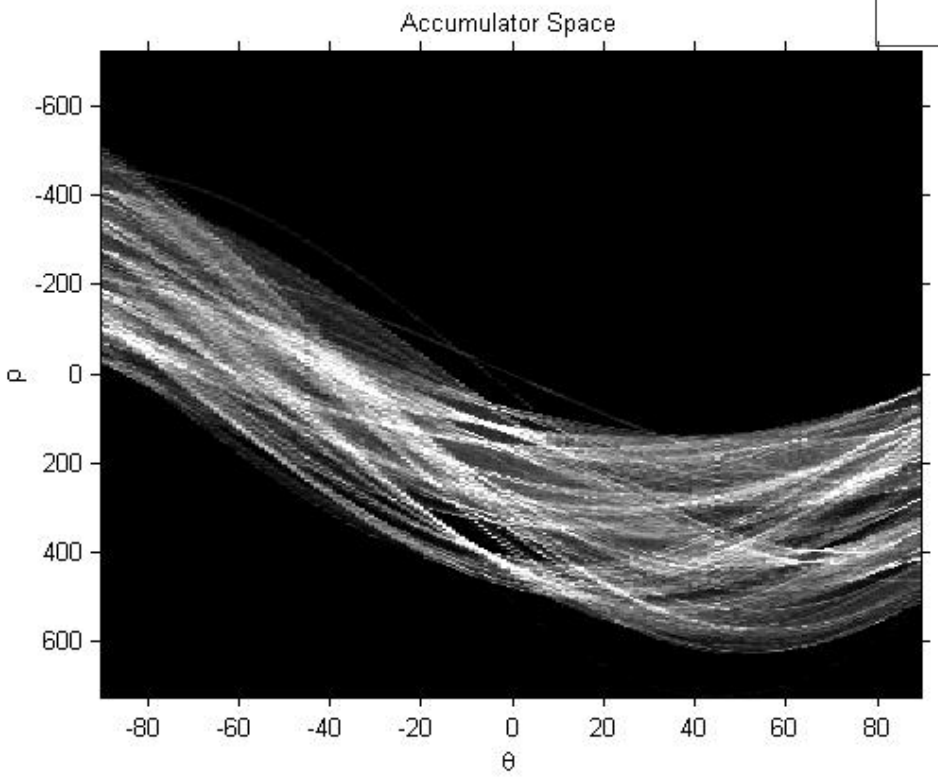
圖片辨識—顏色辨識範例

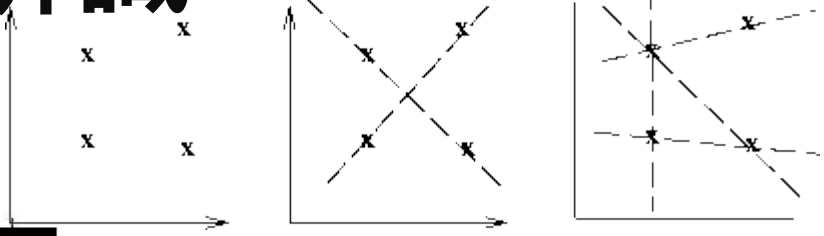
query

result

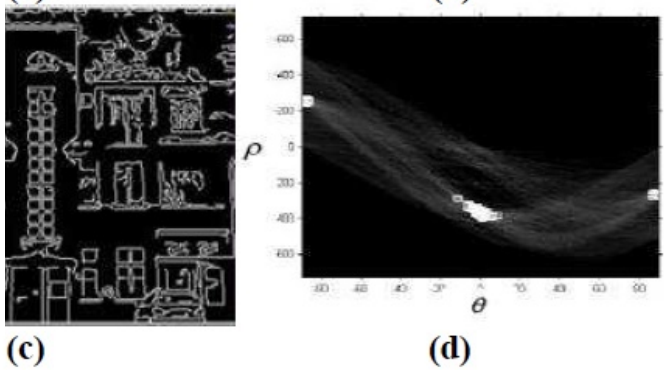


圖片辨識—形狀辨識





(a) (b)



(c) (d)

圖片辨識—特徵值選取

- 計算特徵值
  - The peak percentage 從0至179度
  - The distance ratio feature 從0至179度
  - 共360個維度

dataset\100301.jpg	0.001548	0.019436	0.00479	0.000945	0.022681	0.007074	0.000268	0.009835	0.004457	8.31E-05	0.003626	0.00101	9.23E-06	0.000357	3.69E-05	6.16E-06
dataset\100400.jpg	0.000411	0.000776	0.262822	8.96E-05	2.24E-05	4.48E-05	4.85E-05	1.87E-05	7.47E-06	8.21E-05	2.24E-05	3.36E-05	3.36E-05	1.49E-05	0	1.87E-05

圖片檔名

共儲存有360特徵值

圖片辨識—特徵值選取

- **The peak percentage**

The peak percentage is defined by

$$p_\theta = \frac{n_\theta}{n}$$

where n_θ represents the number of Hough peaks in the θ^{th} bin and n denotes the total number of Hough peaks in the Hough parameter space.

圖片辨識—特徵值選取

- **The distance ratio feature**

Also the distance ratio feature is defined by

$$d_\theta = \frac{\sqrt{\sum_{j=1}^{n_\theta} (\rho_j)^2}}{C_\theta}$$

where ρ_j represents the distance from the θ^{th} Hough cell to the horizontal line and C_θ signifies the weighted-centroid of the θ^{th} cell defined by

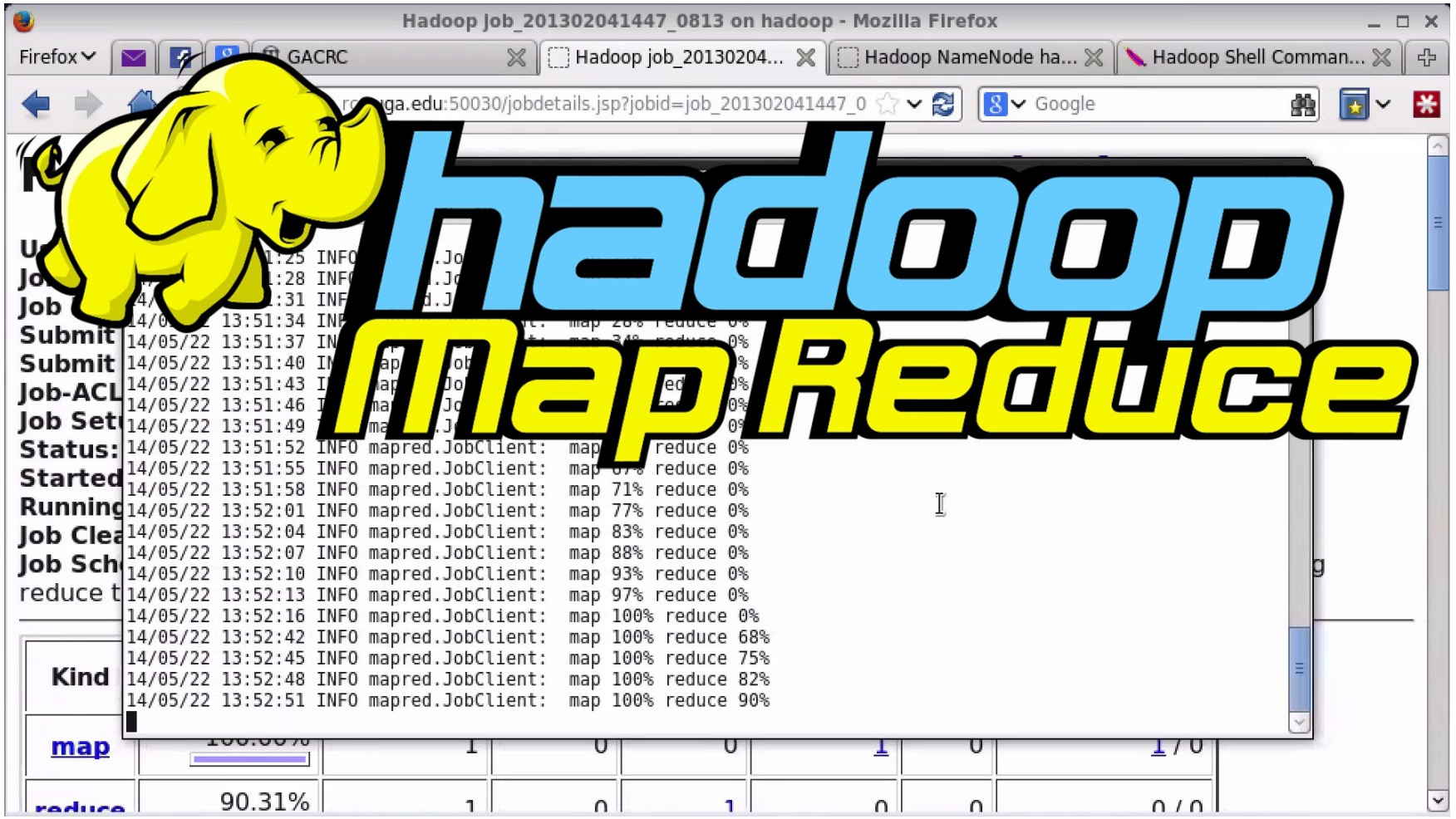
$$C_\theta = \frac{\sum_{j=1}^{n_\theta} w_j \cdot \rho_j}{\sum_{j=1}^{n_\theta} w_j}$$

with w_j representing the number of points in the edge map that form the j^{th} Hough peak. The distance ratio vectors of the

圖片辨識—演算法選擇

- 色塊直方圖相似比對
  - 解決日夜或是季節顏色問題
  - 減少Hough Line特徵值比對運算時間

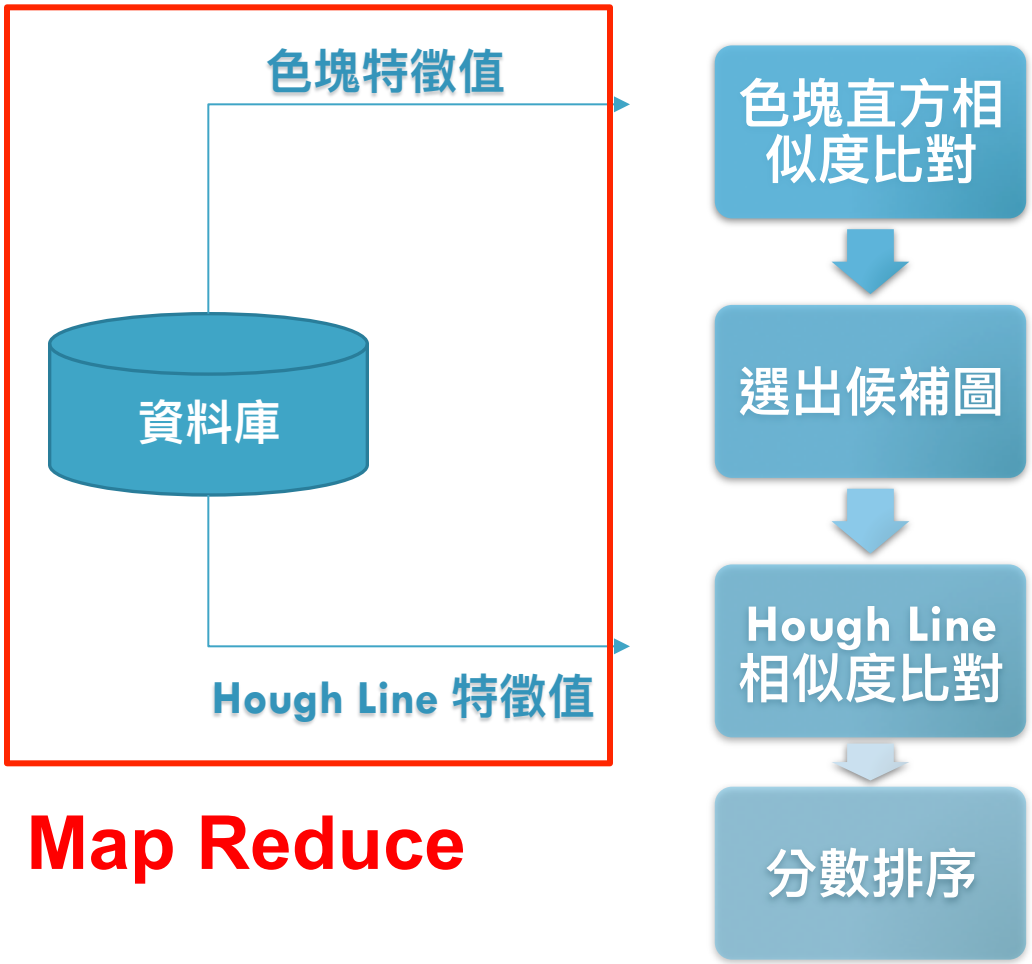
- Hough Line 特徵值相似度比對
  - 解決建築物形狀比對
  - 解決色塊搜尋到不相關圖片


Hadoop job_201302041447_0813 on hadoop - Mozilla Firefox
Hadoop job_20130204...
Hadoop NameNode ha...
Hadoop Shell Comman...
GACRC
Google
hadoop
MapReduce
14/05/22 13:51:52 INFO mapred.JobClient: map reduce 0%
14/05/22 13:51:55 INFO mapred.JobClient: map reduce 0%
14/05/22 13:51:58 INFO mapred.JobClient: map 71% reduce 0%
14/05/22 13:52:01 INFO mapred.JobClient: map 77% reduce 0%
14/05/22 13:52:04 INFO mapred.JobClient: map 83% reduce 0%
14/05/22 13:52:07 INFO mapred.JobClient: map 88% reduce 0%
14/05/22 13:52:10 INFO mapred.JobClient: map 93% reduce 0%
14/05/22 13:52:13 INFO mapred.JobClient: map 97% reduce 0%
14/05/22 13:52:16 INFO mapred.JobClient: map 100% reduce 0%
14/05/22 13:52:42 INFO mapred.JobClient: map 100% reduce 68%
14/05/22 13:52:45 INFO mapred.JobClient: map 100% reduce 75%
14/05/22 13:52:48 INFO mapred.JobClient: map 100% reduce 82%
14/05/22 13:52:51 INFO mapred.JobClient: map 100% reduce 90%
Kind
map
90.31%

分散式運算應用

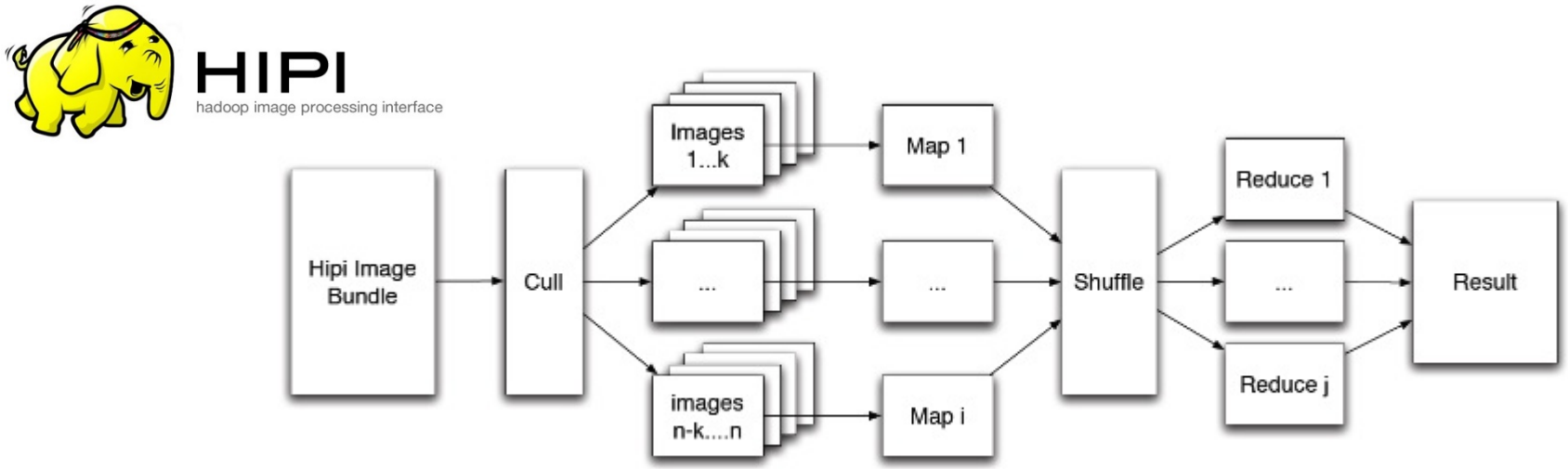
倪裕程

圖片辨識—分散式應用


色塊特徵值
資料庫
Hough Line 特徵值
Map Reduce
色塊直方相似度比對
選出候補圖
Hough Line相似度比對
分數排序

圖片辨識—資料庫特徵值

- 為何使用Hadoop map reduce
  - 行程景點眾多，網路爬取圖片量大
  - 藉由Hadoop進行分散式儲存及運算
  - 可節省利用單機運算的時間



圖片辨識—資料庫特徵值

```
@Override
public void map(HipiImageHeader header, FloatImage image,
		Context context) throws IOException, InterruptedException {

	String output = null;

	if (header == null) {
		output = "Failed to read image header.";
	} else if (image == null) {
		output = "Failed to decode image data.";
	} else {
		System.loadLibrary(Core.NATIVE_LIBRARY_NAME);
		Mat images = this.convertFloatImageToOpenCVMat(image);
		houghLineTransform houghlineChar = new houghLineTransform(
				images);
		Map<Double, Double> DistanceRatio = houghlineChar
				.getDistanceRatio();
		Map<Double, Double> Percentage = houghlineChar.getPercenta

		String filename = header.getMetaData("filename");
		output = filename;

		// Write in to map
		for (double key : Percentage.keySet()) {
			output = output + "," + Percentage.get(key);
		}
		for (double key : DistanceRatio.keySet()) {
			output = output + "," + DistanceRatio.get(key);
		}

	}

	context.write(new IntWritable(1), new Text(output));
}
```

```
@Override
public void map(HipiImageHeader header, FloatImage image,
		Context context) throws IOException, InterruptedException {


	String output = null;

	if (header == null) {
		output = "Failed to read image header.";
	} else if (image == null) {
		output = "Failed to decode image data.";
	} else {
		// Load OpenCV native library
		try {
			System.loadLibrary(Core.NATIVE_LIBRARY_NAME);
		} catch (UnsatisfiedLinkError e) {
			System.err.println("Native code library failed to load.\n" + e
					+ Core.NATIVE_LIBRARY_NAME);
			System.exit(1);
		}
		// convert to OpenCV Mat
		Mat images = this.convertFloatImageToOpenCVMat(image);
		// Hsv hist feature
		MatOfInt bin = new MatOfInt(8, 12, 3);
		ColorDescriptor cd = new ColorDescriptor(bin);
		List<Mat> feature = cd.describe(images);
		String name = header.getMetaData("filename");
		output = name;
		for (int i = 0; i < 5; i++) {

			for (int j = 0; j < 288; j++) {
				String elementOfMat = "" + feature.get(i).get(j, 0)[0];
				output = output + "," + elementOfMat;
			}
		}

	}

	context.write(new IntWritable(1), new Text(output));
}
```

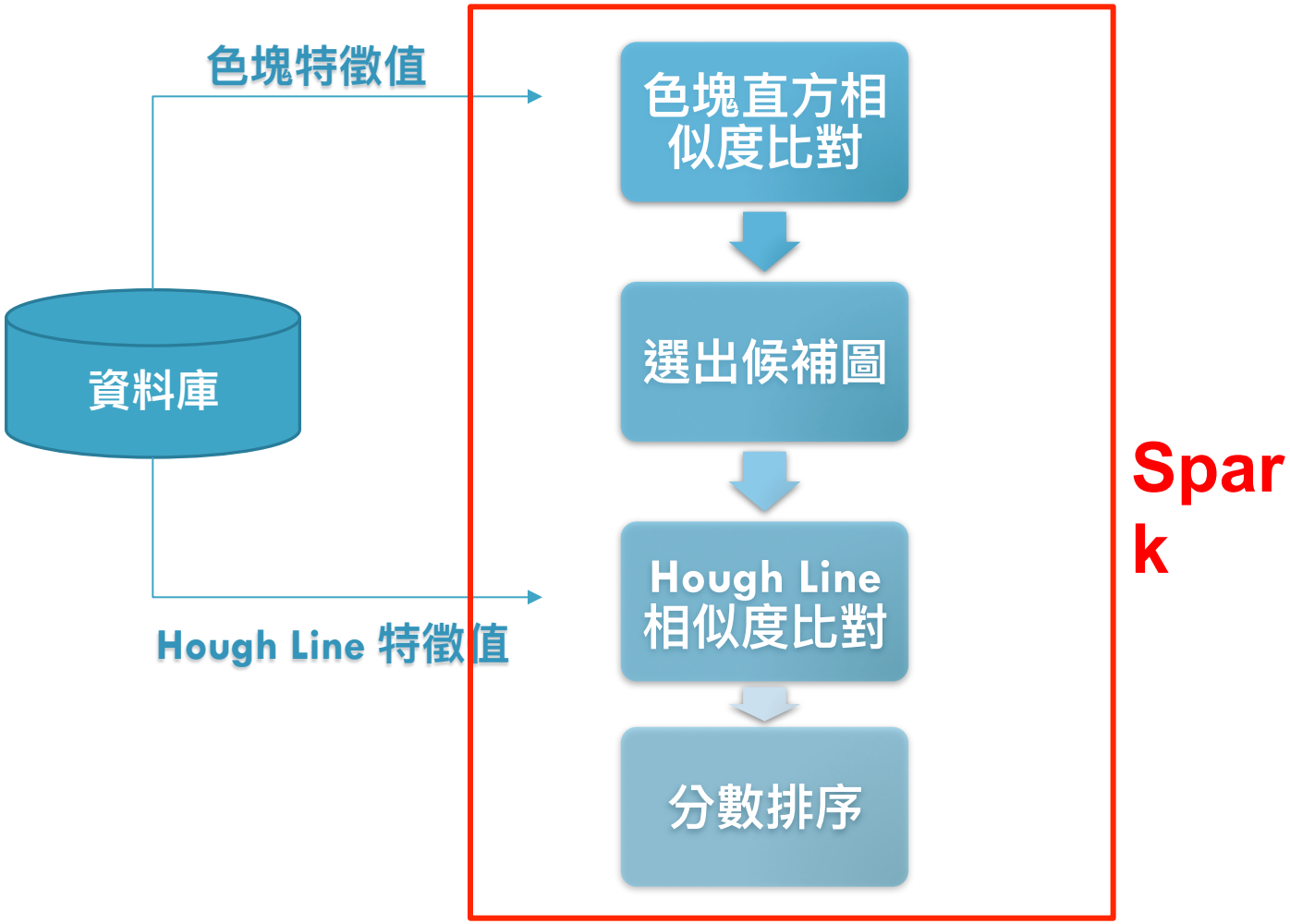
圖片辨識—特徵值提取時間差

- 以10張圖比較Hough line特徵值計算時間
  - 單機- 75s
  - Map reduce- 33s
- 以100張圖比較Hough line特徵值計算時間
  - 單機- 88m42s
  - Map reduce- 56m5s

圖片資料庫搜尋引擎

鄭云鈞

圖片辨識—流程與分散式應用



圖片辨識— SPARK

- 呼叫後可一直在記憶體中持續運行
- 在記憶體中運行計算，計算速度快
- 提供彈性分佈式數據集(RDD)
- 利用Spark計算被查詢的圖片及特徵值比對

```
    bins = (8, 12, 3)
    query = url_to_image(image_seqfile_path)
    queryFeatures = describe(query,bins)

    text = taxiFile.map(lambda k: k.split(",")).map(lambda k: colorfeature(k,queryFeatures)).sortByKey().map(lambda k: k[1])
    colorquery = text.take(105)

    houghquery = hough(query)
    result = taxiFile.map(lambda k: k.split(",")).filter(lambda k: k[0].encode('utf-8') in colorquery).map(lambda
k:rankingFunction(houghquery,k)).reduceByKey(max).map(lambda k: (k[1],k[0])).map(lambda k: k[1])

    final = result.take(3)

    connection = redis.Redis(host = '36.236.48.175', port = 6379, db = 4)
    connection.flushdb()
    connection.lpush('query', *final)
```

圖片辨識—色塊直方相似度比對

- 先進行圖片色塊比對
  - 計算被查詢圖片與資料庫圖片的色塊卡方相似度

```
text = taxiFile.map(lambda k: k.split(",")).map(lambda k: colorfeature(k,queryFeatures)).sortByKey().map(lambda k: k[1])
```

$$\chi^2(x,y)=\sqrt{\sum_{i=1}^{p}\left(\frac{x_i-E(x_i)}{E(x_i)}\right)^2+\sum_{i=1}^{p}\left(\frac{y_i-E(y_i)}{E(y_i)}\right)^2}$$

```
(2.1774022620351747e-07, '123300.jpg'),
(6.4339739536879405e-07, '128300.jpg'),
(2.4936570199980801e-06, '119602.jpg'),
(3.7364309595940535e-06, '145300.jpg'),
(8.0154014764149456e-06, '141601.jpg'),
(9.6966978644122727e-06, '122700.jpg'),
```

  - 數值越小，相似度越高
  - 只要找出前105張圖來做Hough Line比對

圖片辨識—搜尋範例

Query

候補圖

圖片辨識—HOUGH LINE 相似度比對

- 再進行Hough Line形狀比對

```
result = taxiFile.map(lambda k: k.split(",")).filter(lambda k: k[0].encode('utf-8') in colorquery).map(lambda
k:rankingFunction(houghquery,k)).reduceByKey(max).map(lambda k: (k[1],k[0])).map(lambda k: k[1])
```

The m^{th} circular correlation between $\mathbf{p}^q$ and $\mathbf{p}^k$ is defined as

$$s_p^k(m) = \frac{\sum_{j=1}^{J-m}(p_j^q p_{j+m}^k) + \sum_{j=J-m+1}^{J}(p_j^q p_{j+m-J}^k)}{\sqrt{\sum_{j=1}^{J}(p_j^q)^2}\sqrt{\sum_{j=1}^{J}(p_j^k)^2}}$$

which is normalized between 0 and 1. We calculate the maximum circular correlation $\max_m\{s_p^k(m)\}$ between $\mathbf{p}^q$ and $\mathbf{p}^k$, and use to retrieve the candidate images.

```
u'137901.jpg',
u'0.0830464',
u'0.0272187',
u'0.38288',
u'0.0509615',
u'0.0277534',
u'0.0314432',
u'0.0886613',
u'0.0373789',
u'0.00251332',
u'0.220584',
u'0.015882',
```

Similarly, the m^{th} circular correlation between $\mathbf{r}^q$ and $\mathbf{r}^k$ is defined as

$$s_r^k(m) = \frac{\sum_{j=1}^{J-m}(r_j^q r_{j+m}^k) + \sum_{j=J-m+1}^{J}(r_j^q r_{j+m-J}^k)}{\sqrt{\sum_{j=1}^{J}(r_j^q)^2}\sqrt{\sum_{j=1}^{J}(r_j^k)^2}}$$

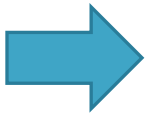
which is normalized between 0 and 1. Also we calculate the maximum circular correlation $\max_m\{s_r^k(m)\}$ between $\mathbf{r}^q$ and $\mathbf{r}^k$, and use to retrieve the candidate images.

```
('137901.jpg', 0.775079348140713),
('147500.jpg', 0.4231559767303664),
('141801.jpg', 0.5377062084315848),
('134503.jpg', 0.5447344501367279),
('148601.jpg', 0.3893849151577147),
('118601.jpg', 0.47478382659779406),
('117801.jpg', 0.6186679239688069),
('123402.jpg', 0.6685508610019886),
```

圖片辨識—
HOUGH LINE 相似度比對

```
result = taxiFile.map(lambda k: k.split(",")).filter(lambda k: k[0].encode('utf-8') in colorquery).map(lambda
k:rankingFunction(houghquery,k)).reduceByKey(max).map(lambda k: (k[1],k[0])).map(lambda k: k[1])
```

```
('137901.jpg', 0.775079348140713),
('147500.jpg', 0.4231559767303664),
('141801.jpg', 0.5377062084315848),
('134503.jpg', 0.5447344501367279),
('148601.jpg', 0.3893849151577147),
('118601.jpg', 0.47478382659779406),
('117801.jpg', 0.6186679239688069),
('123402.jpg', 0.6685508610019886),
```



```
'118201.jpg',
'132700.jpg',
'147500.jpg',
'112001.jpg',
'113402.jpg',
'119600.jpg',
'145300.jpg',
```

- 數值越大，表示建築物形狀越相近
- 選出前三個數值最大的景點作行程推薦

圖片辨識—搜尋範例

Query

候補圖

圖片辨識—行程推薦

- 將Spark查詢結果從Redis資料庫叫出

```
The_Tsukiji_Market
Daigoji_Temple
Ginkaku_ji_Temple
```

- 從Redis資料庫拉出相對應相關行程

```
['GFG0000002920']
['469_1', '469']
['VDR0000001838_NOO061305BR6B', 'VDR0000001838_NOO061005CI6EB', '50499', '50398', '43408', '41639', '39404']
```

- 從使用者選擇的類型計算出最適合行程
  - 計算每個行程中特定類型所占分數，找出最大值
  - 若有同分，則再比較拉手分數
  - 若無選擇喜歡類型，只比較拉手分數

圖片辨識—行程推薦

- 三種結果訊息呈現
  - 該行程所屬旅行社

```
setTour
```

  - 該行程有哪些景點

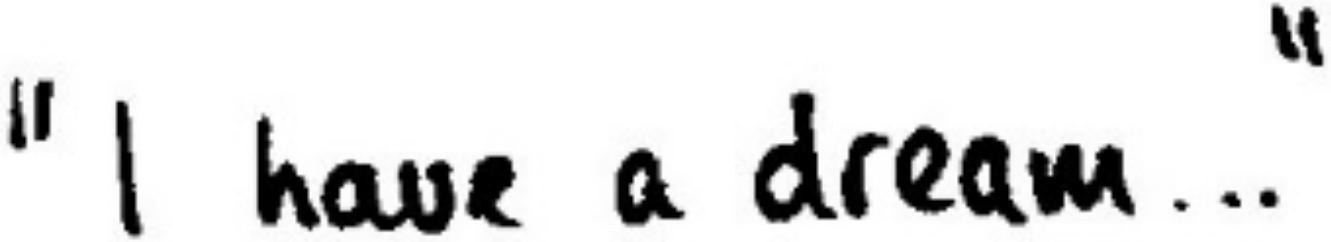
```
Fuji_Hakone_Izu_National_Park
Shin_Nakamise_Shopping_Street
Venetian_Glass_Museum
Senso_ji_Temple
Hakone_Pirate_Ship
Diver_City_Tokyo_Plaza
```

  - 該行程每種類別所佔比例

```
景點與地標 14
自然與公園 3
購物 2
水上樂園和遊樂場 2
旅客資源 1
博物館 1
休閒與娛樂 1
飲食 0
```

未來展望

- 增加風景辨識功能
- 增加自由行功能
- 文字探勘前處理更完善
- 景點行程類型前處理更完善

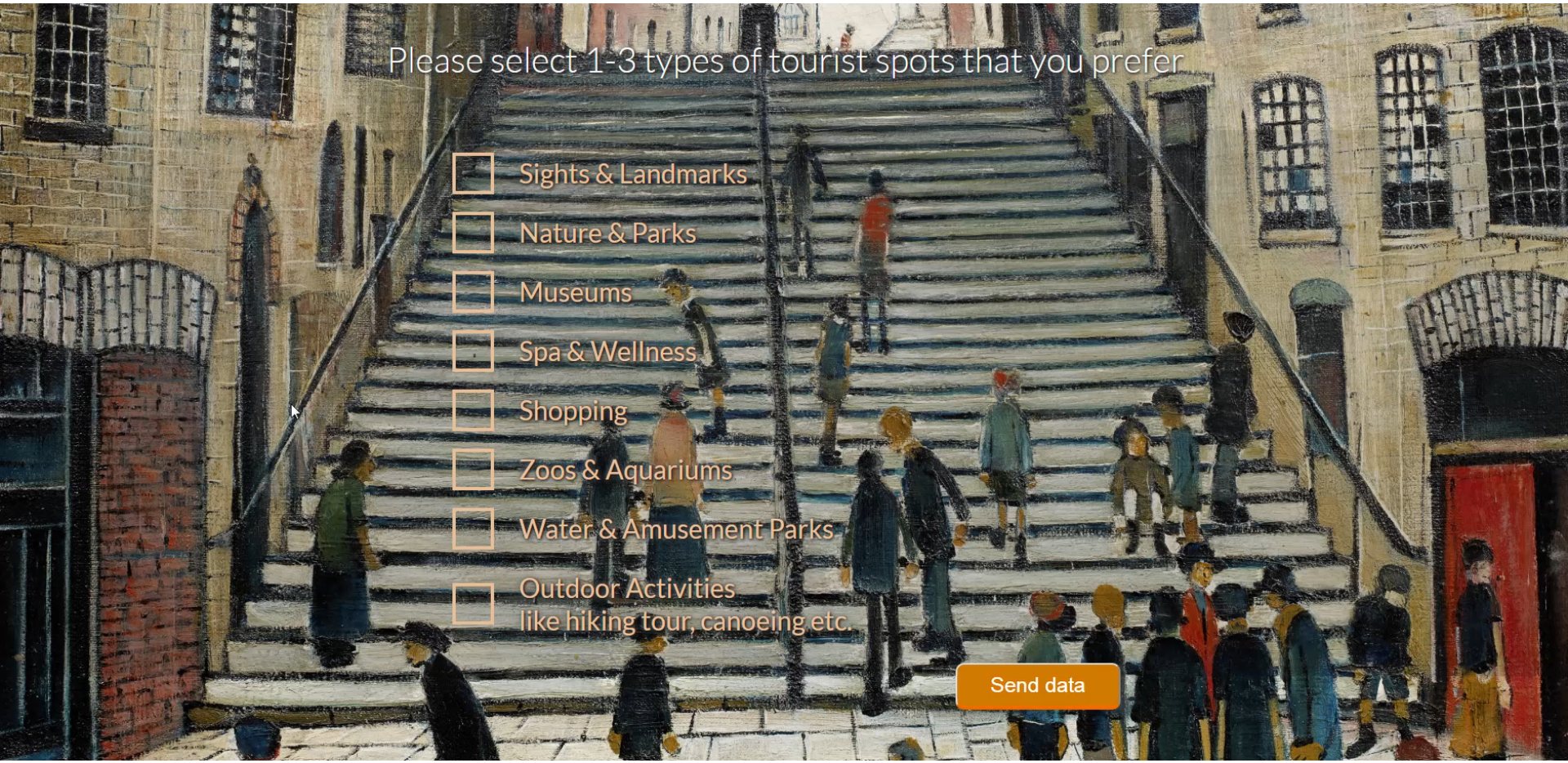


Q&A時間

Thanks for watching

BACKUP


電腦
CCleaner
HDP_2.4_v...
Building_I...
failDownl...
Imgur_The most awes...
設定方式.txt
RedisDesk...
id_rsa
Building_i...
Naraijuku...
新文字文件.txt
test Imgur.html
x64 3.3.0
AnyDesk
pse.exe - 捷徑
totalAtricl...
racle VM rtualBox
WinSCP
Mware orkstati...
amViewer 11
travel
3_final.html
Naraijuku003.JPG - Windows 相片檢視器
檔案(F)
列印(P)
電子郵件(E)
燒錄(U)
開啟(O)


Please select 1-3 types of tourist spots that you prefer
Sights & Landmarks
Nature & Parks
Museums
Spa & Wellness
Shopping
Zoos & Aquariums
Water & Amusement Parks
Outdoor Activities
like hiking tour, canoeing etc.
Send data

非常專業
也不錯
很好
抱怨
很滿意
驚艷
佳
很漂亮
不錯
瑕疵
比較划算
相當不錯
真的好
再麻煩
比較好