The 2015 Conference on Computational Linguistics and Speech Processing
ROCLING 2015, pp. 180-195

基於 Web 之商家景點擷取與資料庫建置

Points of Interest Extraction from Unstructured Web

高霆耀　Ting-Yao Kao
國立中央大學資訊工程學系
Department of Computer Science and Information Engineering
National Central University
kao800208@gmail.com

莊秀敏　Hsiu-Min Chuang
國立中央大學資訊工程學系
Department of Computer Science and Information Engineering
National Central University
showmin1205@gmail.com

張嘉惠　Chia-Hui Chang
國立中央大學資訊工程學系
Department of Computer Science and Information Engineering
National Central University
chia@csie.ncu.edu.tw

摘要

隨著行動裝置的普及，區域搜尋成為了一項新興的熱門服務。然而區域搜尋要能提供完整的服務，必須要讓使用者能夠準確地搜尋到附近的興趣點(Point of Interest, POI)，如餐廳、旅館、巴士站、卡拉 OK、圖書館、藥局等包含食衣住行育樂的地點。為此我們要建構一個完整的 POI 資料庫供使用者查詢。另外由於網際網路的盛行，越來越多的使用者會在他們的部落格或是社交網路上分享旅遊經驗或是 POI 的資料，同時也有更多的商家或組織建立官方網頁，並且在網頁上詳細的介紹他們的資料。隨著這類型網頁的數量累積，整個網際網路成為了最大的 POI 資訊來源。

在本篇論文中我們提出一個基於 Web 資訊的 POI 建置系統，系統可以分為兩大部分，第一部分為包含地址網頁(Address-bearing Page, ABP)的爬取，目的在透過網頁中的地址找尋可能的 POI 以及可用來做為檢索的 POI 相關描述訊息。第二部分為 POI 擷取系統，透過條件隨機域(Conditional Random Field, CRF)作為學習演算法產生的中文組織名稱辨識模型及中文地址辨識模型，找出網頁中所有出現的地址和組織名稱，接著再將地址與組織名稱配對成 POI 資料，最後再為每一個 POI 擷取其相關資訊。

Abstract

With the increased popularity of mobile devices, local search has become a new popular service. Therefore, we need a powerful POI (Points of Interest) database to support local search. In recent years, the web has become the largest data source of POIs. With the prevalence of Internet, people will share their travel experience and information of POIs that they had been visited on social network, their blogs, and even check-in post. Besides, many companies and organizations publish their business on their own websites, resulting a large number of POIs.

In this paper, we propose a POI database construction system from the immense data of the Web. Our system consists of two parts: the query-based crawler, and the POI extraction system. The goal of query-based crawler is to collect address-bearing pages (ABP) from the web as address is a good indicator of POIs. The second part is POI extraction system. We use CRF (Conditional Random Field) to train a Chinese postal address recognition model and a Chinese organization recognition model. After the extraction of addresses and POI names from ABP with these two CRF models, we then leant a model to pair an address and a POI name as a POI. Finally, we extract POI associated information for each POI to construct a complete POI data.

關鍵詞：電子地圖、網路爬蟲、資訊擷取、POI 資料庫

Keywords: electronic map, web crawler, information extraction, POI database.

一、 緒論

電子地圖不僅是數位化後的地圖，因為不受限於有限的空間，可以根據瀏覽者的需求，整合其背後資料利用圖層疊加的特性對社會經濟資料進行標記與分析，所以產生了相當多新穎的服務，如買屋租屋搜尋、景點搜尋等等。另者，由於近年行動裝置的進步與普及，連帶使得行動定位與目的地導航成為一項新興的熱門服務，現今的電子地圖大多整合了以上的功能，提供了完整的適地性服務(Location-based Service)，使得地圖搜尋成為日常生活不可或缺的功能。

雖然電子地圖能夠提供給我們諸多的便利，但除了基本的地理資訊外，電子地圖還必須要仰賴其系統後方豐富且充沛的資料庫，才能更加突顯其功效。多數地理資料庫都是依靠人工編輯，但是要將所有的 POI 都使用人工的方式加入資料庫是一件耗時費力的事情，因此也限制了現今地點資料庫的地點數量與內容。然而在網際網路盛行的現今，雖然政府工業局或商業司有企業登記資料，但企業登記名稱與店家名稱往往不一致，例如嘟嘟房停車實由中興電工經營，因此即使有政府開放資料，商家 POI 資料仍然不夠完整，但是除了政府機構的網站，POI 亦經常伴隨其描述出現在其他網頁中，如連鎖商店的網頁、部落格的餐廳介紹及景點介紹，甚至於社群網站的打卡資訊等，這些網頁中或多或少都包含了 POI 的描述訊息，因此若能有效率地找到上述這些含有 POI 資料的網頁，並由程式自動將其擷取出可用的 POI 資料，便可有效地擴展資料庫的地點數量與內容。

根據 W3C 的定義，一個 POI 會包含許多資訊，像是名稱、位置、電話以及相關資訊等等，其中位置用於定位標記到地圖上，可用地址或經緯座標表示。由於地址的識別率較高相對其他 POI 資訊更容易擷取，因此本篇論文中，我們提出一個 POI 資料庫的建置系統，以地址擷取做為 POI 辨識策略，並且從包含地址的網頁中擷取與地址相對應的 POI 名稱和相關資訊，用來建立一個 POI 資料庫，提供 POI 搜尋服務，POI 資料範例如圖 一。

```
<?xml version="1.0" encoding="UTF-8"?>
- <data>
    <size>1</size>
    <address>97068花蓮縣花蓮市富國路134號</address>
    <title>狀元書局</title>
    <tel>03-8579655</tel>
    <cateory>事務文具</cateory>
    <type>書店</type>
    <longitude>121.5940187</longitude>
    <latitude>23.9896082</latitude>
    <abstract> 狀元書局 營業時間：星期一~星期日 早上 08：00 到 晚上 22：00 新聞文化、圖
    <http/>
  </data>
```

圖 一、POI 範例

圖 二、網頁中的 POI 相關資訊與雜訊

本系統包含三個模組。第一模組是網頁的爬取(Crawler)，我們首先以地址為關鍵字串來蒐集包含地址的網頁(Address-bearing Pages, ABP)，我們引入 Chang 等人在 2012 年[5]和 Lin 等人在 2014 年[10]所提出的兩種模型來從 ABP 中擷取出地址；第二個模組則是 POI 擷取模組，我們使用 Huang 等人在 2015 年[8]提出的中文組織名稱辨識模型來擷取 ABP 當中的 POI 名稱。最後我們會將辨識出的 POI 名稱以及地址組成許多筆 POI，並透過 POI 配對驗證模組將正確的 POI 資料放入資料庫當中。

另外，因為大多數使用者是由關鍵字或是類別反查商店在地圖上的位置，因此用以描述地址的相關資訊是否足夠，會大幅影響查詢系統的檢索效能，為此我們提出了 POI 相關資訊擷取模組，為每個 POI 擷取相關描述來解決此問題。如圖 二所示，網頁中包含許多地址的描述，但同時仍有許多與地址不相關的內容。在本篇論文中我們將應用中文組織名稱辨識模組來加強這類型網頁的地址相關資訊擷取。

本論文共分成五個章節，第一章為緒論，說明研究的動機與目的並簡單的介紹本篇論文；第二章為相關研究，介紹和本論文相關的研究；第三章為系統架構與研究方法，詳述如何從網路中找尋 ABP，並由程式自動擷取出 POI；第四章為實驗，評估系統的效能及 POI 資料的正確性；第五章為結論，總結本論文的貢獻。

二、 相關研究

近幾年來，由於網路上巨量資料的累積與行動裝置的普及，地理資訊檢索(Geographic Information Retrieval)以及區域搜尋開始受到重視。國際間地理資訊檢索領域的研究以 ACM SIGSpatial workshop on GIR 較負盛名，自 2004 年起收錄相關領域的研究報告，相關研究主題包括了地理資訊系統的發展模式、地理數據庫的存取與網路內容與多媒體的分析、基於文字與地理資訊系統整合的方法(如資訊擷取、自然語言處理、空間資料的索引與搜尋等)、以及地理術語的識別與時空(spatio-temporal)的概念。

另外則是從2008開始與WWW同時舉辦的Workshop on Location and the Web(LocWeb)，後續也在 CHI、IoT、CIKM 等會議舉行，某種層次來看，LocWeb 與 Web 的關係更為緊密。而國內研討會則以台灣地理資訊學會舉辦的研討會為主，主題包含了地理空間數據可視化、地理資訊系統技術發展與整合應用、開放資料與群眾外包(Crowdsourcing)、防救災與資通技術整合、以及自然環境資源管理與環境監測相關研究。

相對於學界的小數量蒐集、特定專業性的問題探討，業界對於地理資訊與跨領域整合的潛在商 機與經濟效應上更為積極。例如 Google 在地圖、街景上的投資，同時持續「免費」開放使用其服務，吸引了全球使用者的力量「零成本」貢獻大量的使用者標記，累積了目前任一個國家無能與之匹敵的大數據資料。無論是在地圖、地理數據、網頁文字、圖片及使用者查詢詞紀錄，都讓其他 LBS 應用服務難望其項背。而其他商業巨擘如 Bing Maps、Yahoo! Maps、Apple Maps、Facebook 的地理資料庫所擁有的數據亦不容小覷，甚至是全球性非營利組織的地理位置資訊，如：OpenStreetMap[1]、Wikimapia[2]等也都具備了數千萬的 POI 資訊。

Ahlers 與 Boll 在基於地點的網頁搜尋研究中[1]，提出了一個從網頁中擷取地點的系統架構，主要分為 crawling、斷詞與索引網頁，以及搜尋與排序等三個子系統，其中他們所採用的 crawling 策略又可分為兩種：以地點字典為主和以關鍵字為主的方法[2]，透過自適應化(adaptive)的學習與預測可能包含地點的網頁來有效提升整體召回率(recall)，該研究主要針對德國多個城市進行網頁爬取與索引。

在本篇論文中，我們從網際網路中找出包含地址網頁，並且利用命名實體辨識(Named Entity Recognition, NER)從網頁中擷取地址以及 POI 名稱並且將其配對，在得到 POI，再為其找出相關的描述，使得該筆 POI 資料能夠在資訊檢索系統中被檢索。因此本研究的主要技術可以分為如何有效地爬取目標網頁、命名實體辨識、地址和 POI 名稱的配對以及相關資訊擷取。

我們採用的中文地址擷取方法是 Chang 等人於 2012 年所提出的模型[5]，使用機器學習法中序列標記的 CRF 做為其訓練及測試模型，配合台灣地址的特性建立了 17 種地址特徵並且使用 Start/End 標記法，接著再配合極大分子序列演算法(Maximal Scoring Subsequences)，其準確率約在 94%至 99%之間。

而中文組織名稱擷取模組的建置則是使用 Huang 等人於 2015 年提出的方法[8]，同樣是使用 CRF 做為其訓練及測試的模型，利用組織名稱中常出現的詞彙(e.g., 店、公司)以及組織名稱前後常出現的詞彙等總共建立了 18 種特徵，並且使用 Self-Testing 以及 Tri-Training 等方法再更進一步地提升準確率，最終其準確率可以在非結構化的網頁中達到 86.13%。

[1] http://openstreetmap.tw/
[2] http://wikimapia.org/

在相關資訊擷取的部分，雖然 Li 等人[7]與 Chang 等人[5]的研究中都有提到此部分，但其效能並不佳。在 2012 年 Su[12]發現他們的做法過度理想化各筆紀錄(Record)的儲存標籤皆是採用同一規格標準，若是標籤的樣式稍有例外出現，就會發生連鎖錯誤，導致擷取失敗。為了解決此問題，Su[12]將 2010 年 Wei Liu 所提出基於視覺的資料紀錄擷取演算法[9]套用在地址相關資訊擷取的研究中，並重作 Li[7]的實驗，將 F-measure 由 79.12% 提昇至 95.04%。

三、 系統架構

我們的系統架構圖如圖 三所示。本系統的第一部分是利用關鍵字以及地址 pattern 組合而成的查詢詞透過 Google 搜尋引擎來搜集包含地址網頁(ABP)，並使用代理伺服器提升搜尋效率。本系統的第二部分則利用地址辨識模組以及中文組織名稱辨識模組找出網頁中的地址以及 POI 名稱，接著再用這些地址及組織名稱組成 POI 名稱與位置的基本配對。第三部份則為每一個 POI 配對擷取其相關資訊。最後將每一組配對和配對的相關資訊整理成一筆 POI，並放入 POI 資料庫中供使用者查詢。

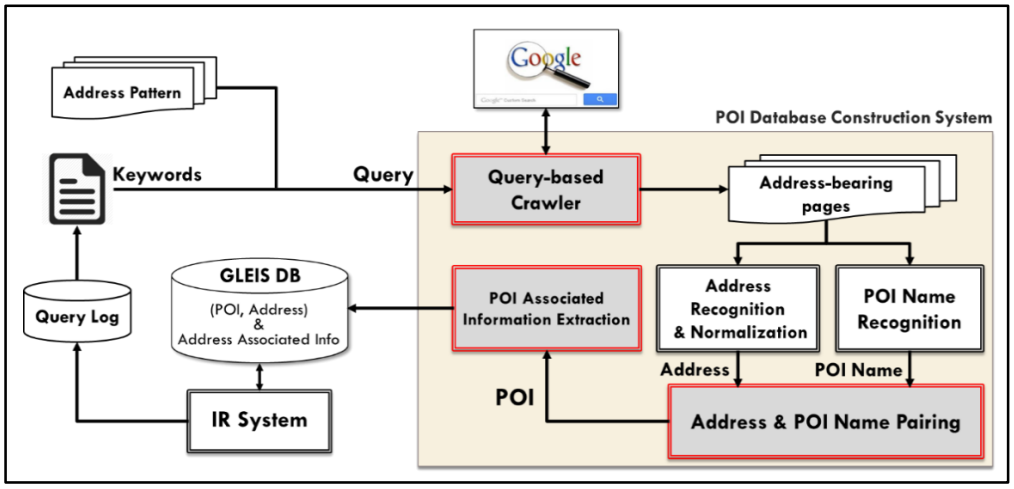


圖 三、POI 資料庫建置系統架構圖

3.1 Query-based 爬蟲

在本論文中，我們設計一個 Query-based 爬蟲取得 ABP。我們之所以收集 ABP 的原因是因為地址相較於其他相關資訊特徵較為明顯，因此使得地址相對容易辨識，此外每一筆 POI 都需要有經緯度的資訊才能定位在電子地圖上，而地址能夠透過許多工具轉換為經緯度。因此我們選擇以地址為基礎，為每一筆地址擷取其名稱和相關資訊並將其整理成 POI。

- 查詢關鍵字

為了使搜尋到的網頁盡可能包含地址，我們使用”關鍵字+地址 pattern”做為我們的查詢詞模型，其中我們使用(路 OR 街 OR 段 OR 巷 OR 弄)* 號做為地址 pattern。再根據關鍵字的類型，分以下兩種作法：

1. 類別：以 26 個縣市加上地址 pattern 以及類別關鍵字(e.g., 餐廳、服飾、交通)做為

查詢詞，接著取回搜尋結果的前 500 個網頁。

2. POI 名稱：以地址 pattern 加上 POI 名稱關鍵字(e.g., 怡客咖啡、星巴克)做為查詢詞，接著取回搜尋結果的前 20 個網頁。

另外，由於多數的連鎖商店都會在自己的官方網站上介紹分店資訊，但是如果我們直接使用連鎖商店名稱搜尋的話，Google 搜尋引擎大多只會回傳官方網站的首頁，因此若我們僅使用上面兩種查詢關鍵字的話，將會漏掉這一類型的網頁。為此我們設計了第三種類型的查詢關鍵字來解決這一問題：

3. 連鎖商家名稱：以連鎖商家名稱及"門市 or 分店"做為查詢詞，接著取回搜尋結果的前 10 個網頁。

圖 四、Query-based 爬蟲所使用的關鍵字

- 搜尋效率的改善

由於 Google 搜尋引擎對於一般使用者的查詢使用量限制，在免費的情況下我們沒有辦法連續使用相同的 IP 對 Google 搜尋引擎做查詢，根據我們的觀察，若要長時間穩定的查詢，兩次查詢間大約需要間隔 120 秒鐘，否則該 IP 就會被封鎖。如此一來會大幅增加搜尋的成本，因此我們使用 Heroku 代理伺服器來解決這一問題。

我們的作法如圖 五所示，首先我們透過 Heroku 代理伺服器取得 Google 搜尋引擎的搜尋結果，接著發出指令讓 Heroku 代理伺服器重新啟動並且進入等待，重新啟動後的 Heroku 代理伺服器會得到一個新的 IP 同時也會喚醒爬蟲程式，重複以上的步驟。

根據我們的觀察，若使用一般的方法查詢，一個小時約獲得 20 次搜尋結果。若使用 Heroku 代理伺服器的作法，一個小時能夠查詢 70 次，相較之下效率提高了 3.5 倍。

圖 五、使用代理伺服器的爬蟲運作流程圖

3.2 POI 擷取模組

透過 Query-based 爬蟲蒐集了大量 ABP 後，我們分析這些 ABP 並從中擷取出 POI 名稱、

地址和 POI 相關資訊等資料。由於地址是一筆 POI 資料中最基礎的資料，因此我們使用 Chang 等人提出的中文地址辨識模型[5]，從 Query-based 爬蟲蒐集回來的 ABP 中擷取出地址。

POI 名稱是人們指稱 POI 必要的資訊，因此在擷取 ABP 中的地址後，我們進行 POI 名稱辨識。POI 名稱辨識可以視為實體名稱辨識(Named Entity Recognition)中的公司組織名稱辨識，其做法主要採用 CRF 做為學習演算法來建立擷取模型。在本篇論文中，我們則使用 Huang 等人提出的中文組織名稱辨識模組[8]進行擷取。

從 ABP 中擷取 POI 的範例如圖 六所示，首先找出網頁中的地址，接著從地址前後固定字數的範圍內(Window Size)辨識出 POI 名稱後和該地址組合產生配對。

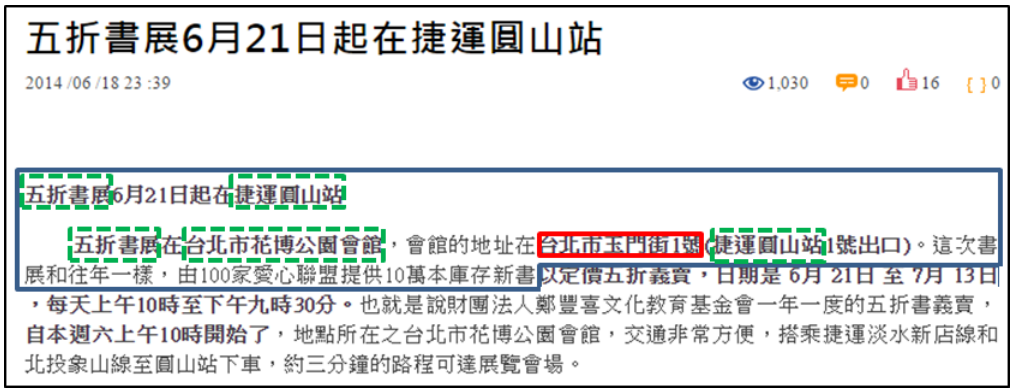


圖 六、地址以及 POI 名稱擷取示意圖，藍色實線為窗框範圍、紅色實線為擷取出的地址、虛線為擷取出的 POI 名稱

- POI 名稱和中文地址之辨識與配對

由於每筆地址可能對應到多個 POI 名稱，因此必須藉由計算所有 ABP 中各筆地址與 POI 名稱配對的相關度，找出最可能的地址與 POI 名稱配對。配對的方法如下，首先我們利用 POI 名稱和地址間的文字距離倒數做為權重，計算該配對的相關分數，分數越高表示該筆配對越有可能存在。接著對於每一個地址我們都選相關分數前三高的 POI 配對作為候選 POI 配對。接著我們將 POI 配對問題視為候選 POI 配對的分類問題，因此計算完相關分數後，我們用 POI 配對驗證模組來判斷候選 POI 是否正確。

在 POI 配對模組的訓練中，我們使用 21,899 個 POI 配對做為訓練資料，對於每一個 POI 配對我們分別使用地址、POI 名稱，及其組合做為查詢詞，透過 Google 回傳頁面中一些較強的指標做為 POI 配對驗證模組的特徵(Features)，最後我們使用 LibSVM[4]並且搭配 Chuang 等人[6]的方法使用 27 個特徵訓練出一個 POI 配對驗證模組，詳細的特徵及說明如表 一所示。我們所用到的特徵大致可以分為以下八類：搜尋結果的數量(F1~F5)、地址和 POI 同時出現在單筆 Google 摘要中的比例(F6~F8)、用皮爾森相關係數計算地址及 POI 分別與 Google 摘要的相關度(F9~F11)、Google 摘要間的餘弦相似性(F12~F14)、各別 Google 摘要的排序分數(DCG)(F15~F17)、網頁的最新修改日期(F18~F20)、Google 摘要中的語意詞(Semantic word)(F21~F22, F26)、地址和 POI 間的文字距離(F23~F25)、該筆 POI 配對是否存在於 Google Map 中(F27)。

表 一、POI 配對驗證模組的特徵

F	Name	Descriptions
1	logC(*a*)	# of search results for query a in log scale
2	logC(*s*)	# of search results for query s in log scale
3	logC(*a*, *s*)	# of search results for query *a*+*s* in log scale
4	R(*a* + *s*|*a*)	the ratio of C(*a*+*s*) to C(*a*)
5	R(*a* + *s*|*s*)	the ratio of C(*a*+*s*) to C(*s*)
6	P(*a* + *s*|T_a)	the percentage of top 10 snippets from Q_a that support POI pair (*a*,*s*)
7	P(*a* + *s*|T_s)	the percentage of top 10 snippets from Q_s that support POI pair (*a*,*s*)
8	P(*a* + *s*|T_{a+s})	the percentage of top 10 snippets from Q_{a+s} that support POI pair (*a*,*s*)
9	Corr(*a*, *s*|T_a)	Correlation of a and s in T_a
10	Corr(*a*, *s*|T_s)	Correlation of a and s in T_s
11	Corr(*a*, *s*|T_{a+s})	Correlation of a and s in T_{a+s}
12	cos(T_a, T_s)	the cosine similarity for snippet T_a and T_s
13	cos(T_a, T_{a+s})	the cosine similarity for snippet T_a and T_{a+s}
14	cos(T_s, T_{a+s})	the cosine similarity for snippet T_s and T_{a+s}
15	NDCG(*s*|T_a)	the rank of *s* in top 10 snippets from T_a
16	NDCG(*s*|T_s)	the rank of *s* in top 10 snippets from T_s
17	NDCG(*s*|T_{a+s})	the rank of *s* in top 10 snippets from T_{a+s}
18	Date(*a*, *a* + *s*)	D_a - D_{a+s} in log scale
19	Date(*s*, *a* + *s*)	D_s - D_{a+s} in log scale
20	Date(*a* + *s*)	Today - D_{a+s} in log scale
21	W(*p*, T_{a+s})	# of true words in snippet T_{a+s}
22	W(*n*, T_{a+s})	# of false words in snippets T_{a+s}
23	Lenmin(T_{a+s})	the minimum word count of string between a and s in snippets T_{a+s}
24	Lenmax(T_{a+s})	the maximum word count of string between a and s in snippets T_{a+s}
25	AvgLen(T_{a+s})	the average word count of string between a and s in snippets T_{a+s}
26	W(Dict, T_{a+s})	the average count of connection words (e.g., address is, TEL is, located on) for each middle string in snippets T_{a+s}
27	MarkMap	whether the pair is marked on Google Maps or not

***** a 為地址、s 為 POI 名稱、T 則表示 Google 摘要

3.3 POI 相關資訊擷取

多數的時候使用者會用關鍵字或是類別來查詢商家，因此我們必須有描述 POI 的相關資訊來使得這類型的查詢能夠成立。本論文中的 POI 相關資訊來源可以分為以下兩類：

1. 包含地址網頁

對於含有多筆地址的網頁，Su 提出的相關資訊擷取方法[12]可以找出每一筆地址的相關資訊範圍，但是這一演算法並不適用於僅包含單一地址的網頁，同時因為 HTML 的正規化的失敗，造成不少多筆地址的網頁無法處理的問題。因此我們在本篇論文中提出新

的做法，透過加入中文組織名稱來幫助我們從地址網頁中找出地址的相關資訊範圍。

本論文中的 POI 相關資訊擷取模組的輸入是一對組織名稱和地址的配對。首先我們會先將網頁轉換成文件物件模型(Document Object Model)架構，並將網頁視為樹狀結構，接著找出地址所在樹葉節點的位置以及組織名稱所在樹葉節點的位置(如圖 七所示)，最後以地址節點和離地址節點最近的組織名稱節點之最小共同祖先節點做為新的根節點，並將該子樹(圖 七中虛線部分)視為此配對的相關資訊。

2. 搜尋結果片段

POI 的相關資訊來源除了網頁本身之外，Google Snippets 也是我們考慮的項目，因為網頁中出現的地址未必與該網頁的主題性相同，因此若資料來源僅使用網頁內容，可能會造成部分配對的相關資訊完全錯誤的問題。為此我們使用配對中的“名稱”＋“地址”做為查詢詞(地址與名稱皆加上雙引號)，取回 Google 搜尋引擎回傳前十筆網頁的 Snippets 做為該配對的相關資訊。

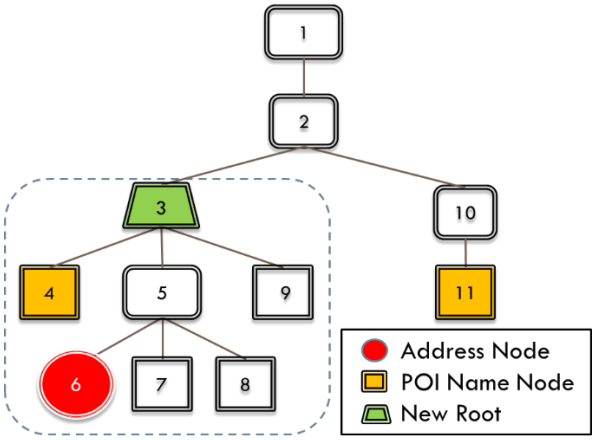


圖 七、ABP 中的相關資訊子樹

- 相關資訊摘要

經過 POI 相關資訊擷取後，對於每一個配對我們都有許多的描述訊息，但有些描述訊息可能是與該配對無關或是描述不夠精確的句子，因此我們使用資訊檢索模型來計算給定一個 POI 名稱 *poi* 做為查詢關鍵字這些候選相關資訊所產生的句子與查詢的相關分數 (|*poi*)，以便選出最相關的句子 *s* 做為地址相關資訊。也就是說，將相關資訊句子視為文件，而 POI 名稱視為查詢詞：

$$(\ |poi) = \frac{(poi| \ ) \ (\ )}{(poi)} \propto \ (poi| \ )$$

我們使用貝式定理來估計 (|*poi*)，對同一個查詢詞 *poi* 而言， (*poi*)是固定的，所以可以省略不計。假設每一個文件的機率 ( )都是一致的，因此 ( )亦可以省略。

為了整合句子的相關分數與主題模型以有效萃取出代表性的摘要，我們使用詞頻與倒詞頻來估計 *P*(*poi*|*s*)，並藉由 Latent Dirichlet allocation (LDA)[3] 產生的語言模型來做為 *P*(*poi*|*s*) smoothing 的方法，再加入λ係數調整權重，公式如下：

$$(poi|\ ) = \lambda(tf * idf) + (1\text{-}\lambda)P_{da}(poi|\ )$$

對於每一個文件 s，我們使用 LDA 取得其多項式分布θ ，並利用潛藏主題 z 計算 $(poi|\ ,\Phi_k)$做為 LDA 產生的語言模型 $P_{lda}(poi|s)$的估計，公式如下：

$$P_{da}(poi|\ ) = \ \ (poi|\theta\ ,\Phi_k) = \sum\ \ \ \ (poi|\ ,\Phi_k)\ \ (\ |\theta\ )$$

(這裡的Φ_k是主題 k 中的詞分布，而Θ 是文件 s 的主題分布)

我們利用此公式算出的 $(poi|\ )$，為每一筆 POI 相關資訊中的所有句子給予一個分數並排序，最後選擇分數較高的句子做為該 POI 的相關資訊。

四、 結果實驗

本論文中我們進行了三個實驗，分別針對系統的多個模組進行效能與效率的評估。第一個實驗是爬蟲的搜尋效率，第二個實驗是 POI 配對的準確率評估，最後第三個實驗是相關資訊擷取的評估。本研究的實驗中，我們定義了以下兩個測量值：

- 地址包含率(ABR)
  ABR = 包含地址的網頁 / 拜訪的網頁數量
- 投資報酬率(ROI)
  ROI = 不重複的地址數量 / 拜訪的網頁數量

4.1 Query-based 爬蟲搜尋效率

我們分別以地址 pattern 的效能、Heroku 代理伺服器的搜尋效率和 ABP 的搜尋效率這三個實驗來評估 Query-based 爬蟲的效能。

- 地址 pattern 的效能評估

在這個實驗中我們比較了兩種不同的地址pattern的效能：

1. <城市名稱> * 號
2. <城市名稱> * (路 OR 街 OR 段 OR 巷 OR 弄) * 號

為了分析不同的地址pattern在不同深度下所抓取回來的網頁的地址包含率(ABR)，我們分別比較前100、200、300、500、750及1000筆搜尋結果的ABR。從圖 八中我們可以觀察到不論對於哪一種深度，第二種地址pattern的ABR都高於第一種地址pattern，因此我們最後選擇使用第二種地址pattern做為查詢字串。另外值得注意的是，這兩種地址pattern在過了前500筆搜尋結果後其ABR都大幅降低，因此為了有效率地蒐集ABP，我們的搜尋深度設定為前500筆網頁。

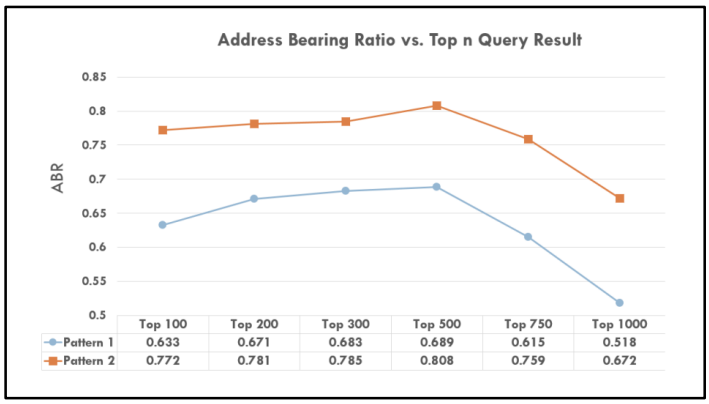
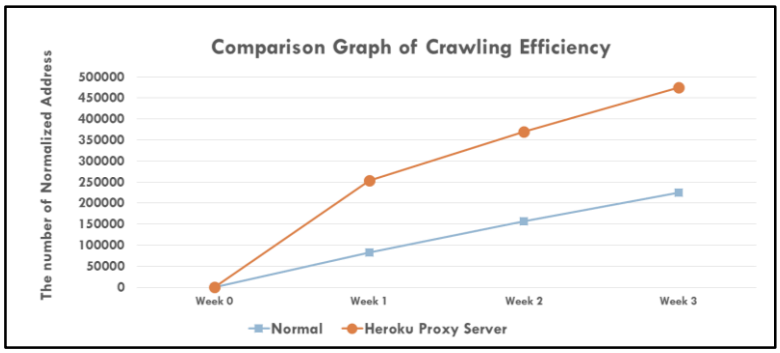


圖 八、不同的地址 pattern 的 ABR 比較圖

- Heroku 代理伺服器搜尋效率

在這個實驗中，我們使用傳統搜尋爬蟲做為基本方法，與Heroku代理伺服器的改進方法比較ABP的爬取效率，並且觀察搜集到的地址數量持續3周，圖九為效能比較結果。從圖九中可以看出使用Heroku代理伺服器的方法的搜尋效率比基本方法還要好上許多，因此利用代理伺服器的方法可以明顯的增進抓取的效率。



圖九、代理伺服器方法和基本方法的爬取效率比較

- ABP 搜尋效率

在這個實驗中，我們比較了三種不同爬蟲的ABP搜尋效率，Baseline是一般的廣度優先爬蟲，黃頁爬蟲是專門爬取各種黃頁資料的爬蟲，例如：中華黃頁[3]、愛評網[4]，Query-based爬蟲則是本論文設計的爬蟲，比較結果如表 二所示。

[3] https://www.iyp.com.tw/
[4] http://www.ipeen.com.tw/

表 二、三個爬蟲搜尋 ABP 的效能比較結果

Items		Baseline	Yellow Page Crawler	Query-based Crawler
# Visited Webpages	(a)	132,628,290	105,727	652,693
# Address-Bearing Pages	(b)	508,038	105,693	488,036
# Extracted Address	(c)	1,034,402	944,864	6,359,283
# Distinct Address	(d)	190,180	693,868	888,838
ABR	(b)/(a)	0.004	0.999	0.748
ROI	(d)/(a)	0.001	6.563	1.362

從表 二中可以看出雖然Baseline的方法因為沒有Google搜尋引擎的限制，因此能夠爬取相當大量的網頁，但Baseline所找到的地址數量仍然非常的少而且投資報酬率和地址包含率都相當低。Query-based爬蟲的投資報酬率和地址包含率都比Baseline還要高上許多，但是比黃頁爬蟲還要來的相對低一些，其原因是因為Query-based爬蟲的資料來源並不像黃頁一樣侷限在某些特定的網頁和領域，也因此Query-based爬蟲能夠更找到更為普及的網頁中所包含的地址，同時從表 二中也可以觀察到Query-based爬蟲確實能找到比黃頁爬蟲更多的地址。

此外為了瞭解Query-based爬蟲能夠找到多少存在於黃頁以外的地址，我們先將兩種方法所擷取的地址進行正規化後，比較Query-based爬蟲和黃頁爬蟲所擷取到的地址重疊數量。然而根據我們的統計結果，Query-based爬蟲所擷取到的88萬筆地址當中有超過50萬筆是黃頁爬蟲中所沒有的，因此若僅使用黃頁中的商家資料做為電子地圖的POI，會有相當多的POI沒有辦法被電子地圖所查詢到。這個實驗說明了Query-based爬蟲可有效地補充黃頁爬蟲所不足的POI。

4.2 POI 配對準確率

在擷取88萬筆地址後，我們使用黃頁中一對一的POI做為正確答案，計算Query-based爬蟲所搜尋到的50萬個網頁分別以不同的窗框大小(Window Size)所產生的POI配對的準確率。對於每一個地址，我們將直接選用相關分數最高的配對為正確的POI的方法做為Baseline方法，在此實驗當中我們比較了使用Baseline做法在不同的窗框大小下所產生的POI的涵蓋率與準確率，詳細結果如表 三及圖 十所示。

1. $$\text{Coverage Ratio} = \frac{\#\text{ Overlapping addresses containing correct POI pairs}}{\#\text{ Overlapping addresses with yellow pages addresses}}$$

2. $$\text{Accuracy} = \frac{\#\text{ of addresses that are predicted correct}}{\#\text{ Overlapping addresses containing correct POI pairs}}$$

表 三、Baseline 方法在不同窗框大小下的效能比較

Items	Window Size 50	Window Size 100	Window Size 150
# Recognized POI names	10,773,585	20,539,371	30,144,909
# Distinct POI names	702,793	844,165	934,896
# Pairs	4,406,985	7,630,332	11,062,343
# POI	694,730	743,555	764,840
# Overlapping addresses with yellow pages	264,342	264,342	264,342
# Overlapping address containing correct POI pairs	107,257	121,932	129,913
# of addresses that are predicted correct	52,222	53,536	54,031

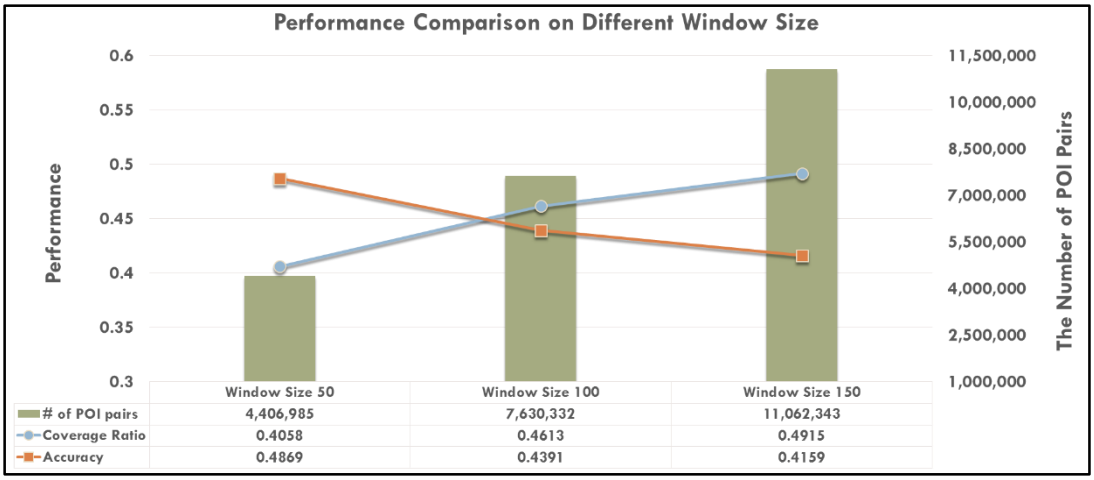


圖 十、Baseline 在不同窗框大小下的 POI 配對效能圖

從圖 十中的結果可以看出在窗框大小為100時有平衡的涵蓋率與準確率，因此後續的實驗我們都選用100個字的範圍做為我們預設的窗框大小。為了評估POI配對驗證模組的效能，我們隨機選取7500個地址並產生了21,899個POI配對(對於每一個地址我們都取3個候選POI配對)做為我們的訓練資料，接著我們做3-folds cross-validation來評估我們的POI配對驗證模組的效能，實驗結果如圖 十一所示。

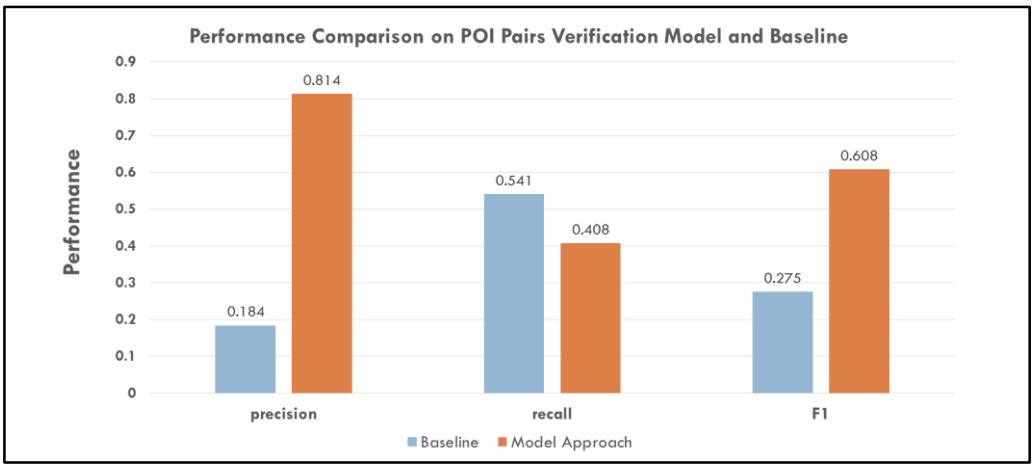


圖 十一、使用不同大小自動關鍵詞庫比較效能

從圖 十一的結果中我們可以看出，雖然在 recall 方面 POI 配對驗證模組的效能略低於 Baseline，但是在 precision 方面卻遠高於 Baseline。我們認為原因是 Baseline 的做法對於每個地址都一定能找到一個 POI 名稱做配對，因此導致雖然有較高的 recall 但 precision 卻相對的非常低，然而透過 POI 配對驗證模組的做法因為判斷較為嚴謹，因此造成 recall 的部分稍低但在 precision 的部分可以有非常好的效果。我們認為在 POI 配對這項任務上 precision 的重要性遠高於 recall，因為我們不能夠提供錯誤的 POI 資料給使用者，因此要盡可能的確保 POI 資料庫中的資料的正確性。

4.3 相關資訊擷取效能評估

因為相關資訊的正確與否較難以判定，因此為了有效評估相關資訊的效能及實用性，我們設計了一個 IR 實驗，透過 POI 檢索系統來測量 POI 相關資訊的品質。在本實驗中，我們透過上個實驗中敘述的 Baseline 作法設計了兩個 POI 檢索系統 POIDB_AI 和 POIDB，其中 POIDB_AI 是由包含相關資訊在內的 POI 資料庫建置的 POI 檢索系統，POIDB 則是由除了相關資訊以外的 POI 資料所建置的 POI 檢索系統。我們透過兩個 POI 檢索系統回傳的 POI 的正確性(Average Precision@10)及數量來測量相關資訊正確性以及實用性。我們選擇了 9 個地點做為檢索中心點，其中包含了市中心、市中心旁的地區以及離市中心較遠的地區，然後每個中心點再各自配合 200m、500m、1000m 三種不同的檢索半徑，共形成 27 種組合。對於每一種組合，我們再分別使用 18 個與日常生活較為相關的查詢詞，如：餐廳、服飾、電影等來進行查詢，實驗結果如圖 十二所示。從實驗結果中可以看出 POIDB_AI 所能查找到的 POI 數量是 POIDB 的兩倍以上，而且 POIDB_AI 的 AP@10 在非常小的檢索半徑中依然可以維持在 80%以上。從本實驗的結果讓我們可以確信 POI 相關資訊擷取模組所擷取的相關資訊可以真正的幫助 POI 的檢索。

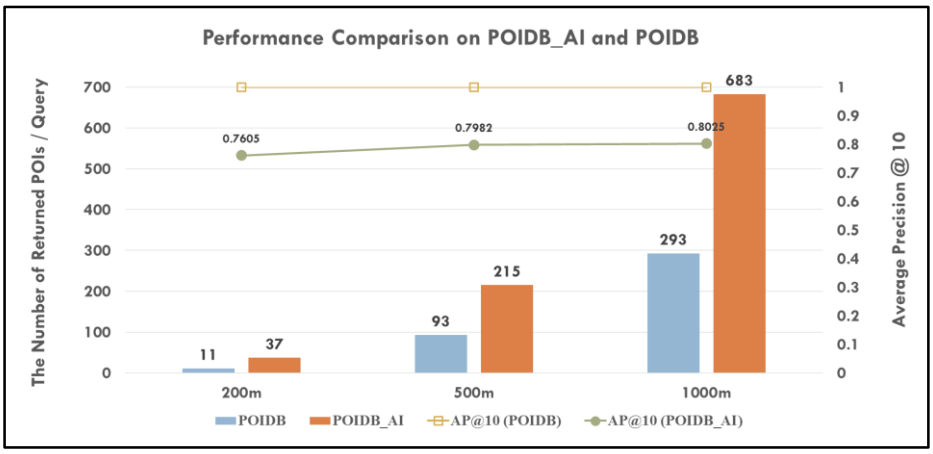


圖 十二、POIDB_AI 和 POIDB 的效能比較圖

五、 結論與未來研究

電子地圖的發展大大改變了現代人的生活習慣，且已經在我們日常中扮演了不可或缺的重要角色。在智慧型手機普及的現代，只要在能連接網路的地方隨時都能夠透過電子地圖獲取任何地點的資料，甚至還有路線規劃和即時導航等附加功能，讓人們不會在前往陌生的地點時感到不便。雖然這樣莫大的改變確實影響了我們的日常生活，但是要倚靠人工的方式來建立出含有極大量POI的電子地圖是一件相當艱難的任務。不過歷史較電子地圖悠久的網際網路，其實早已累積了大量的地理資訊可供我們使用。

本論文透過Query-based爬蟲在網際網路中找出含有地址的網頁(ABP)，並藉由命名實體(NER)辨識找出其中的地址以及組織名稱，接著透過地址與組織名稱配對系統找出正確的配對，然後從網頁中或是Google Snippets中摘要每一組配對的相關資訊，最後將這些資料做為電子地圖中的POI來使用，如此就能快速增加電子地圖中POI的資料量。

此外，為了瞭解藉由我們的系統自動化所產生的地址相關資訊是否能真實應用到電子地圖的檢索上，我們藉由資訊檢索的實驗結果顯示，Query-based爬蟲搜集而來的網頁中擷取出的POI資料，確實具有相當的實用性。

在未來我們會專注在如何整合我們擷取的POI以及現有POI資料庫中的POI，如此一來不僅僅是利用網路上的資料創建一個全新的POI資料庫，同時也能夠利用現有的POI資料庫再進一步的豐富我們POI資料庫中的資料。此外我們也希望系統除了定期爬取新的POI之外，同時能夠定期檢查我們的POI資料庫中現有的資料，並將過期的POI過濾掉，以確保POI資料庫能夠隨時提供給使用者正確的POI。

References

[1] D. Ahlers and S. Boll, Location-based Web Search. The Geospatial Web, pp. 55-66, Springer, 2007.

[2] D. Ahlers and S. Boll, Adaptive Geospatially Focused Crawling. CIKM, China, Nov. 2–6, 2009.

[3] D. M. Blei, A. Y. Ng, and M. I. Jordan, Latent Dirichlet allocation. Journal of Machine

Learning Research, 993–1022, 2003.

[4] C.-C. Chang, C.-J. Lin, LIBSVM: a library for support vector machines. ACM Transactions on Intelligent Systems and Technology, 2(27):1-27, 2011.

[5] C.-H. Chang, C.-Y. Huang and Y.-S. Su, Chinese Postal Address and Associated Information Extraction. The 26th Annual Conference of the Japanese Society for Artificial Intelligence, 2012.

[6] H.-M. Chuang, C.-H. Chang and T.-Y. Kao, Effective Web Crawling for Chinese Addresses and Associated Information. EC-Web, Munich, Germany, 2014.

[7] Chia-Hui Chang, Shu-Ying Li, MapMarker: Extraction of Postal Addresses and Associated Information for General Web Pages. Web Intelligence, 2010.

[8] Y.-Y. Huang, C.-L. Chou, C.-H. Chang, Web NER Model Generator Tool based on Google Snippets, submitted for publication, 2015.

[9] Wei Liu, Xiaofeng Meng, Weiyi Meng, ViDE: A Vision-based Approach for Deep Web Data Extraction. Transactions on Knowledge and Data Engineering, IEEE, 2007

[10] Yu-Yang Lin, Chia-Hui Chang, 網頁商家名稱擷取與地址配對之研究 (Store Name Extraction and Name-Address Matching on the Web). ROCLING, 2014.

[11] G. Stirling. “Study: 78 percent of local-mobile searches result in offline purchases”, Search Engine Land. Apr. 9, 2014.

[12] Y.-S. Su, Associated Information Extraction for Enabling Entity Search on Electronic Map, National Central University, 2012.