基於群眾地理資料之自動化文章位置標記

黃郁菱[1] 黃俊穎[2]
國立台灣海洋大學 資訊工程學系[1]
國立交通大學 資訊工程學系[2]
10357039@ntou.edu.tw[1]
chuang@cs.nctu.edu.tw[2]

摘要

從文章中擷取地理資訊一直以來都是一個熱門的研究主題。過去的研究曾嘗試從部落格文章、Twitter 等不同文字來源進行地理位置標記。在這些研究中，大多數都使用自然語言處理的方式進行關鍵詞分析，再搭配數種不同的啟發式規則進行地理位置擷取。然而這些研究大多得面對兩個挑戰性問題:語言處理的困難度及地點標示的精細度。在這個研究裡。我們嘗試發展一個高精細度且同時支援多種語言的地理位置擷取系統。我們主要針對新聞網站的內容進行擷取。和現有的研究之不同點是，我們並不使用自然語言處理的方法，取而代之的是，我們基於簡單的 N-gram 來進行斷詞，並搭配由群眾貢獻的地理資料進行地名詞庫的識別，其中還包括了額外的地理特徵，像是地名詞的大小及詞庫之間的關係進行分析，以提高系統精確度。基於這些創新的方法設計，我們所提出的方法可以精準的從不同語言的文字中擷取出足以代表來源文字的地理位置。我的實驗結果顯示我們的所提出的方法達96%的準確率。

關鍵字:群眾地理資料、OpenStreetMap、地名詞庫、地理資訊擷取

1. 前言

因應各項資訊快速成長，閱讀網路文章來獲取新知成為現代人最常使用的方法，年輕人流行在各種論壇、討論區交換意見已成為生活不可取代的一部分，甚至透過不同平台創造凝聚力，進而造就了白衫軍運動、太陽花學運等等的活動。

在這些網路文章中，部分包含著地名訊息，有些作者在發表文章時，將地點與座標實際做結合，讓閱讀者了解事件實際發生的地點，例如:FB 的「打卡機制」、蘋果日報的「新聞打卡地圖」，皆是方便讓讀者在閱讀文章的同時，快速了解文章的發生地並且做出反應，除此之外，地名訊息也富含不同的價值，例如分析軍用文章可以了解戰爭當時的動向或是其戰略方向;分析早期新聞資料可以了解當時聚落發展度等等，皆是相當有用的資訊。但要將文章與地點正確的做連結並不是容易的事，主要的困難點在於地點設定的精細度及自然語言的處理。如何從文章中準確找出關鍵詞(Geo-parsing)，再從這些關鍵詞中找出屬於地理名詞 (Geo-name)的部分，並且將他們對應到正確的位置 (Geo-tagging)，最後再決定這些名詞中哪個才是文章實際發生的地點(Disambiguation)。

從目前的研究發現，在標記地點的精細度上通常只到縣市階級，讓閱讀者無法了解更詳細的發生地點，例如:XX 區或是 OO 路。而在自然語言的處理上，大多只對單一語言做語意上分析，利用該語言政府機關所發布的行政地區當作地名詞庫 (Gazetteer)，找出文章內的地理名詞，而篩選方法也多採用計算地點次數或是距離遠近等等方式來判斷。因為對於地點的精細度要求較低，限制了研究成果的應用範圍。

為了能夠處理多國語言及提高地點的精細度，在自然語言的處理上，我們忽略各國語言的語法，改用使用最基礎的 N-gram 來做文章的斷詞處理，有鑒於網路的興盛，我們使用群眾地理資料之一的 OpenStreetMap (OSM)做為地名詞庫，此舉除了解決各地地理名詞上的稱謂差異，還可利用 OSM 中，當地人所提供的區域特徵來增加精細度，除了以往常用的判定方法，我們提出 5 種新的篩選方法來判斷正確的地理名詞，此外我們也分析各個地理名詞間的關係來剔除不合適的地點，並搭配使用圖學演算法 (CGAL)來提高整體系統的判斷準確率。基於這些新的設計，我們所提出的方法可用在不同的語言的文章中，並有 96%的地名判斷正確率。

本論文的章節安排如下。在第二節我們介紹幾個相關的研究；在第三節與第四節我們詳細說明我們所提出的方法，包括方法的設計及其實作；第五節則呈現各項方法及整體系統的測量結果。最後，結論及未來展望則在第六節討論。

2. 相關研究

自古以來，人們就相當重視文章所透露出的地點資訊。在戰爭時期，軍方利用敵方文章內有提及的地理名詞進行分析，每個地點均有其戰略價值。像是 TAY 等人[20]利用地理資料來將低森林火災的誤判率，首要工作便是正確的找到該文章的對應位置。

2.1 平面型文章

在網路盛行前，研究者只能利用歷史性的文章與報表資料等傳統平面資料，去了解古人們的生活環境。

2.1.1 消除歧異的方式

在找出文章中所含的所有地理名詞之後，我們便需判斷哪個才是該文章所指的正確地點。常用的方法包括:常用地名、地名出現的次數、各個地名間的距離或者關係，甚至從郵遞區號或是人口數多寡來判斷。

常用消除歧異的方法如 Ester 等人 [18] 始想在含有地名的文集中，新增對該地名的額外描述，如:失業率。如此再將這些地名與地圖資料做比對時便能得到更加明確的關係分佈圖。而到了 [19] 便提出了相對於純粹使用地名文集，若使用 neighborhood graphs 以及資料庫的話，會有更快的資料探勘速度。Rauch 等人 [10] 使用了 MetaCarta gazetteer 來當作地名與經緯度的對應資料庫來源，利用拉丁語系特有的方是判斷地名，像是特殊的標點符號和大小寫等等來撈出專有名詞。但是光撈出專有名詞還不足以判斷地，因為該專有名詞不見得就事地名，有可能是指向某個人名，之後再用 MetaCarta 挑出特定的地名，在進行消除歧異的動作。David A.Smith 與 Gergory Crane [8] 使用 Perseus 來當作地名資料庫，利用 Perseus 內建的函式庫來判定地名，找出文章中的地名之後，再使用加權法來判斷其地名位置。

另外 Li 等人 [12] 將精細度目標設定在城市階級，給予範圍越大的地名越重的權重，並使用了郵政編號製作了一個權重關係加權表來幫助判斷。Amitay 等人 [9] 使用 Perl 來處理存有地名的 XML 檔案，當遇到文章中出現複數地名文章時，則使用了人口數來判斷地名是否可以繼續存留。

2.2 網路文章

網路文章的好處是研究者可以簡易的從網路取得所需的研究資料，並利用網路特性來進一步找出文章中所指地點，像是整體網站的架構。

2.2.1 傳統網頁文章

Keven S.McCurley [16] 提出了利用網頁中的 URL 來對應 IP 進一步找到所在地，但成功機率不穩定，因使用者可能購買其他供應商的機器來架設網站，這樣 URL 對應 IP 並不會指到正確的地點，針對這個問題，進而改為利用網站內的電話號碼與郵政號碼，對應到一個經緯度，在把經緯度與該 URL 做配對後放進資料庫。Wang 等人 [6] 明確的提出一個網站會包含的一些基礎位置架構，分別是 Provider location、Content location 以及 Serving location 這三個地點觀念，如圖 1。Provider location 是利用該網站的單位，於架設出來的網站中可以找到該單位的資訊; Content location 便是該網站中的文章內所指的地理位置; Serving location 則是該網站的服務範圍。Morimoto 等人 [22] 從網站上抓取網頁來源後，再對這些來源做地點標記，像是郵政編號、電話號碼、IP 地址、路由資訊、地理名詞特徵、超連結等來抓出地理資訊，來判斷網站屬於哪一類型。

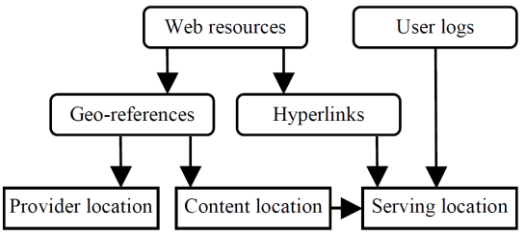

圖 1、網站基礎架構圖，來源為[6] Figure 2

2.2.2 短文辨識

近年來社交型網路文章越來越多，像是 Twitter 與 Facebook 等。由於這類網站所產生的文章通常較短，像是人們的對話一樣，北本朝展 (Asanobu Kitamoto) [17] 利用 Twitter 上的社交訊息來補足科學資料上的不足。而 Kalev Leetaru[14]則替維基辭典標定文章地點，有鑒於 Twitter 的盛行，與隔年 [15] 把系統搬到 Twitter 上做實驗。Scheffler 等人 [21] 針對德國的 tweets 做分析，缺點是其他語言支援度較差，例如日文只有 9%正確率。

目前的研究方法眾多但最主要的問題仍然是語言的支援度，大部分的研究皆是偏向某種語言或是為某個網站量身打造一個合適的判斷系統，非支援語言的正確率較低。

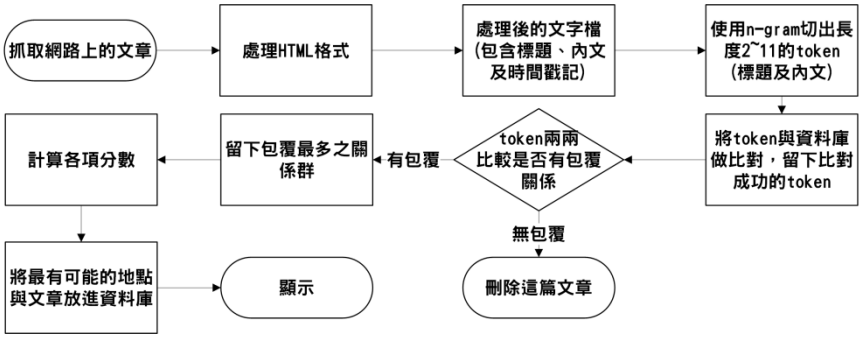

圖 2、系統整體流程圖

```
<node id="3759762857" lat="25.0894653" lon="121.645777" version="1" timestamp="2015-09-26T01:51:13Z" changeset="34255144"
 uid="3220182" user="愛兜風的貓">
        <tag k="name" v="OK超商"/>
        <tag k="shop" v="convenience"/>
</node>
<way id="138996582" version="3" timestamp="2015-07-04T10:24:56Z" changeset="32406193" uid="274857" user="Supaplex">
        <nd ref="1523687584"/><nd ref="1523687589"/><nd ref="1523687638"/>
        <nd ref="1523687537"/><nd ref="689542768"/><nd ref="1523687584"/>
        <tag k="name" v="淡大城區部"/><tag k="amenity" v="university"/>
        <tag k="addr:city" v="臺北市"/><tag k="addr:street" v="金華街199巷"/>
        <tag k="addr:district" v="大安區"/><tag k="official_name" v="淡江大學臺北校區"/>
        <tag k="addr:housenumber" v="5"/>
</way>
<relation id="33574" version="12" timestamp="2015-10-23T15:44:41Z" changeset="34823124" uid="1215520" user="bananatw">
        <member type="way" ref="60793791" role="outer"/><member type="way" ref="364410686" role="outer"/>
        <tag k="name" v="西門國小"/><tag k="type" v="multipolygon"/>
        <tag k="phone" v="+886-2-2389-2182"/><tag k="amenity" v="school"/>
        <tag k="website" v="http://core.hmes.tp.edu.tw/"/><tag k="operator" v="臺北市政府教育局"/>
        <tag k="addr:city" v="臺北市"/><tag k="addr:full" v="10845臺北市萬華區成都路98號"/>
        <tag k="addr:street" v="成都路"/><tag k="addr:country" v="TW"/>
        <tag k="addr:district" v="萬華區"/><tag k="addr:postcode" v="10845"/>
        <tag k="official_name" v="臺北市萬華區西門國民小學"/><tag k="addr:housenumber" v="98"/>
</relation>
```

圖 3、OSM 資料格式

3. 系統架構與實作

本論文將實際抓取網路上的文章來做實驗，經過處理後的新聞將帶有地點資訊，並將之存入資料庫中於地圖中顯示該文章的所在位置，我們主要有兩個目標。首先，我們必須可以支援多種語言，所以我們捨棄建立語言地名詞庫，改用最原始的 N-gram 來做斷詞。再來，我們期望可以精準的找出最符合的地點，比較地名之間的相互關係來剔除不符合的地名。整體流程圖如圖 2。

3.1 地圖資料

再做判斷系統時必定會需要的根基即是地名詞庫，而現在最直覺的取得方式便是直接使用真實的地圖資料。在本論文中需要大量的重複查詢，由於 Google Map 的使用條約中限制免費用戶每日最高查詢 2 萬次，因此我們轉而使用 OpenStreetMap (OSM)來做為判斷新聞地名時的地圖資料來源。OSM 為人們共同編輯與維護的地圖資料，並且能自行下載 OSM 上的地圖資料來做使用，使用者可自行劃選所需下載的區域，提供使用者一個 XML 格式進行壓縮後的壓縮檔進行下載，下載之後我們使用 LEX&YACC 擷取所需的地圖資料，在將處理過後的資料存入資料庫。OSM 資料格式如圖 3。

3.1.1 地圖資料的格式與關係

OSM 使用的資料格式是地形資料結構，當中由四個核心的元素，元素關係如圖 4:

- 節點 (Node)

儲存經緯度，但不儲存節點在地圖上的實際大小，比如說一個景點或者山峯。

- 道路(Way)

有序排列的節點，以折線的形式呈現，如果是封閉圈的話，那麼可能是以多邊形的方式呈現，但形成多邊形的相鄰兩個邊不一定是頭尾相連。這類原始資料大多用來呈現爲街道，河流，一個區塊，比如說森林，公園，停車場或者是一面湖。

- 關係 (relation)

有排序的節點，道路和關係，在這裡每個成員選擇性擁有一個「角色」。關係是用來表示各個原始資料的關係，比如說道路與道路的拐彎限制，一條道路分叉出來的各種小道路，或者一個區域中的一個洞，這時「角色」字串就能用來形容它們之間的關係。

- 標籤 (tag)

鍵值對 (key-value pairs)，用來儲存地圖上物件的元資料（metadata）。標籤無法獨立存在，它們必須依附在一個已存在的原始資料，比如說一個節點，道路和關係。在 OSM 的維基辭

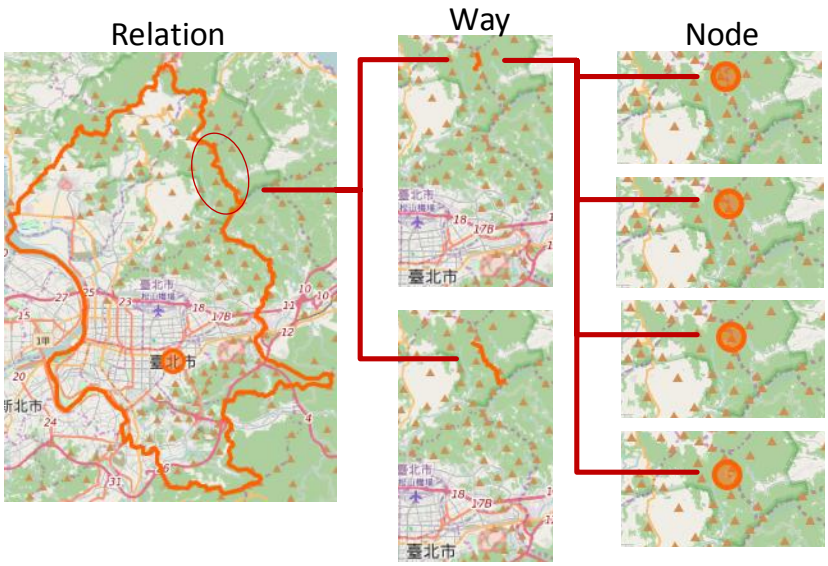

圖 4、OSM 元素關係

典上有比地圖中物件對映的關係 （ontology）和對標籤有比較詳細的介紹。

圖 4 顯示出了一個 Relation 可能為數個不同的 Way 所組成，而 Way 又可以拆解成許多的 Node。

3.2 資料庫

本論文的實驗環境中，對於資料庫的需求有兩大特點。其一是需可承擔用 N-gram 所切出的字串做大量且相異性極高的查詢;其二為可放入大量的地圖資料。我們比較傳統 SQL 資料庫與 No-SQL 資料庫來挑選符合需求的資料庫。

- 傳統 SQL 資料庫

在傳統 SQL 資料庫中我們挑選 MySQL 做比較，MySQL 資料庫在查詢時會對查詢過的字串做快取，做過快取的資料將會於下次查詢時加快查詢的速度，但如果放在異質性極高的查詢底下，快取的資料大小會急劇增加，且因每次查詢目標皆不同，也無法有良好的加速效果。

- No-SQL 資料庫

有別於傳統資料庫的 No-SQL 資料庫，這裡選用 MongoDB 來做討論。對於高密度的異質性查詢，MongoDB 在設計上不會對其做快取的動作，所以不會產生龐大的快取檔案;而再放入大量地圖資料這點來說，No-SQL 資料庫的特性之一便是良好的拓展性 (Scalability)，當有需求要擴建機器時會比較容易，而且還能減緩整體資料庫的負擔，這方面在 MongoDB 中大致分為兩方面，一個是複製集 (Replication)，存放相同資料，另一個便是分片集 (Shard)，存放不同資料。容易新增新的分片集，並於查詢時作分擔。

3.3 斷詞處理

任何語言處理的系統都必須先能分辨文本中的詞才能進行進一步的處理，現今較常用的方式是根據處理語言建立一個詞彙庫，運用此詞彙庫來判斷找出可能包含的詞，如以下兩種系統。

1) 中研院斷詞系統 (CKIP Client)

中研院提供的線上斷詞系統，使用包含一個約拾萬詞的詞彙庫及附加詞類、詞頻、詞類頻率、雙連詞類頻率等資料。在斷詞的同時也會輸出精簡詞類之標記，限制每天查詢次數，對於過長的語句支援性差。

2) 結巴中文分詞程式 (JIEBA)

Python Based 的開源中文斷詞程式，JIEBA 中文斷詞所使用的演算法是基於 Trie Tree 結構去生成句子中，中文字所有可能成詞的情況，然後使用動態規劃 （Dynamic programming）算法來找出最大機率的路徑，這個路徑就是基於詞頻的最大斷詞結果。對於辨識字典詞庫中不存在的詞則使用了 HMM 模型 （Hidden Markov Model）及 Viterbi 算法來辨識出來。

以上兩者在斷詞功能上主要是依靠完整的詞彙庫，且使用限制較多，對於詞彙庫不存在的詞支援性較差。因此我們捨棄了現在常用的斷詞系統，採用較直觀的 N-gram，優點是不需要詞彙庫，不受語言的限制，而且可以把文章中語詞的所有可能性都找出來，缺點是需要經過篩選才能找出我們需要的詞。

我們將擷取下來的文章切成長度為 2 至 11 的詞 (token)，再將切好的 token 與地圖資料庫做比對，留下與地圖資料庫比對成功的 token，這些就是這篇文章可能發生的地點候選人，斷詞範例如圖 5。

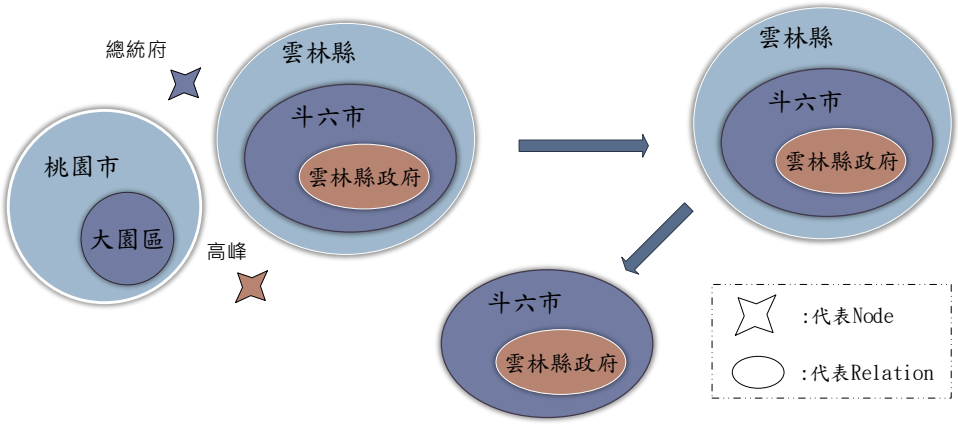

圖 6、包覆關係範例

輸入:任何語言處理

2N-gram:任何 何語 語言 言處 處理

3N-gram:任何語 何語言 語言處 言處理

4N-gram:任何語言 何語言處 語言處理

圖 5、N-gram 斷詞範例

3.4 選擇正確的地點

於其他論文中所長見的地點篩選法不外乎有「該地名在文章內出現的次數」、「計算各地點間的距離」和「地理名詞資料庫」，而在本論文中我們加上了「字串的長度」及利用圖形演算法的「地點面積」、「判斷區域間的包覆關係」來篩選地點，為了計算地點面積及包覆關係我們使用了 CGAL 來當作圖形演算法的基礎。

3.4.1 判斷區域間的包覆關係

一篇文章中可能會有很多的地名，而地名間也存在各種關係，像是大小關係、距離遠近，接下來要討論的是地名之間存在的包覆關係，像是總統府在臺北市之中，臺北市又在臺灣之中，但是總統府與行政院兩處處於沒有包覆關係的狀態，我們將判定兩兩 token 之間是否有包覆關係。

要判斷 token 之間是否有包覆關係，我們必須先找出所有能代表 token 地點的座標，和 token 圈出封閉圖的座標群。首先我們把所有 token 與地圖資料庫做比對，留下比對成功的 token，並找出這個 token 的所在地經緯度，如果 token 是一個 Way 或者 Relation，則從所有的經緯度座標裡隨機挑選一組來代表這個 token。接下來要找出哪些 token 是有圈出封閉圈，將 token 與地圖資料庫做比對，且只留下在「Relation」類別中比對成功的 token，並找出此 token 的座標群，因為在 Relation 中必定為封閉圈，所以我們捨棄從其他兩個元素中比對。最後我們將 token 的座標群用 Convex hull 化簡減少我們判斷的複雜度，Convex hull 將在本章第三小節介紹。

接著將 token 與有封閉圈的 token 兩兩比較，判斷彼此的包覆關係。例如有兩 token，A、B，且 B 可圈出一封閉圈，如果 A 的代表座標被包覆在 B 的座標群中，我們稱 A 與 B 有包覆關係，且 B 的包覆數量為 1，B 的上一層則為 A。比對成功之範例表如表 1。

表 1、包覆關係表
(R 代表為 Relation、N 代表為 Node)

地名	種類	包覆數量	上一層
大園區	R	0	桃園市
高峰	N	0	N/A
桃園市	R	1	N/A
大後	N	0	N/A

表 1 顯示桃園市與大園區有關係，而且大園區在桃園市中。token 之間的包覆關係可以協助判斷 token 的重要性，包覆數量越多的 token，代表這個 token 的關係群在文章中出現許多次，有較高可能是合適的地理名詞。我們留下包覆數量最多的群組繼續篩選，如果群組中有 token 的分類為 Relation 則捨棄最外圈的 token，要注意的是因為使用 Convex hull 減少複雜度，token 與 token 之間的關係有可能重疊，但包覆數量的判定必須為完全包覆才計算，包覆範例如圖 6。

3.4.2 篩選方法

為了評斷哪個 token 是能夠代表文章的地理名詞我們提出了五種不同的評分方法:

- 出現次數 (Count)

計算每個 token 在該篇文章中出現的次數，出現次數越多代表在這篇文章中相對重要，因此次數越高則分數越高。

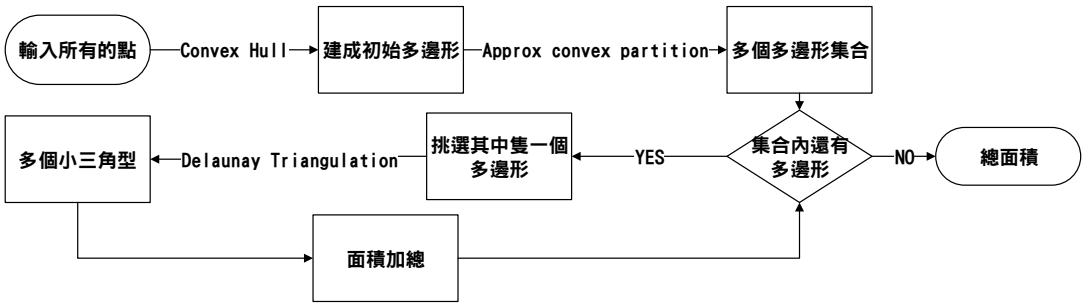

圖 8、面積計算流程圖

- 平均距離 (Average distance)

計算 token 和該篇文章中的其他所有 token 之間的距離平均，因將經緯度以座標的方法計算，計算出來距離不是實際上的距離，而是相對距離，距離越短代表此 token 在所有 token 分佈中屬於中心的位置，因此平均距離越短則分數越高。

- 字串長度 (Word length)

Word length 為計算 token 本身的字串長度，長度越長的 token 代表地名完整度越高，簡稱的機率較低，因此長度越長則分數越高。

- 面積 (Area)

計算 token 的面積，有面積的 token 代表能在地圖上圈出一個範圍，越有可能是正確的地點。為了簡化計算過程，我們使用 Convex hull 來減少複雜度，計算出來的面積不是實際上的面積，而是相對面積，面積越小分數越高，無法計算者不採計。

- 常用字 (Common word)

隨機選擇大量文章，將文章用 N-gram 切成長度為 2~4 的 token，來建成常用字資料庫，統計 token 在所有文章出現的次數，出現次數越多代表這個詞越常被使用，越有可能是我們所需要的地理名詞。

3.4.3 使用 CGAL 來計算面積

CGAL 是一個 C++的開源函式庫，提供了大多數已發表的圖形演算法，甚至更包含了 3D 的圖形演算法，這也讓 CGAL 大多數運用在圖學研究上。

要計算該區域面積不外乎要對多邊形做三角化，不過因為直接做三角化的計算效率不佳，所以進一步的做法是先把多邊形切塊，之後對每塊多邊形做三角化，再計算各個三角形的面積。為了減少整體運算的複雜度，在計算面積前先使用 Convex hull 來減少運算的點數，在所有點的外圍繞一圈可得一個凸多邊形，即是 Convex hull，Convex hull 圖形內任兩點的連線不會經過圖形外部，其時間複雜度為 O(nlogn)，範例如圖 7。接著使用 Approx Convex Partitioning 來切割多邊形，其時間複雜度為 O(n)，三角化則使用 Delaunay Triangulation，時間複雜度為 O(n)。

首先從資料庫取得組成該地點的所有點，接著把這些點做 Convex hull 化簡得到一個凸多邊形，而這個多邊形再利用 Approx Convex Partitioning 切成更小的多邊形以加速 Delaunay Triangulation 三角化的計算，最後把這些三角型做面積的加總便能夠得到該地區的面積大小，流程如圖 8 所示。

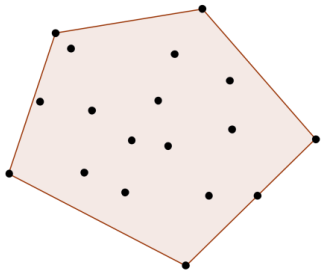

圖 7、Convex hull 範例

4. 實作與測量結果

4.1 資料庫效率測量

為了了解 MySQL 資料庫與 MongoDB 之間的效率差異，我們比較了兩者在查詢時所花的時間。測試用的查詢資料皆是從新聞上擷取出來後經由 N-gram 處理而成的字串。我們分別測試當資料庫擁有不同數量的資料筆數時，兩者的查詢時間，實驗結果如圖 9、圖 10。

- 實驗 A
  - A 資料集

    資料數量:230334 筆

    查詢數量:57956 筆
  - B 資料集

    資料數量:436597 筆

    查詢數量:57956 筆
- 實驗 B
  - A 資料集

    資料數量:436597 筆

    查詢數量:57779 筆
  - B 資料集

    資料數量:436597 筆

    查詢數量:346694 筆
  - C 資料集

    資料數量:436597 筆

    查詢數量:693348 筆

表 4、實驗結果範例

Name	Total	Avg_dis	Avg_dis score	Count	Count score	Common	Common score	Size	Size score	Long	Long score
竹南鎮	60	0.05055	16	3	16	5	6	234.5	6	3	16
苗栗縣	56	0.09383	8	3	16	97	8	1863.87	8	3	16
公義路	40	0.057647	12	1	8	2	4	N/A	0	3	16

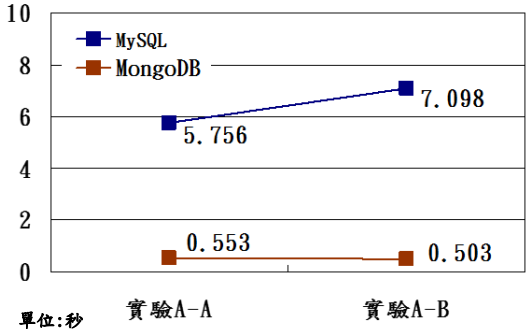

圖 9、效率測量-實驗 A

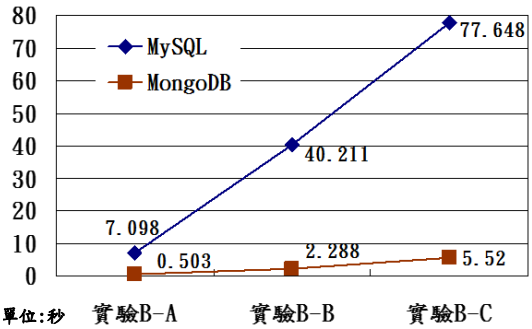

圖 10、效率測量-實驗 B

從圖 9、圖 10 可以發現 MongoDB 的查詢時間不會隨著資料數量增加而成長，查詢時間也遠小於 MySQL。當查詢筆數增加時，MongoDB 的查詢時間也遠小於 MySQL。綜合以上結果，MongoDB 在查詢效率上遠比 MySQL 來的高，較適合本論文的實驗。

4.2 篩選方法測量與分數計算

為了瞭解各項篩選方法對於正確性的影響，將每個篩選方法各自實驗，以了解篩選方法表現。實驗使用 udn 新聞網 1 月至 3 月新聞共 200 篇，測量結果如表 2。

表 2、篩選方法測量結果

篩選方法	正確篇數
Word length	138
Average distance	106
Count	76
Area	146
Common word	65

表 3、篩選方法分數分配

方法 \ 名次	1	2	3	4	5
Word length	16	12	8	4	2
Average distance	16	12	8	4	2
Count	16	12	8	4	2
Area	8	6	4	2	1
Common word	8	6	4	2	1

依照測量結果，正確性最高前三名分別是 Area、Word length 及 Average distance，因此我們給予較高的配分，而因為非所有的地點皆有面積，可計算的地點只有約 32%，所以將 Area 分數略為調降，補上第四名的 Count，而每項篩選方法我們只取各項分法的前 5 名給予配分，例如:A、B、C 三者的面積大小排序為 B > C，而 A 無法計算，則 A、B、C 在 size 這項的分數則分別是 0、8、6。整體分數分配如表 3。

先將 token 分別用各個方法評分，最後再加總，取分數最高者當作最佳地點，根據表，如當總分相同時則優先為:字串長度 (Word length) 高於平均距離(Average distance)高於出現次數 (Count)，計算結果範例如表 4。

4.3 正確率測量結果

本實驗分為三種做法:

- 做法一:使用 Count、Average distance 及 Common word 三個方法來計算分數，取分數最高者做為結果。
- 做法二:做法一再多加上 Area 來計算分數，取分數最高者做為結果。
- 做法三:用 token 之間的包覆關係來決定 token 是否是合適的地點，捨棄不合適 token，剩下的 token 用五種篩選方式計算分數，取分數最高者做為結果。

在判定實驗結果是否正確時，我們採用多重答案，例:假設該篇文章的發生地點為台北市政府，實驗結果如為「台北市政府」、「市府路 1 號」或「台北市信義區」，我們皆視為正確，但如果為桃園市則視為錯誤。三種做法皆取用 udn 新聞網 2015 年 1 月至 3 月社會及地方新聞共 1921 篇，測量結果如圖 11。

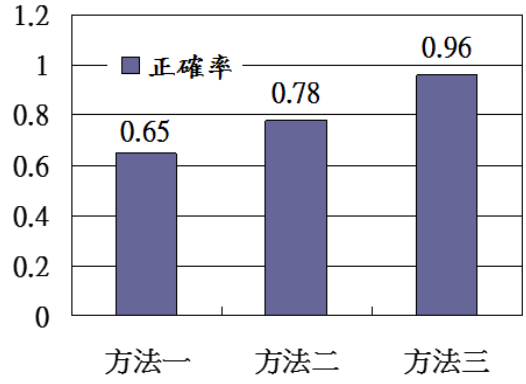

圖 11、實驗結果

由圖 11 所示，方法三實驗之正確率達到 96%，合理推測使用 token 之間的包覆關係來篩選合適的地點效果相當的卓越，主要的錯誤大部分在於所有在地圖資料庫中皆比對成功的 token，全部都不是正確地理名詞，或許可以再新增其他方法來剔除這些錯誤的 token。

5. 結論

目前網路上存在著許多文章，數量多到無法以人工的方式來辨別所在地，而現今有許多可靠性高的開源地圖資料，本研究便與之結合，方便使用者自動化找出文章地點。

文中使用了 CGAL 函式庫來貼近一般人觀看文章時，看到復數個地名十會有的想法，像是兩個區域之間的大小差別以及包覆關係，像這種方式叫符合人們的思考方式，我們使用了群眾地理資料的 OpenStreetMap 來當作地圖資料庫來判斷地理名詞。另外我們設計了幾種方法來增加判斷的準確性，像是距離遠近、文字長度等，使用配分的方式找出最合適的地點，從量測結果可以看出，主要影響標記地點的有:token 關係、面積大小及距離遠近，而常用字的比重最低，因為這通常只代表該地名在大多數情況下的重要性，不一定是該篇文章的重要性。而我們的實驗準確性達到 96%，未來可再增加判斷方法來提高正確率，或是搭配顯示介面讓使用者方便獲取資料，相信可以讓地理資訊的運用更加完善。

參考文獻

[1] OpenStreetMap: a free wiki world map https://www.openstreetmap.org/

[2] MongoDB: a no-sql database https://www.mongodb.org/

[3] CGAL: a commercial license from GeometryFactory. http://doc.cgal.org/4.2/CGAL.CGAL/html/index.html

[4] JIEBA:結巴分詞 https://github.com/fxsjy/jieba

[5] CKIP Client:中研院斷詞系統 http://ckipsvr.iis.sinica.edu.tw/

[6] Chuang Wang, Xing Xie, Lee Wang, Yansheng Lu and Wei-Ying Ma, "Detecting geographic locations from web resources," Proceedings of the 2005 workshop on Geographic information retrieval, pp. 17--24

[7] C. DIgnazio, R. Bhargava, E. Zuckerman, and L. Beck, "Cliff-clavin: Determining geographic focus for news," In NewsKDD: Data Science for News Publishing, at KDD 2014, 2014.

[8] David A. Smith and Gregory Crane, "Disambiguating Geographic Names in a Historical Digital Library," Proceedings of the 5th European Conference on Digital Library, pp. 127--136, 2001.

[9] Einat Amitay, Nadav Har'El, Ron Sivan, Aya Soffer, "Web-a-Where: Geotagging Web Content," Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval, pp. 273--280, 2004.

[10] Erik Rauch, Michael Bukatin, and Kenneth Baker, "A confidence-based framework for disambiguating geographic terms," Proceedings of the HLT-NAACL on Analysis of geographic references, pp. 50--54, 2003.

[11] Geoffrey Brun, Catherine Dominguès, M.-D. Van Damme, "TEXTOMAP: determining geographical window for texts," Proceedings of the 9th Workshop on Geographic Information Retrieval, Article No. 17, 2015.

[12] Huifeng Li, Rohini K. Srihari, Cheng Niu, and Wei Li, "Location Normalization for Information Extraction," Proceedings of the 19th international conference on Computational linguistics, pp. 1--7, 2002

[13] Judith Gelernter, "Data Mining of Maps and their Automatic Region-Time-Theme Classification," SigSpatial Special 1(1), pp. 39--44, 2009.

[14] Kalev H. Leetaru, "Fulltext Geocoding Versus Spatial Metadata for Large Text Archives: Towards a Geographically Enriched Wikipedia," D-Lib Magazine, 18(9), pp. 5, 2012.

[15] Kalev Leetaru, Shaowen Wang, Guofeng Cao, Anand Padmanabhan, Eric Shook, "Mapping the global Twitter heartbeat: The geography of Twitter," First Monday 18(5).

[16] Kevin S. McCurley, "Geospatial Mapping and Navigation of the Web," Proceedings of the 10th international conference on World Wide Web, pp. 221--229, 2001.

[17] Asanobu Kitamoto, "Toponym-based geotagging for observing precipitation from social and scientific data streams," Proceedings of the ACM multimedia 2012 workshop on Geotagging and its applications in multimedia, pp. 23--26, 2012.

[18] Martin Ester, Hans-Peter Kriegel and Jörg Sander, "Spatial Data Mining: A Database Approach," Proceedings of the 5th International Symposium on Advances in Spatial Databases, pp.47--66, 1997.

[19] Martin Ester, Hans-Peter Kriegel and Jörg Sander, "Spatial Data Mining: Database Primitives, Algorithms and Efficient DBMS Support," Data Mining and Knowledge Discovery 4(2), 2000.

[20] Seng Chuan Tay, Wynne Hsu, Kim Hwa Lim and Lee Chen Yap, "Spatial data mining: clustering of hot spots and pattern recognition," In International Geoscience & Remote Sensing Symposium 6, 2003.

[21] Tatjana Scheffler, Johannes Gontrum, Matthias Wegel and Steve Wendler, "Mapping German Tweets to Geographic Regions," Proceedings of the NLP4CMC Workshop at Konvens, 2014.

[22] Y. Morimoto, M. Aono, M. E. Houle, K. S. McCurley, "Extracting spatial knowledge from the web, Applications and the Internet," Proceedings of the Applications and the Internet, 2003.