國立中山大學資訊工程學系

專題製作期末報告

專題名稱

空間關鍵字查詢之行動裝置應用程式

Mobile Apps for Spatial Keyword Searching

指導教授

張玉盈 教授

專題小組成員

B023040013　林嘉羽

B023040024　林聖洋

摘要

由於資訊傳遞越來越方便而衍生的資訊爆炸，雖然讓「搜尋」越來越容易，但同時也變得越來越有負擔。我們的專題目標即是希望能夠解決「搜尋太過困難」的問題，讓使用者可以一次丟好幾個關鍵字，讓搜尋引擎去找到更精確的結果。

關鍵詞：空間關鍵字、屬性資料、壓縮、應用程式、資料庫、分析

Abstract

The transfer of the information becomes more and more convenient day by day . And unconsciously , the information explosion has became a problem and bothered the internet users . Although nowadays “ Searching “ is really easy and powerful , but the burden that comes with it is growing . And that is something we shouldn't ignore .

That so , the ultimate purpose of our project is to solve the “ Searching has became harder and harder “ problem . We hope that we can let user to key in a bunch of keywords while they are searching anything . And the search engine can find some more precise results due to multiple keywords .

Keywords : Spatial keywords , Property data , Compression , App ,
Database , Analysis

目錄

摘要 2

第一章、簡介 6

第二章、研究方法 8

第三章、研究過程 10

利用演算法處理空間關鍵字 10

利用演算法為屬性資料建 tree 11

利用演算法進行座標壓縮 13

設計程式架構 14

第四章、本學期結語 18

附錄、暑假進度計畫 19

參考文獻 20

圖次

Fig.1 Fast Nearest Neighbor Search with Keywords , page 8 .

Fig. 2 (a) Shows the locations of points and (b) Gives their associated texts , page 10 .

Fig. 3 Example of bit string computation with L = 5 and m = 2 , page 11 .

Fig. 4 Example of an IRR-Tree . (a)Shows the MBRs of the underlying R-Tree (b) Gives the signature of the entries , page 11 .

Fig. 5 Example of an inverted index , page 12 .

Fig. 6 Converted values of the points in fig. 2a based on Z-curve , page 13 .

Fig. 7 Relative values of the points in fig. 6 , page 14 .

Fig. 8 程式架構 , page 15 .

第一章　簡介

隨著時代的變遷，越來越發達的互聯網狹帶著持續爆炸性成長的資訊量滲透到我們平日生活圈的每個角落。從過去找個資訊得實地探勘、前往圖書館人工地毯式搜索，進步到搜尋引擎問世後輕輕鬆鬆輸入關鍵字就能一覽天下事。而時至今日，資訊量已經大到 Google 一個關鍵字「飯店」，就會在一秒內得到 26,500,000 筆 (*1) 搜尋結果。資訊唾手可得當然是件美事，但這麼多的資訊，一個正常的人類又怎麼可能輕鬆消化找到理想標的物呢？

除了透過增加搜尋限制（如日期、語言、地區）進行進階檢索的方式來縮小範圍以外，另一個方式就是增加搜尋關鍵字的數量，一次給予多個關鍵字，去搜尋能夠符合每一個關鍵字的物件。

但今天搜尋的平台轉換到地圖上還行得通嗎？地圖上的資訊多半就是地名、店家名、地址，假如說今天我要找一間距離我最近且「免費提供早餐、有泳泳池、有健身房、禁止吸煙、允許寵物進入」的飯店，我該怎麼進行搜尋？相信你和我一樣都會打開地圖，從最近的幾家飯店一家一家人力篩選，找到理想的飯店為止。難道沒辦法加速這個過程嗎？答案是可以的。

在原有的物件其名稱以及二維坐標資訊以外，我們再為每一個物件添加空間資訊，讓每個物件都擁有特定的屬性，如免費提供早餐、有游泳池、有健身房等等。如此一來當我們在輸入多個關鍵字進行搜尋時，就能夠輕鬆的去搜尋出符合我們要求的目標物。

註記 1：為 2016 年 6 月 4 日在 Google 搜尋引擎上測試的結果。

但事與願違，這個解決方法理論上是正確的，但在實作時便會發現到，因為地圖上無數個物件各自擁有數以百計的屬性資訊，資料庫太過於龐大的狀況下，檢索速度就成反比地呈現相當低落的狀態了。而這個棘手的問題，就是我們此次專題的任務——「盡可能地將空間關鍵字搜尋的速度提升，同時還能篩選出距離最近的結果」。讓一次使用多個關鍵字在地圖上尋找理想目標物不再是個夢想，也不會是個讓人用了一次就決定土法煉鋼人工篩選的龜速搜尋引擎。

而我們將用行動裝置上的應用程式，來呈現此次專題的成果。

第二章　研究方法

除了簡介所提到空間關鍵字檢索有效率問題的疑慮以外，現行在搜尋時可能所會使用到的一些演算法在套用到空間關鍵字時，也會有難以忽略的 false hits 問題。

因此研究過程主要會分成三大部分，加速及優化空間關鍵字進行搜尋的速度、降低 false hits 出現的機率、以及最終用來呈現之行動裝置應用程式的撰寫。而再以這三個部分去做細部的分工。

1. 研究本次主題「優化空間關鍵字鄰近目標查詢準確度及速度」之相關演算法的論文內容

Fast Nearest Neighbor Search with Keywords
Problem Definitions
The IRR-Tree
Solutions Based on Inverted Indexes
Other Relevant Work
Merging and Distance Browsing
Spatial Inverted List
The Compression Scheme
Building R-Trees

Fig.1 Fast Nearest Neighbor Search with Keywords

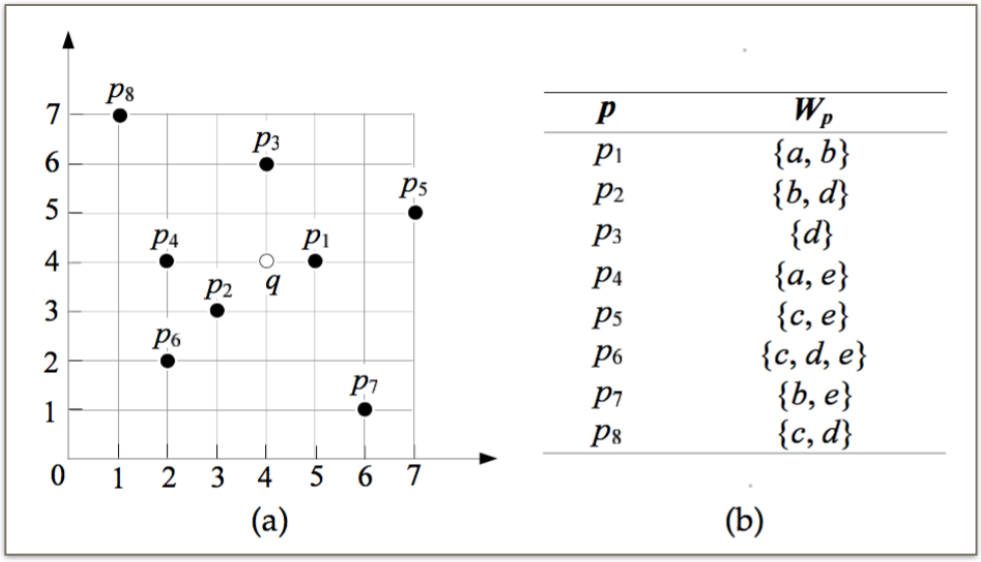
2. 討論呈現方式

3. 設計程式架構
4. 手機應用程式設計及撰寫
5. 後端空間資料庫設計及撰寫
6. 利用演算法提升速度 — 座標資訊壓縮為一維相對座標
7. 利用演算法提升速度 — 建立 R-Trees 簡化搜尋過程
8. 優化 R-Trees 降低 false hits 的出現機率
9. 最終調整及結果呈現

第三章　研究過程

一、利用演算法處理空間關鍵字

最初的工作就是要先讓我們的物件擁有自己的屬性資料（空間資料），而不是只有物件名稱、地址資訊、座標資訊。圖二 (a) 中，q 點即為使用者、初始原點的位置，其餘 p1 ~ p8 則為座標平面中其他的物件。而在圖二 (b) 中，可以看到每個 p 點都有屬於自己的屬性資料 Wp，p1 中有 a,b 兩種特性，p2 中有 b,d 兩種特性，以此類推。



p	W_p
p_1	{a, b}
p_2	{b, d}
p_3	{d}
p_4	{a, e}
p_5	{c, e}
p_6	{c, d, e}
p_7	{b, e}
p_8	{c, d}

Fig. 2 (a) Shows the locations of points and (b) Gives their associated texts

而在圖三中，可以看到我們利用了 [2] superimposed coding (SC) 來處理我們的屬性資料。首先先為每一個屬性資料 a,b,c,d,e 建立相對應 L bits 長的的 bit string，並將 bit string 中的每一個位元都歸零。第二步驟為利用 SC 對每一個屬性資料進行 m 次「隨機選取其中一個位元並設定為 1」的動作，並在最後確認每一個屬性資料都擁有不同的 bit string 表示式。

word	hashed bit string
a	00101
b	01001
c	00011
d	00110
e	10010

Fig. 3 Example of bit string computation with L = 5 and m = 2

二、利用演算法為屬性資料建 tree

接下來就是要將剛剛 bit string 化的屬性資料以 tree 的方式建立起一個能夠減輕搜尋量負擔的結構。在開始建樹結構之前，要先將每一個物件 p 中的屬性資料做合併，如擁有 a,b 兩種特性的 p1，利用 OR 的方式合併屬性資料之後，p1 的 bit string 表示式就成為了 01101。

下圖四，示範的演算法為 IRR-Tree。圖四 (a) 為利用 MBRs 來將座標平面上的各個物件分類，兩兩為一組、兩組再為一組，以此規則類推，最後便能建構出一個樹狀的結構如圖四 (b)。如此一來在進行搜尋時，便可以利用想搜尋的合併屬性資料 bit string 表示式，以更快的速度搜尋到符合的目標。

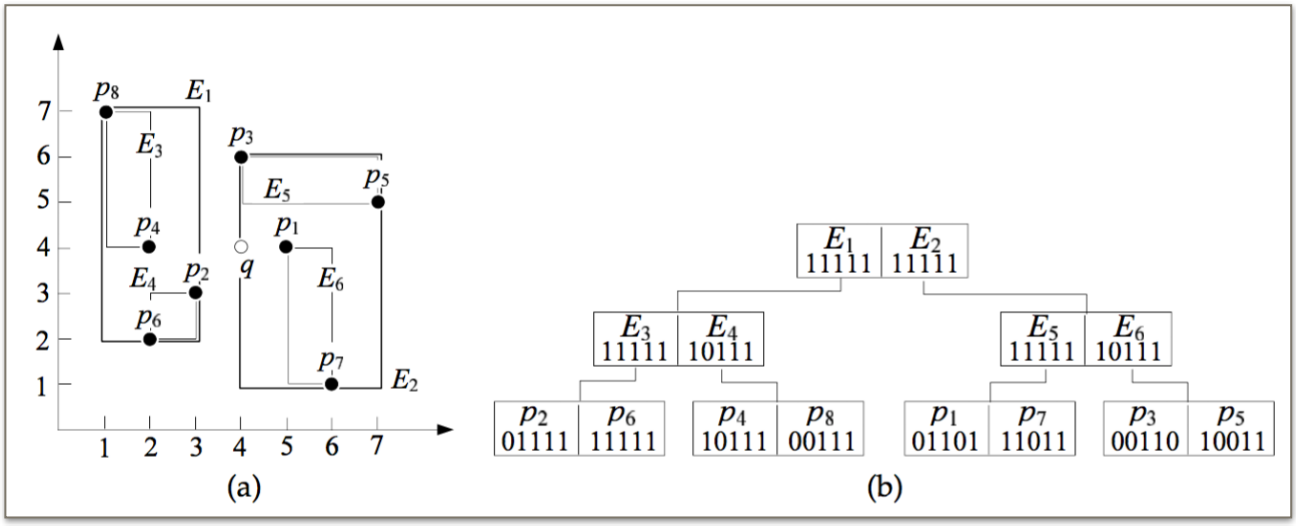


Fig. 4 Example of an IRR-Tree .
(a) Shows the MBRs of the underlying R-Tree (b) Gives the signature of the entries

可惜的是，如此方便的檢索方式必有他的缺陷存在，而 IRR-Tree 帶來的缺點，就是他會有不容忽視的 false hits 機率。由於各個物件擁有的合併屬性資訊、多重關鍵字檢索的合併屬性資訊所使用的 bit string，為多個屬性內容 bit string OR 處理後的結果，因此可能會碰上不一樣的合併屬性資料卻擁有相同的 bit string 表示式這種狀況。如圖三所示，a,b,c 三者的 bit string 表示式和 a,b,c,d 四者的 bit string 表示式，同為 01111。

而我們要怎麼克服這個問題呢？我們的解決方式是建立一個 inverted list 如圖五，列出擁有各個屬性資料 a,b,c 之物件 p1,p2。在 Tree 檢索完畢之後，便利用 inverted list 來檢驗所得到的結果物件，是否真的擁有所求的幾種屬性資料。另外可以注意到 inverted list 的地方是順序排入的，如此也可以降低檢驗的次數，假如說今天檢驗的目標物件是 p3，那麼找到 p4 時就可以知道 p3 是不符合我們所要求的了。

word	inverted list
a	p_1 p_4
b	p_1 p_2 p_7
c	p_5 p_6 p_8
d	p_2 p_3 p_6 p_8
e	p_4 p_5 p_6 p_7

Fig. 5 Example of an inverted index

三、利用演算法進行座標壓縮

而接下來要進行的就是壓縮座標的部分，原始的二維座標檢索對我們來說還是太過繁複了，若能夠減輕座標檢索的複雜度，讓每一個被搜索物件都會擁有的座標資訊其資訊量降到最低，將能有效的提升檢索時的效率。

首先要動刀的地方就是將原先的二維座標，轉換表示為一維座標。

p_6	p_2	p_8	p_4	p_7	p_1	p_3	p_5
12	15	23	24	41	50	52	59

Fig. 6 Converted values of the points in fig. 2a based on Z-curve

在圖六的地方，各個物件下方的數值即為經過 Z-curve 演算過後所得到，以原點為 0 基準的新一維座標，並以遞增的方式排列表示。轉換過後的座標資訊雖然不能夠如原先的二維座標精確的表示出位置，但仍然能夠沒有誤差的給予我們各個物件相對距離遠近的回饋，而這樣的資訊回饋在搜索上就已經足夠了。

除此之外，我們還可以進一步的減少已經處理過的一維座標資訊量，其關鍵就在於「相對位置」。圖六的座標資訊已經是遞增排序的資料，接下來我們只要將「每兩個座標之間的距離差距」計算出來，用以取代原先的一維座標，即可以達到減少座標位元數的目的（尤其在有相當多物件時，減少之效果相當可觀）。

同樣的，這樣的座標壓縮方式也不會影響到每個物件之間相對距離的關係，因此在檢索上也是可行的。結果就如下圖七所示。

p_6	p_2	p_8	p_4	p_7	p_1	p_3	p_5
12	3	8	1	17	9	2	7

Fig. 7 Relative values of the points in fig. 6

四、設計程式架構

由於我們想要以真實的「地圖」最直接、一目了然地展現最終的演算法撰寫成果，因此選擇了應用程式這個模式。而在便利性、觸及度上，現今又以行動裝置最為廣泛，故最後便以行動裝置上的應用程式作為撰寫目標，其中又以觸及人數較多、且 Google Map 行動裝置套件原生支援的 Android 為優先完成目標。

我們決定使用由 Android 開發廠商 Google 本身所發佈的 Android Studio 作為開發平台，除了因為是原廠直接支援以外，Andrioid Studio 更是許多前輩推薦的撰寫平台。且因為我們打算使用 Google Map 套件在我們的成果展示中，因此能最快速使用 Google Map 套件的 Android Studio ，就顯得更為方便了。

不過因為目前尚無法確定 Google Map 在應用程式的套件上，是否有支援讓開發者使用地圖上每個物件其空間關鍵字的部分，只知道他能夠實作如使用者定位、目標設定等等的基礎功能。因此我們打算以原先的備案「資料庫架設

輔以 Google Map」，也就是地圖介面使用 Google Map、後方的空間關鍵字資料庫由我們自己建立、架設，為主要的準備目標。

圖八即為我們的程式結構。

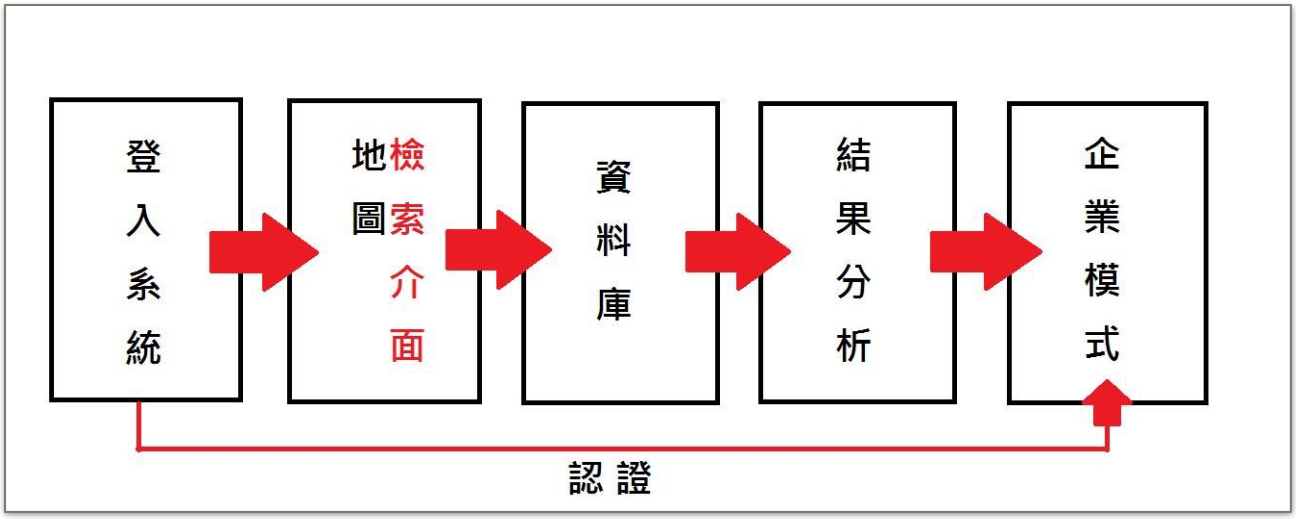


Fig. 8 程式架構

第一層、登入系統：

其實現今的行動裝置多半都擁有能夠讓應用程式暫存程式資料的功能，因此我們登入系統的目的並不是儲存用戶的瀏覽、搜尋紀錄，而是要讓程式架構中第五層的「企業模式」，能夠在使用上有更進一步的認證。

而在這邊會出現帳號資料輸入視窗、註冊選項、忘記帳號資料選項、已無帳號方式使用此軟體等等功能選單。

第二層、地圖（檢索介面）：

主要會以一個 Google Map 為主，一些輔助功能的 icon 為輔。在這邊我們可以確定用戶的所在位置並將他顯示出來，又或者是讓使用者選擇想要出發的原點位置。同時也能讓用戶輸入想要搜尋的一個以上關鍵字們，以進行我們的主題。

第三層、資料庫：

這部分並不會讓使用者看到，是隱藏的一層。主要是因為 Google Map 可能沒有提供行動裝置應用程式開發者空間關鍵字的使用，因此我們將會自己建立一個以中山大學為中心，附近上百個物件的屬性資訊資料庫。

而同時這裡也會有另一個資料庫，儲存每一個物件目標被選擇時，使用者連結過來的關鍵字是哪些，這些資料在第五層將被用上。

第四層、結果分析：

空間關鍵字查詢最近目標之演算法所在頁面。而在這邊會展示出使用者輸入關鍵字後的結果分析。以距離近遠次序列出各個符合的物件，並顯示該物件、該物件擁有的屬性、位置等等的資訊。

同時在這邊我們會將搜尋出來的結果及最終選擇的目標物件，其關鍵字資料回饋給第三層的第二個資料庫，將資料儲存在其中。

第五層、企業模式：

登入的使用者可以在這個介面進行我們所提供的其他進階功能。如透過認證的方式，來認領你所擁有的店家、認領店家成功後可以查看該店家被搜尋的主要關鍵字分析等等，主要目的在於幫助店家能夠快速的掌握消費者選擇自己的目的。

第四章　本學期結語

就如簡介時所提到的，由於現在流通的資訊量越來越龐大，提升檢索速度必然是個勢在必行的任務。而其中又以空間關鍵字為我們最關注的領域，相信 Google 也已經在為自己旗下的 Google Map 準備類似的功能了。

目前已經大概瞭解了 Fast Nearest Neighbor Search with Keywords 其演算法中各個步驟的用意、整體程式架構也已經草創完畢，接下來的目標就是成功的將它實作在我們的應用程式上。期待最後的結果除了能夠達到我們的預期之外，還有時間能夠完成其他我們還沒有與我們邂逅的擴充功能。也希望最後的專題成果能夠對現在的空間關鍵字領域有一定的正面影響。

附錄　暑假進度計畫

第一階段：

學習 Android Studio 使用方式，並先完成除了演算法及資料庫以外，介面的部分。預計將會進行兩個星期。

第二階段：

實作演算法。因為演算法的部分是我們專題的核心內容，所以希望能儘早開始，其他部分則能夠因應演算法的撰寫速度來調整進度。預計將會進行四至五個星期，並和後面幾個部分進度重疊。

第三階段：

實作資料庫。這方面是指導教授張玉盈教授實驗室的主要研究項目之一，由於我們沒有資料庫撰寫的經驗，可能得花一段時間先認識它。這部分尚無法預計進行時間。

第四階段：

實作登入系統及企業模式。因為登入系統和企業模式的部分和我們的專題主題較無相關，比較像是附加的功能，因此會將這個部分擺到進度的最末端，以避免主角沒有塑造完畢配角已經在一旁納涼的窘境。預計花費三至四星期。

第五階段：

後期修飾。我們希望能在暑假結束以前完成專題的大部分內容，而將下學期專題發表以前的時間用來做最後的修飾。

參考文獻

[1] Yufei Tao and Cheng Sheng . Fast Nearest Neighbor Search with Keywords . IEEE Transactions on knowledge and data engineering , VOL. 26 , NO. 4 , April 2014 .

[2] I.D. Felipe, V. Hristidis, and N. Rishe,"Keyword Search on Spatial Databases," Proc. Int'l Conf. Data Eng. (ICDE), pp. 656-665, 2008 .