國立臺灣大學工學院工程科學及海洋工程學研究所

碩士論文

Department of Engineering Science and Ocean Engineering
College of Engineering
National Taiwan University

Master Thesis

使用社交圖與情境感知之行動餐廳推薦系統

A Context-aware and Social Graph based Restaurant Recommender System for Mobile Devices

郭羿呈

Yi-Cheng Guo

指導教授：黃乾綱 博士

王勝德 博士

Advisor: Chien-Kang Huang, Ph.D.

Sheng-De Wang, Ph.D.

中華民國 101 年 7 月

July, 2012

致謝

想到剛入學時，懵懵懂懂的模樣，時光匆匆流轉，心境不斷改變。起初也曾期許自己，獨立思考，不隨波逐流；但現在才知道，研究歷程大抵相同，路上的箇中三昧，卻只能親身體會。

這篇論文是我兩年耕耘的成果，每一次回頭審視，總為那些不夠完美的部分，感到可惜；但也為得到的些許成果，感到欣慰。在撰寫論文的日子裡，漸漸發現我不只是做研究，更是學習如何在這個環境生存，多虧朋友、女友總是在我失望低落時，帶給我一點溫暖；沒有你們，我無法完成這篇論文。

在台大求學的日子雖然辛苦，卻收穫豐碩。首先，感謝工科系黃乾綱教授、電機系王勝德教授的共同指導，讓我能在碩士生涯中，探索自己喜歡的研究主題。這兩年裡，不管是研究、報告或邏輯思考上，都因為老師的教誨，讓我有所成長；故在此刻，我要向老師致上最深的謝意。

此外，感謝瑋強、士綱與振和同學，總是在我陷入困境時，耐心聆聽我的煩惱，為我解決各種疑難雜症。感謝元鴻學長、嘉德、漢威與証凱學弟，在論文和口試上給予許多協助。感謝實驗室的同學們，如果沒有大家互相扶持，這段路會更加艱辛。感謝資工系鄭卜壬教授、工科系張瑞益教授在口試時，提供寶貴的建議。

在碩士生涯中，我也做過一些蠢事，但卻充實而美好；感謝鈺雯、有順、雋凱、士綱、蔚萱、美智、建胤、小強、宗博、傑仁，陪我一起追逐夢想。感謝電腦研習社的朋友們，正因你們的鼓勵，我才能堅持到底。

最後，感謝我親愛的父母，總是無私奉獻，不求回報，讓我能無後顧之憂的完成論文；希望我畢業後的成就，能讓你們感到驕傲。

郭羿呈

中文摘要

隨著行動裝置與行動網路日益普及，人們可以隨時隨地存取的資訊激增；如何解決當前資訊超載（Information Overload）問題，並提供個性化推薦（Personalized Recommendation）服務是一項重要的研究議題。本論文利用 Facebook 開放圖（Open Graph）的打卡資料（Check-ins）設計一個行動餐廳推薦系統；以協同過濾法（Collaborative Filtering）為基礎，從個別使用者偏好總結團體偏好，實現團體推薦服務；並考慮社交圖（Social Graph）與情境資訊（Contextual Information）提升推薦品質；考慮的情境有位置、距離、年齡、性別指標、時間、星期、月份、同伴人數與同伴類型。本論文還設計一個方法，單獨評估位置與距離情境帶來的影響。

本論文的實驗資料是從 Facebook 徵集 69 名受測者，收集 2010/8/15 至 2012/4/30 期間，3928 名使用者對 2691 家餐廳的 8264 次打卡。實驗結果顯示，本系統在中、長距離（3 到 5 公里）的情境下，準確度相較於基於流行性推薦有顯著成長，成長率約 38%。這意味著，如果使用者尋找餐廳所設定的範圍比較大，相較於基於流行性推薦，本系統可以產生更好的推薦結果。

關鍵字：推薦系統、餐廳推薦、團體推薦、協同過濾，社交圖、情境感知、行動裝置

英文摘要

With the increasing popularity of mobile devices and mobile networks, people can get a soaring amount of information, anywhere, anytime. How to solve the problem of the current information overload and provide personalized recommendation services is an important research topic. This thesis exploits the check-ins of Facebook Open Graph to design a mobile restaurant recommender system, which is based on collaborative filtering. The system summarizes the group preferences from individual users check-in in order to provide group recommendation services. Furthermore, the system considers social graph and contextual information to enhance the recommendation quality. These contextual information includes location, distance, age, sex index, time of day, weekday, month, number of companion and type of companion. In this thesis, we also proposed a method to evaluate the impact of location and distance context.

Our experimental data is collected from the 69 volunteers in Facebook, which includes the 8264 check-ins. These check-ins are contributed by 3928 users in 2691 different restaurants from 2010/8/15 to 2012/4/30. The experimental results reveal that the accuracy of our system can be increased by approximately 38% while suggest restaurants within the area of 3-5 km radius, compared to popularity-based recommendation. It means that the proposed system can provide better

recommendations than popularity-based recommendations, if the user asks for a restaurant suggestion in a larger area.

Keywords: Recommender System, Restaurant Recommender, Group Recommendation, Collaborative Filtering, Social Graph, Context-awareness, Mobile Device

目錄

致謝 I

中文摘要 II

英文摘要 III

目錄 V

圖目錄 VII

表目錄 VIII

1 導論 1

1.1 動機 1

1.2 目的 2

1.3 論文架構 2

2 背景與相關研究 3

2.1 情境感知（Context-aware） 3

2.2 社交圖與開放圖 4

2.2.1 社交圖（Social Graph） 4

2.2.2 開放圖（Open Graph） 5

2.2.3 打卡（Check in） 6

2.2.4 Graph API 7

2.3 推薦演算法 8

2.3.1 協同過濾法（Collaborative Filtering Approaches） 9

2.3.2 基於內容法（Content-based Approaches） 11

2.3.3 混合法（Hybrid Approaches） 11

2.3.4 結合情境之協同過濾法 12

2.4 行動餐廳推薦系統 13

3 使用社交圖與情境感知之行動餐廳推薦系統 14
3.1 資料定義 15
3.2 問題定義 16
3.3 解決方案 16
3.3.1 系統架構與資料模型 16
3.3.2 社交圖與情境感知推薦 18
3.3.3 社交圖與情境資訊比較 21
4 實作 25
4.1 雲端計算 25
4.2 系統流程 27
4.3 使用者介面 29
5 實驗與評價 31
5.1 資料收集 31
5.2 選擇相似度測量法 37
5.3 調整係數 41
5.4 評價 45
6 結論 49
6.1 總結貢獻 49
6.2 未來工作 50
參考文獻 51
附錄 53

圖目錄

圖 2.1 描繪社交圖 5

圖 2.2 烹調食譜的開放圖.....6

圖 2.3 描繪社交圖與開放圖.....8

圖 3.1 系統架構與資料模型.....17

圖 3.2 系統通訊過程 18

圖 4.1 系統流程圖 28

圖 4.2 使用 WEB 介面瀏覽地圖和尋找餐廳的截圖.....29

圖 4.3 使用 WEB 介面請求建議的截圖 30

圖 4.4 使用 MOBILE 介面的截圖.....30

圖 5.1 收集到的所有餐廳之分布圖.....33

圖 5.2 統計收集到的餐廳打卡資料.....35

圖 5.3 各距離參數所涵蓋的範圍（以台北車站為中心）.....37

圖 5.4 本論文使用 LOOCV 的驗證過程.....38

圖 5.5 三種相似度測量法之推薦準確度.....39

圖 5.6 僅參考位置與距離參數的各距離參數之準確度.....40

圖 5.7 僅參考位置與距離參數的各距離參數之準確度(篩選推薦數量小於該距離參數之餐廳數量的情況）.....40

圖 5.8 情境相關性測量模型之各項係數值的平均命中位置（紅點為最佳係數）.....44

圖 5.9 各方法與距離參數之推薦準確度.....47

圖 5.10 各方法與距離參數之推薦準確度（篩選推薦數量小於該距離參數之餐廳數量的情況）...47

圖 5.11 本系統對基於流行性推薦的準確度成長率（篩選推薦數量小於該距離參數之餐廳數量的情況）.....48

表目錄

表 2.1 使用者配置文件與情境資訊.....4

表 2.2 從 GRAPH API 取得的臺灣大學之物件.....7

表 3.1 *TC_RELEVANCEMATRIX*.....23

表 5.1 收集到的各項資料之數量.....32

表 5.2 收集到的各項資料之比率.....32

表 5.3 收集到的各項資料之平均值、中位數與標準差.....34

表 5.4 總結情境相關性測量模型的各項最佳係數.....45

表 5.5 各情境維度的影響程度.....46

1 導論

本研究是關於利用 Facebook 開放圖（Open Graph）的打卡資料（Check-ins）為行動裝置設計一個餐廳推薦系統，實現團體推薦服務，並考慮社交圖（Social Graph）與情境資訊（Contextual Information）提升推薦品質。接下來，我將介紹本研究的動機與目的。

1.1 動機

近年來，隨著行動應用程式（App）風行全世界，各國政府與傳統產業也開始研究如何提供相關服務；由於數位內容將成指數成長，其中幫助使用者找到所需資訊的適地性服務（Location-based Service）與個性化推薦服務（Personalized Recommendation Service）是一項非常重要的研究工作。

其中協同過濾法（Collaborative Filtering）是現今最成功且被廣泛應用的推薦系統之一，但是也面臨很多的問題，例如：如何對新使用者進行推薦或如何推薦新項目給使用者之冷啟動（Cold Start）問題；在任何推薦系統中，已經評分的項目通常比需要推薦的項目數量少很多，而導致精確度低落之稀疏性（Sparsity）問題，面對日益增多的使用者，資料量急劇增加之可擴展性（Scalability）問題等[1]。

由於一般推薦服務主要針對個別使用者，餐廳和電影推薦則應該同時考慮幾個人的偏好，因此團體推薦服務是另外一項重要的研究議題。

除此之外，使用者偏好的推薦項目，在不同情境下會有所差異，而團體偏好也會隨著同伴類型改變，因此如何得到並利用情境因素提升推薦品質，是許多情境感知之推薦系統努力的目標。

1.2 目的

因此，本研究目的是利用 Facebook 開放圖（Open Graph）的打卡資料（Check-ins）為行動裝置設計一個餐廳推薦系統，實現團體推薦服務，並考慮社交圖與情境資訊提升推薦品質。

1.3 論文架構

本論文其他部分的架構如下：第二章概述情境感知、社交圖、開放圖與推薦演算法的背景與相關研究，第三章闡述問題定義，提出解決方案，並介紹系統架構，第四章敘述實作技術與流程，並展示使用者介面；第五章評價這個系統；最後第六章是結論與未來工作。

2 背景與相關研究

在本章中，我將介紹與情境感知、社交圖、開放圖、打卡與推薦系統的定義與相關研究。

2.1 情境感知（Context-aware）

1994 年 Schilit 和 Theimer[2]在 *context-aware computing* 首次提出這個術語，在它們的定義中：

Context-aware computing is the ability of a mobile user's applications to discover and react to changes in the environment they are situated in.

許多研究者追隨他們的腳步，將情境感知計算應用到各式各樣的領域中；隨後，Dey[3]提供了一個更通用的定義：

A system is context-aware if it uses context to provide relevant information and/or service to the user, where relevancy depends on the user's task.

Woerndl 和 Schlichter[4]則在推薦系統中明確區分使用者配置文件（User Profile）與情境資訊（Contextual Information），如表 2.1：

User profile	Context
Rather static and somewhat longer lasting	Highly dynamic and transient
Stored in user profile	Not stored permanently
Implicitly observed or explicitly provided by user	Observed only, never manually entered by user
Example: user preferences or interests	Example: current location and time

表 2.1 使用者配置文件與情境資訊

另外，Schilit 等人使用環境(實體、社交)、自我(裝置狀態、生理、認知)與活動(行為、任務)的三維空間來描述情境。

經過以上介紹，我們已經對情境感知有了基本的了解；透過情境感知，我們可以讓系統更有效率。例如：人們在行動裝置上使用應用程式查看餐廳資訊，如果裝置支援 GPS，應用程式將可得到使用者當前的位置，並依餐廳遠近依序列出清單，人們將更容易發現自己偏好的餐廳，省時省力；系統也會受惠於人們翻找餐廳的次數減少，有效降低伺服器流量與負載，增進效能。

在餐廳推薦服務的例子上，人們通常對太遠的餐廳不感興趣，因此使用者位置將可減少候選餐廳的數量，大幅降低系統計算時間並提升準確度。

2.2 社交圖與開放圖

在本節中，我將概述社交圖、開放圖與打卡的定義，並介紹 Graph API。

2.2.1 社交圖（Social Graph）

2007 年 Facbook f8 conference 中，這個詞被使用於解釋 Facebook 平台而普及；而最早的社交圖出現在由 Philippe Bouzaglou 發表在哈佛經濟系網站的文章中，該文章第一次概述了一個完整的社交圖[5]。

社交圖說明了每個人在社會網路的交互關係；在社交圖上每個人經由被稱為節點的圓圈表示，交互關係經由被稱為邊的直線表示，如圖 2.1：

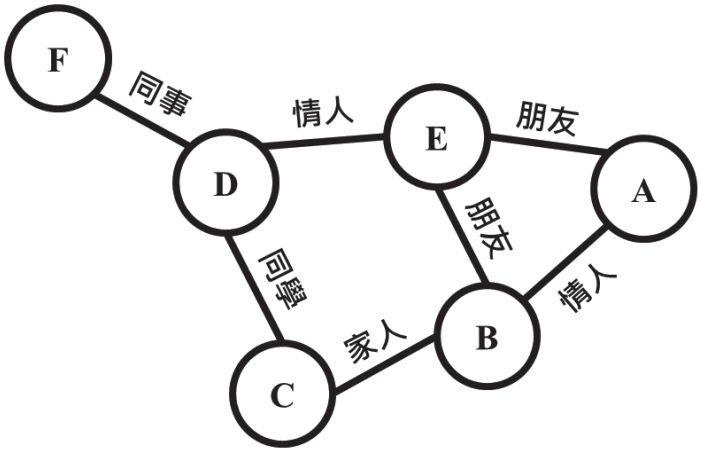


圖 2.1 描繪社交圖

2.2.2 開放圖（Open Graph）

Facebook 擴展了社交圖的概念使其可應用於虛擬非人之物件，而不僅僅是個人之間的關係[5]。

開放圖允許應用程式去塑造以動作（action）和物件（object）為基礎的使用者活動 。例如：慢跑的應用程式有能力去定義跑（action）一段路線（object）；閱讀的應用程式有能力去定義閱讀（action）一本書籍（object）；食譜的應用程式有能力去定義烹調（action）一份食譜（object），如圖 2.2：

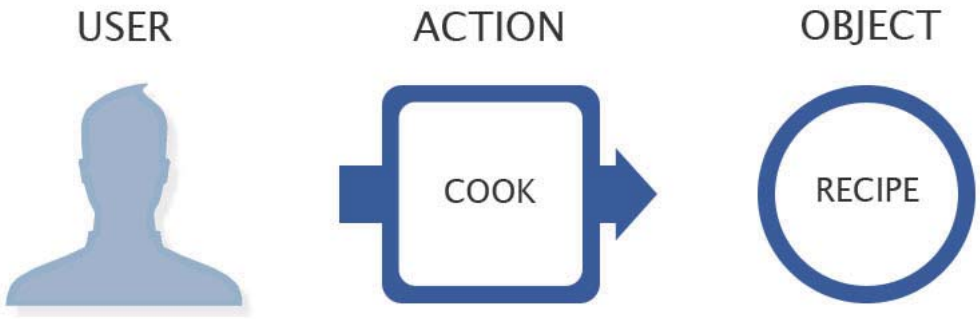


圖 2.2 烹調食譜的開放圖

動作是使用者執行在應用程式的動詞，物件是應用於動作的名詞，Facebook 建立了一套常見的動作和物件的使用案例，也提供了一套工具讓開發者自己客製化動作和物件。當使用者參與此應用程式，定義的動作會連接使用者與物件並被發佈到 Facebook[6]。

2.2.3 打卡（Check in）

許多社群網站，例如：Facebook、Twitter、Google+與 Foursquare，允許使用者對實體地點打卡並與朋友分享他們的位置。

使用者能在行動裝置上經由行動應用程式對一個具體位置打卡，而行動應用程式將使用行動裝置的 GPS 去找到當前的位置。

許多應用程式都有一個地方的按鈕，在那裡使用者可以看到一個可供打卡的附近地點清單，如果當前的位置沒有出現在附近地點清單，使用者可以直接加入此位置。一旦使用者打卡，他們可以選擇在社群網路上分享他們的位置[7]。

2.2.4 Graph API

Graph API 呈現了一個簡單、一致的 Facebook 社交圖之視圖（View），一致的表現圖中物件（例如：人們、照片、事件和頁面）和他們之間的連接關係（例如：朋友關係、分享內容和照片標籤）[8]。

在社交圖中任何的物件都有唯一的 ID，開發者可以訪問一個物件的屬性經由請求 https://graph.facebook.com/ID，例如：臺灣大學的 ID 是 106522332718569，所以你能在 https://graph.facebook.com/106522332718569 取得這個物件，如表 2.2


```
{
    "id": "106522332718569",
    "name": "\u53f0\u7063\u5927\u5b78",
    "picture":
"http://external.ak.fbcdn.net/safe_image.php?d=AQDf6wGxms-FZM6-&w=100&h=3
00&url=http\u00253A\u00252F\u00252Fupload.wikimedia.org\u00252Fwikipedia\u0
0252Fzh\u00252Fa\u00252Fa4\u00252FNational_Taiwan_University_logo.jpg&fallba
ck=hub_education&prefix=s",
    "link":
"http://www.facebook.com/pages/\u0025E5\u00258F\u0025B0\u0025E7\u002581\u00
25A3\u0025E5\u0025A4\u0025A7\u0025E5\u0025AD\u0025B8/106522332718569",
    "likes": 19473,
    "category": "University",
    "is_published": true,
    "is_community_page": true,
    "description": …
```


表 2.2 從 Graph API 取得的臺灣大學之物件

透過 Graph API 我們可以發現連接個別或團體使用者與物件的各種動作，以及每個人在社會網路的交互關係，如圖 2.3：

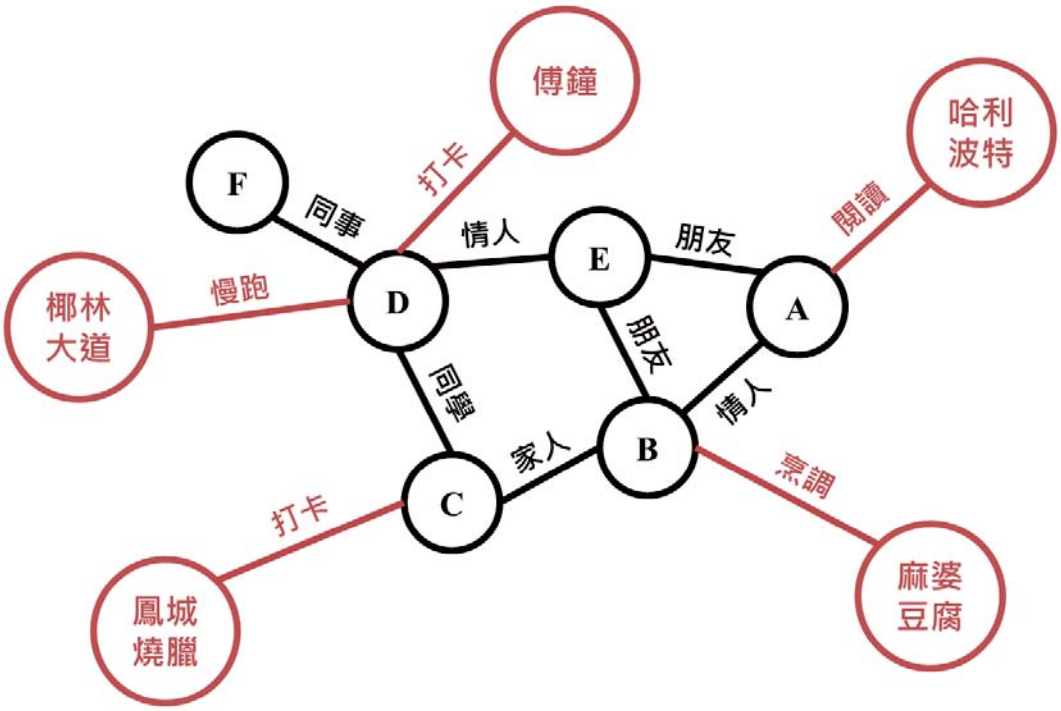


圖 2.3 描繪社交圖與開放圖

2.3 推薦演算法

隨著網際網路日益普及，數位內容呈現指數成長趨勢，使人們淹沒在資訊洪流之中，例如：Netflix 上有數萬部電影，Amazon 上有數百萬本書籍，Del.icio.us 上有超過十億的網頁收藏，這麼多的訊息，我們無法一一瀏覽，而傳統的搜尋引擎只能呈現給使用者一樣的排序清單，無法根據不同使用者偏好提供個性化服務，而個性化搜尋與推薦系統正是解決當前資訊超載的有效方法之一。

其中推薦系統是資訊過濾系統的子集，它試圖預測使用者將賦予他們還沒有考慮的項目（例如：書籍、音樂或電影）或社交元素（例如：個人或團體）評分或偏好。

在一個真實的推薦系統中需要考慮的項目可能會有成千上萬，甚至超過百萬，例如：Amazon、eBay 和 YouTube 等，使用者數量也會非常龐大，因此我們需要準

確且有效率的推薦系統。在激烈的競爭環境之下，現有推薦系統已經在電子商務領域取得卓越成果，但在不同領域之下，還需要各方先進和專家努力發展。

一個完整的推薦系統由三個部分組成，收集使用者資訊的行為紀錄模組，分析使用者偏好的模型分析模組與推薦演算法模組；行為紀錄模組負責記錄使用者行為，例如評分、購買或瀏覽等；模型分析模組負責分析使用者行為，以建立合適的模型來描述使用者偏好；推薦演算法模組負責即時從項目集合中篩選出使用者偏好項目進行推薦，此為推薦系統中最為核心的部分。

主要的推薦演算法可以分為協同過濾（Collaborative Filtering）、基於內容（Content-based）與混合法（Hybrid Approaches），接下來我將一一介紹，並說明結合情境感知之推薦系統的一些例子。

2.3.1 協同過濾法（Collaborative Filtering Approaches）

協同過濾法是目前最成功的推薦演算法之一，通常的做法是搜尋一大群人，從中找到與我們品味相近的一小群人。演算法會針對這些人偏好的其他內容進行計算，產生一個有序的推薦清單。

Goldberg 設計 Tapestry[9]是最早應用協同過濾法的系統，該系統允許人們根據自己對文件感興趣的程度添加標記，並利用此資訊為他人過濾文件。Konstan 等人設計的 GroupLens[10]則是應用在新聞篩選上，幫助讀者過濾其偏好的新聞內容，使用者看過內容後給一個評分，系統會將分數記錄下來供日後參考，不同於 Tapestry 僅在同一個系統中過濾，GroupLens 是跨系統的新聞過濾機制；除此之外，Tapestry 不會將同項目的評分總合起來，GroupLens 則會對同項目從不同使用者得到的評分加總。

協同過濾法可以分為兩類：基於記憶（Memory-based）和基於模型（Model-based）的演算法；基於記憶的演算法根據系統中所有被評分的項目進行預測；設

$U = \{u_1, u_2, \dots, u_n\}$為使用者集合，$I = \{i_1, i_2, \dots, i_m\}$為項目集合，$r_{u,i}$是使用者 u 對項目 i 的評分，常見的使用者 u 對項目 i 預測未知評分的三種方法[11]：

$$predict(u,i) = \frac{1}{N}\sum_{u' \in U} r_{u',i} \quad (2.1)$$

$$predict(u,i) = k\sum_{u' \in U} sim(u,u') \times r_{u',i} \quad (2.2)$$

$$predict(u,i) = \overline{r_u} + k\sum_{u' \in U} sim(u,u') \times (r_{u',i} - \overline{r_{u'}}) \quad (2.3)$$

k 是正規化因子，通常$k = 1/\sum_{u' \in U}|sim(u,u')|$。

協同過濾法中去計算 u和u'兩個使用者相似度，$sim(u,u')$常用的相似度測量法有 **Distance-based**：

$$sim(u,u') = \frac{1}{1+\sqrt{\sum_{i \in I_{uu'}}(r_{u,i} - r_{u',i})^2}} \quad (2.4)$$

、**Correlation-based**：

$$sim(u,u') = \frac{\sum_{i \in I_{uu'}}(r_{u,i} - \overline{r_u})(r_{u',i} - \overline{r_{u'}})}{\sqrt{\sum_{i \in I_{uu'}}(r_{u,i} - \overline{r_u})^2 \sum_{i \in I_{uu'}}(r_{u',i} - \overline{r_{u'}})^2}} \quad (2.5)$$

、**Cosine-based**：

$$sim(u,u') = \frac{\sum_{i \in I_{uu'}} r_{u,i} r_{u',i}}{\sqrt{\sum_{i \in I_{uu'}}(r_{u,i})^2}\sqrt{\sum_{i \in I_{uu'}}(r_{u',i})^2}} \quad (2.6)$$

其中$I_{u,u'}$是經由使用者u和u'共同評分過的項目集合。

總結來說，協同過濾法有以下優點：具有推薦新資訊的能力，可以發現使用者潛在偏好；能夠推薦音樂、餐廳和電影等難以進行內容分析的項目。

雖然協同過濾法應用廣泛，但是也有許多問題，例如：如何對新使用者進行推薦或如何推薦新項目給使用者之冷啟動問題（Cold Start）；在任何推薦系統中，已評分項目通常比需要推薦項目數量少很多，而導致精確度低落之稀疏性問題（Sparsity），面對日益增多的使用者，資料量急劇增加之可擴展性問題（Scalability）等。因此有的推薦系統採用基於項目相似度的協同過濾法[12]，在項目數量相對穩定的系統，這種方法是很有效的。

2.3.2 基於內容法（Content-based Approaches）

最初基於內容法是協同過濾法的延續與發展，他不需要依賴使用者對項目的評價，而是依據使用者已經選擇的項目內容計算使用者之間的相似度，藉此推薦合適的項目；基於內容的推薦系統依賴資訊檢索與過濾，經由分析已經購買或瀏覽的內容，對使用者和項目建立配置文件（Profile），系統可以比較使用者和項目配置文件的相似度，直接向使用者推薦最相似的項目。

總結來說，基於內容法有以下優點：可以處理新使用者與新項目的冷啟動問題；不會受到評分稀疏性的限制；能夠推薦新項目和非流行項目，能夠發現隱藏的項目；可以透過配置文件列出推薦項目的特徵，深入了解為什麼推薦這些項目。

基於內容法也面臨常見的資訊檢索問題，例如：多媒體資料（圖片、音樂與影片）難以進行文本結構化，而受到了很大的限制。

2.3.3 混合法（Hybrid Approaches）

每一種演算法都存在著一些缺點，為了擇優汰劣，有些推薦系統結合了不同的方法，我們稱之為混合法。而目前最常見的混合推薦系統即是基於內容與協同過濾法的搭配，主要又可分為獨立系統相互結合、協同過濾系統加入基於內容法。

獨立系統相互結合是分別利用兩種以上的演算法產生推薦結果，然後將這些結果整合起來，利用預測評分的線性組合進行推薦[13, 14]。

而在協同過濾系統中加入基於內容法的 Fab[15]，即是使用者配置文件使用協同過濾法產生，使用者相似度則使用基於內容法，而非參考共同評分項目，這樣可以讓系統不僅能推薦擁有共同評分的使用者項目，也可以推薦與使用者配置文件相似的項目。

2.3.4 結合情境之協同過濾法

傳統協同過濾法可以被視為二維問題，其中一維是使用者，另外一維是項目，每個使用者與項目的交互是使用者對項目的評分，當我們加入情境到推薦問題上，它就變成了多維度問題。

傳統推薦演算法無法直接處理這類問題，然而許多研究人員試圖修改原有方法來支援情境感知；Adomavicius[16]提出了一個相當著名的方法，其主要精神是在推薦時固定維度，也就是只考慮該情境下所有的評分，如此一來多維問題就變成了二維問題；然而這種方法會使得冷啟動問題變得更加棘手，因此 Adomavicius 等人還提出了一個擴展有等級屬性之情境的方法，為了增加準確度，他們還設計一種演算法將情境分段。

另外一個方法是測量不同情境的相似度去加權評分；由於使用者偏好項目會隨著情境改變，若在某些情境下使用者一致偏好此項目，Chen[17]推論這個些情境即非常適合該項目。因此他使用 Correlation-base 相似度測量法去計算項目 i 在兩個情境 x 和 y 下的相似度權重：

$$sim_t(x,y,i)=\frac{\sum_{u\in U}(r_{u,i,x}-\overline{r_i})(r_{u,i,y}-\overline{r_i})}{\sqrt{\sum_{u\in U}(r_{u,i,x}-\overline{r_i})^2\sum_{u\in U}(r_{u,i,y}-\overline{r_i})^2}} \quad (2.7)$$

其中 U 是在情境 x 和 y 對項目 i 評分的使用者集合。

Chen 還假設每個維度都是線性獨立的，如此一來兩個情境的相似度可以在每個維度中分別計算，例如：$r_{u,i,x}$ 是使用者 u 在情境 x 對項目 i 原始的評分，$R_{u,i,c}$ 則是被當前情境 c 加權過的評分：

$$R_{u,i,c}=k\sum_{x\in C}\sum_{t=1}^{z}r_{u,i,x}\times sim_t(c,x,i) \quad (2.8)$$

其中 k 是正規化因子。內迴圈涵蓋情境中的每個維度，外迴圈涵蓋該維度中的所有值，然後利用該項目在兩個情境間的相似度去加權評分。

2.4 行動餐廳推薦系統

隨著行動裝置與行動網路日益普及，人們可以隨時隨地存取的資訊激增，行動推薦系統已經成為重要的研究議題。Nguyen and Ricci[18]為行動推薦系統提出了Critique-based 方法去發現使用者長期（Long-term）和特定期間（Session-specific）的偏好，讓使用者更容易得到理想的建議。

由於使用者偏好的推薦內容會隨著情境不同而有所差異，我們應該考慮這些情境提供更好的服務；Sadeh 等人[19]的 MyCampus 提供許多情境感知的行動服務，它可以根據使用者位置、偏好、天氣和時間給出不同的建議。Huang[20]則設計了一個在行動裝置上的餐廳分享與推薦系統，並藉由位置、時間、天氣、熱指數、指外線指數、同伴類型與人數增進系統性能。

然而，團體推薦服務是另外一項重要的主題，Park 等人[21]使用貝氏網路（Bayesian Network）塑造每位使用者的偏好並應用層級分析法（Analytical Hierarchy Process）做多準則決策。

3 使用社交圖與情境感知之行動餐廳推薦系統

一般推薦服務主要針對個別使用者，但是餐廳或電影推薦應該同時考慮幾個人的偏好，而使用者偏好會跟隨著情境改變，團體偏好也會隨著同伴類型改變，因此如何得到並利用情境因素提升推薦品質，是許多情境感知之推薦系統努力的目標。

本研究目的是利用 Facebook 開放圖（Open Graph）的打卡資料（Check-ins）為行動裝置設計一個餐廳推薦系統，實現團體推薦服務，並考慮社交圖與情境資訊提升推薦品質。

理想的情境感知之推薦系統會將所有可能的情境因素加以考慮，然而這些情境因素通常無法輕易取得，例如：Huang[20]使用天氣、熱指數、指外線指數增進系統性能，因此他必須解析中央氣象局的網頁；我們則期望透過 Graph API 現有的大量資料，取得社交圖與開放圖中所有可能的情境因素，由於 Graph API 提供的欄位繁多，本論文無法一一詳列，但我們將挑出所有可能且可用的因素進一步實驗。總結我們將測試的情境因素與範圍：

- Location (l)
- Distance (d)
- Age (a): $\{13\sim110^{+}\}$
- Sex Index (s): $\{0\sim1\}$
- Time of Day (td): $\{00{:}\,00\sim23{:}\,59\}$
- Weekday (w): $\{Sun, Mon, Tue, Wed, Thu, Fri, Sat\}$
- Month (m): $\{Jan, Reb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Now, Dec\}$

- Number of Companion (*nc*): $\{1\sim50^{+}\}$
- Type of Companion (*tc*):

  $\{Alone, Friend, Classmate, Colleague, Family, Lover\}$
- Companions (*CS*)

我將修改結合情境之協同過濾法實作本系統；推薦過程中系統需要考慮社交圖與情境資訊，因此系統需要取得每次打卡的社交圖與情境資訊，以及要求推薦的個別或團體使用者當前的社交圖與情境資訊。

3.1 資料定義

Definition 1 Contextual Information and Social Graph

這個系統的請求情境 r 是一個元組$t = \langle l, d, a, s, td, w, m, nc, tc\rangle$和一個集合 $CS = \{cs_1, \dots, cs_n\}$，其中 n 是 Companions 的數量，$CS \in U$。$context: c \rightarrow t$ 函數的域是所有打卡和情境元組 t 集合；範圍是這個集合所有可能的情境元組 t。

Definition 2 Data Set

這個系統有 3 個資料集，分別為使用者、餐廳和打卡。$U = \{u_1, \dots, u_n\}$是使用者集合，其中 n 是使用者人數。$P = \{p_1, \dots, p_m\}$是餐廳集合，其中 m 是餐廳個數。 $C = \{c_{u1,p1,c1}, \dots, c_{ui,pj,ck}\}$是打卡集合，其中 $0 < i <= n$ 和 $0 < j <= m$。 $rating(c)$ 函數的域是所有打卡和偏好評分集合；範圍是 1 到 3。偏好評分值 1 代表去過一次，偏好評分值 2 代表回訪過一次，偏好評分值 3 則代表回訪過一次以上；偏好評分值反應了使用者偏好的強度。

3.2 問題定義

Definition 3 Context-aware and Social Graph based Prediction

給出餐廳集合 P 和使用者 u 的請求情境 r，這個系統應該能夠預測使用者對每家餐廳 $p \in P$ 的偏好評分。偏好評分應該反應使用者品味、社交圖與情境的影響。

Definition 4 Recommendation Process

給出餐廳集合 P、使用者 u 的請求情境 r 和使用者對每家餐廳 $p \in P$ 的預測偏好評分，這個系統應該可以產生一個依據預測偏好評分排序與尺寸 n 的餐廳清單，其中 n 是一個小整數使清單適合顯示在行動裝置的小螢幕上並傳送給使用者。

3.3 解決方案

為了實現目標，我提出了使用社交圖與情境感知之行動餐廳推薦系統，在本章節中，我將詳細介紹系統架構、資料模型與通訊過程，並解釋如何應用社交圖與情境資訊到推薦過程中。

3.3.1 系統架構與資料模型

這個系統使用 Graph API 發現使用者偏好的餐廳、情境資訊與社交圖，系統架構與資料模型，如圖 3.1：

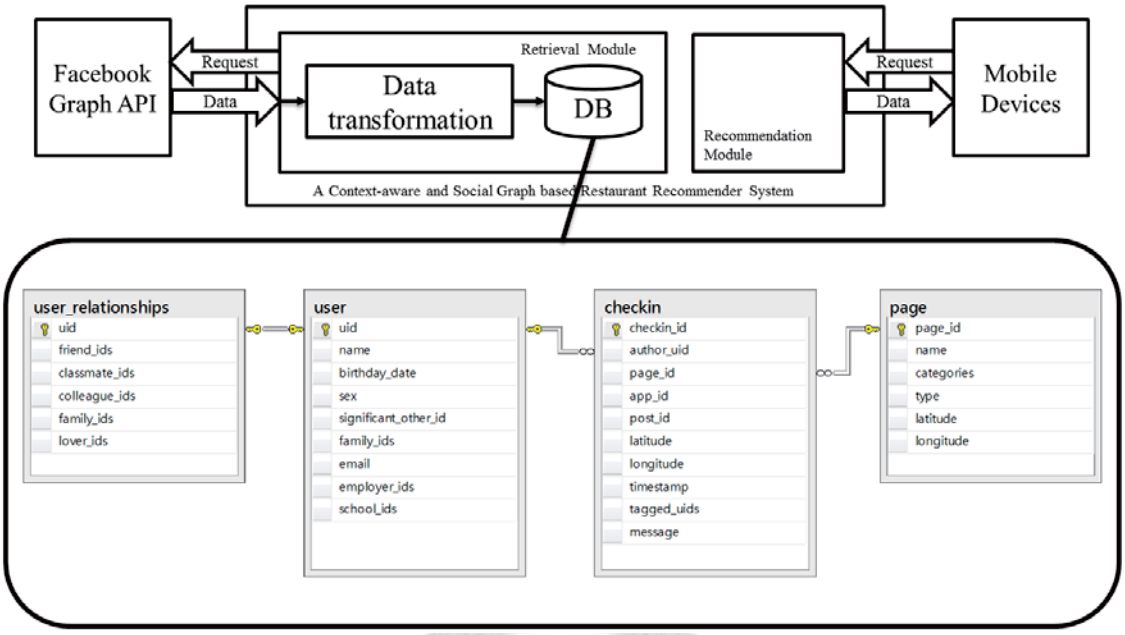


圖 3.1 系統架構與資料模型

使用者從行動裝置上使用 Facebook 帳號登入後，系統將向 Graph API 請求社交圖與關於打卡的開放圖，所有資料都將預載到資料庫中，系統包含三個部分：

行動裝置：允許使用者隨時隨地透過行動介面瀏覽附近餐廳或向推薦系統請求建議，而系統將有能力透過全球衛星定位系統（Global Positioning System）或 IP 位址（Internet Protocol Address）等定位技術，自動發現使用者當前的位置。

推薦系統：在這個系統中扮演了關鍵角色，負責檢索和儲存社交圖與開放圖，並依據個別或團體使用者的請求情境，產生賓主盡歡的餐廳推薦結果。

Graph API：Facebook 透過表徵狀態轉移(Representational State Transfer)為基礎的通訊，提供開發者搜尋和操作資料，例如：所有公開的張貼、人們、頁面、活動、團體、地方與打卡等。

行動裝置、推薦系統與 Graph API 的通訊過程，如圖 3.2：

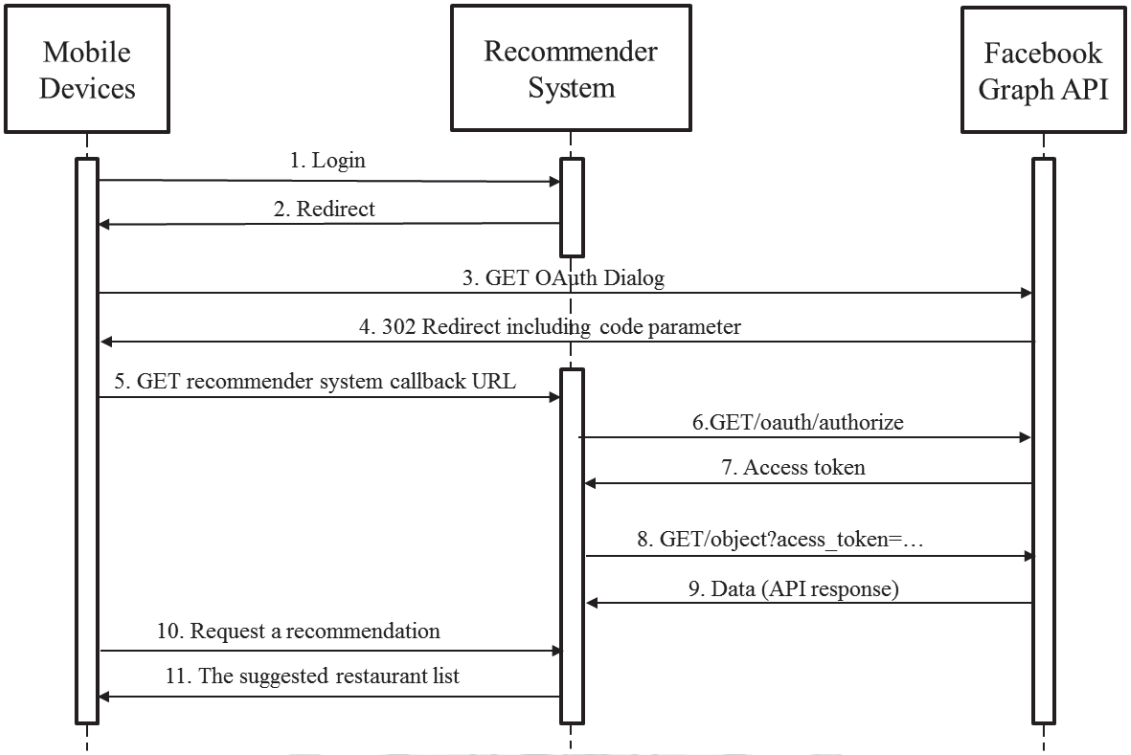


圖 3.2 系統通訊過程

3.3.2 社交圖與情境感知推薦

傳統推薦問題中，使用者需要對項目評分，但本系統使用餐廳打卡次數近似偏好評分（通常偏好的餐廳回訪次數較多），並可藉此得知當時的位置、時間、日期、人數與同伴；有了同伴，我們便能透過社交圖總結年齡、性別指標與同伴類型，組成完整的情境元組。

應用協同過濾法在這個問題上，我們必須在預測過程中總結每位使用者對各家餐廳的偏好評分，最簡單的方法是計算每位使用者對各家餐廳的平均偏好評分，但這將喪失社交圖與情境資訊的影響。

為了體現不同社交圖與情境資訊的差異，我們可以測量請求情境與每次打卡之情境的相關性，然後利用相關性去加權每位使用者對各家餐廳的偏好評分，因此使用者 u 在情境元組 t 對餐廳 p 的總結偏好評分：

$$sum(u,p,t) = \frac{\sum_{c \in C_{u,p}} rel(t,context(c)) \times rating(c)}{\sum_{c \in C_{u,p}} rel(t,context(c))} \times max_{c \in C_{u,p}} rel(t,context(c)) \quad (3.1)$$

其中$C_{u,p}$是使用者 u 對餐廳 p 的打卡集合；方程式中，較不相關情境會讓偏好評分的貢獻變少，相反的相關的情境會讓貢獻變多；為了避免不相關的情境得到高分，我們在最後乘上了情境因素中最高的相關性權重，讓較不相關的情境之總結偏好評分下降到合理的數值。

然後我們改寫相似度測量法之方程式 2.4、2.5、2.6，如下：

$$sim(u,u',t) = \frac{1}{1+\sqrt{\sum_{p \in P_{uu'}} (sum(u,p,t) - sum(u',p,t))^2}} \quad (3.2)$$

$$sim(u,u',t) = \frac{\sum_{p \in P_{uu'}} sum(u,p,t) \times sum(u',p,t)}{\sqrt{\sum_{p \in P_{uu'}} sum(u,p,t)^2 \times \sum_{p \in P_{uu'}} sum(u',p,t)^2}} \quad (3.3)$$

$$sim(u,u',t) = \frac{\sum_{p \in P_{uu'}} sum(u,p,t) \times sum(u',p,t)}{\sqrt{\sum_{p \in P_{uu'}} sum(u,p,t)^2} \times \sqrt{\sum_{p \in P_{uu'}} sum(u',p,t)^2}} \quad (3.4)$$

其中$P_{u,u'}$是被使用者 u 和u'共同打卡過的餐廳集合；我們可以看到在不同情境元組中，兩個使用者的相似度可能會有所不同。

最後我們改寫預測未知評分之方程式 2.2

$$predict(CS,p,t) = k \sum_{u \in CS} \sum_{u' \in U} sim(u,u',t) \times sum(u',p,t) \quad (3.5)$$

其中 CS 是請求情境之同伴集合，$k = 1/\sum_{u \in CS} \sum_{u' \in U} |sim(u,u',t)|$；我們可以看到在不同的請求情境之元組與同伴集合中，每家餐廳之預測偏好評分可能會有所不同。

回顧本系統對請求情境 r 的定義，請求情境 r 包含一個元組 t 與一個集合 CS，其中 c 包含 9 個成員，分別為 Location(l)、Distance(d)、Age(a)、Sex Index(s)、Time of Day(td)、Weekday(w)、Month(m)、Number of Companion(nc) 和 Type of Companion

(*tc*)，而 *CS* 是同伴集合。本論文剩下的部分，我將用 *r.t* 代表請求情境 *r* 之元組 *t*，*r.t.l* 代表請求情境 *r* 之位置， *r.CS* 代表請求情境 *r* 之同伴集合等等，依此類推。

當系統接收到使用者請求情境後，會先利用位置與距離參數過濾可能的候選餐廳，然後總結當下的社交圖與情境資訊；對於每家候選餐廳，系統使用方程式 3.5 去預測偏好評分並排序，然後將前 *n* 家候選餐廳推薦給使用者，演算法 1 和演算法 2 描述完整的過程：

Algorithm 1 recommendation

Input: *P*, the set of all restaurant
u, the user
r, the user's request
Output: *M*, the suggested restaurant list
$M \Leftarrow \{\}$
for all $p \in P$ **do**
  if $distance(p, r.t.l) \leq r.t.d$ **then**
    $M \Leftarrow M \cup \{p\}$
  end if
end for
$r.CS \Leftarrow r.CS \cup u$
$r \Leftarrow summarize\_context(r)$
for all $p \in M$ **do**
  $rating \Leftarrow predict(r.CS, p, r.t)$
  $assign\_rating(p, rating)$
end for
$sort\_by\_rating(M)$
$M \Leftarrow M[1 : n]$
return *M*

Algorithm 2 summarize_context

Input: *r*, the user's request
Output: *r*, the summary context

```
r.t.a ⇐ avg_age(r.CS)
r.t.s ⇐ sex_index(r.CS)
n ⇐ count(r.CS)
if n = 1 then
    r.t.tc ⇐ Alone
else
    R ⇐ {Friend : 0, Classmate : 0, Colleague : 0, Family : 0, Lover : 0}
    for all cs1 ∈ r.CS do
        for all cs2 ∈ r.CS do
            if (cs1 != cs2) then
                rel ⇐ relationship(cs1, cs2)
                assign_point(R, rel)
            end if
        end for
    end for
    r.t.tc ⇐ relationship_of_max(R)
end if
return r
```

3.3.3 社交圖與情境資訊比較

最後我將說明如何測量社交圖與情境資訊之間的相關性；為了從偏好評分中觀察不同社交圖與情境資訊之間的相似性，我將使用啟發式方法（Heuristic Approach）去測量不同社交圖與情境資訊之間的相關性，從中所得到的相關性權重是一個介於 0 到 1 的實數，我們將用於協同過濾法中加權偏好評分。

Location(l)與 Distance(d)將用於篩選候選餐廳，因為人們會根據需求考慮適當距離內的餐廳，所以不需要進行相關性測量，剩下的請求情境之元組 t 和元組t'的相關性測量模型如下：

Age (*a*)：我假設平均年齡的相關性是指數函數，差異越大，相關性越低，係數的實際值將從收集到的資料中取得。

$$rel_a(t,t') = \begin{cases} 1, & if\ t.a = t'.a \\ Coef_a^{\,abs(t.a-t'.a)}, & otherwise \end{cases} \quad (3.6)$$

Sex Index (*s*)：我假設性別指標的相關性是指數函數，差異越大，相關性越低，係數的實際值將從收集到的資料中取得。

$$rel_s(t,t') = \begin{cases} 1, & if\ t.s = t'.s \\ Coef_s^{\,abs(t.s-t'.s)}, & otherwise \end{cases} \quad (3.7)$$

Time of Day (*td*)：我假設時間的相關性是指數函數，差異越大，相關性越低，係數的實際值將從收集到的資料中取得。

$$rel_t(t,t') = \begin{cases} 1, & if\ t.td = t'.td \\ Coef_{td}^{\,time\_difference(t.td,t'.td)}, & otherwise \end{cases} \quad (3.8)$$

Weekday (*w*)：一般來說，星期可以分為平日或週末，所以我用一個係數調整不同情況下的相關性模型，係數的實際值將從收集到的資料中取得。

$$rel_t(t,t') = \begin{cases} 1, & if\ isWeekend(t.w) = isWeekend(t'.w) \\ Coef_w, & otherwise \end{cases} \quad (3.9)$$

Month (*m*)：我假設不同月份的相關性是指數函數，差異越大，相關性越低，係數的實際值將從收集到的資料中取得。

$$rel_m(t,t') = \begin{cases} 1, & if\ t.m = t'.m \\ Coef_m^{\,month\_difference(t.m,t'.m)}, & otherwise \end{cases} \quad (3.10)$$

Number of Companion (*nc*)：我假設同伴人數的相關性是指數函數，差異越大，相關性越低，係數的實際值將從收集到的資料中取得。

$$rel_{nc}(t,t') = \begin{cases} 1, & if\ t.nc = t'.nc \\ Coef_{nc}^{\ abs(t.nc-t'.nc)}, & otherwise \end{cases} \quad (3.11)$$

Type of Companion (*tc*)：人們可能因不同的同伴類型而選擇不同的餐館，所以我用幾個係數調整各種同伴類型之間的相關性模型，如表 3.1。係數的實際值將從收集到的資料中取得。

$$rel_{tc}(t,t') = TC\_RelevanceMatrix[t.tc][t'.tc] \quad (3.12)$$

	Alone(α)	**Friend(β)**	**Classmate(γ)**	**Colleague(δ)**	**Family(ε)**	**Lover(ζ)**
Alone	1	$Coef_{\alpha\beta}$	$Coef_{\alpha\gamma}$	$Coef_{\alpha\delta}$	$Coef_{\alpha\varepsilon}$	$Coef_{\alpha\zeta}$
Friend	$Coef_{\alpha\beta}$	1	$Coef_{\beta\gamma}$	$Coef_{\beta\delta}$	$Coef_{\beta\varepsilon}$	$Coef_{\beta\zeta}$
Classmate	$Coef_{\alpha\gamma}$	$Coef_{\beta\gamma}$	1	$Coef_{\gamma\delta}$	$Coef_{\gamma\varepsilon}$	$Coef_{\gamma\zeta}$
Colleague	$Coef_{\alpha\delta}$	$Coef_{\beta\delta}$	$Coef_{\gamma\delta}$	1	$Coef_{\delta\varepsilon}$	$Coef_{\delta\zeta}$
Family	$Coef_{\alpha\varepsilon}$	$Coef_{\beta\varepsilon}$	$Coef_{\gamma\varepsilon}$	$Coef_{\delta\varepsilon}$	1	$Coef_{\varepsilon\zeta}$
Lover	$Coef_{\alpha\zeta}$	$Coef_{\beta\zeta}$	$Coef_{\gamma\zeta}$	$Coef_{\delta\zeta}$	$Coef_{\varepsilon\zeta}$	1

表 3.1 *TC_RelevanceMatrix*

兩個請求情境的元組 t 和元組t'之整體相關性是上述方程式的乘積：

$$rel(t,t') = \prod_{f\in F} rel_f(t,t') \quad (3.13)$$

$$where\ F = \{a, s, td, w, m, nc, tc\}$$

為了使時間 23 時對 0 時和月份 12 月對 1 月等案例之差距為一個單位，我們需要正確的計算方法，演算法 3 和演算法 4 描述了完整的過程：

Algorithm 3 time_difference

Input: *t1*, time of the check-in
t2, time of the user's request
Output: *td*, time difference
t1 ⇐ *abs(t1 - t2)*
if *(t1* > 12*)* **then**
    td ⇐ 24 - *t1*
end if
return *td*

Algorithm 4 month_difference

Input: *m1*, month of the check-in
m2, month of the user's request
Output: *md*, month difference
m1 ⇐ *abs(m1 - m2)*
if *(m1* > 6*)* **then**
    md ⇐ 12 - *m1*
end if
return *md*

4 實作

在章節 3.3.1 中，我提到這個系統包含了三個部份：行動裝置、推薦系統與 Graph API，系統架構、資料模型與通訊過程顯示在圖 3.1 和圖 3.2，而推薦系統在這之中扮演了關鍵角色，我將在本章節說明如何在雲端平台上實作這個系統，接著展示使用者介面。

4.1 雲端計算

雲端運算（Cloud Computing）指的是一種將計算與儲存容量作為服務，並按需求提供給異構（Heterogeneous）社區的終端接收者。雲端計算具有下列核心特性[22]：

- **敏捷性**改進使用者重新分配基礎設施的能力。
- **應用程式介面**的可達性使軟體與雲端交互就像人機互動介面一樣，它通常使用基於 REST 架構的 APIs。
- **成本降低**是因為在公有雲的傳輸模式中支持已經轉變為營運成本。
- **設備和位置獨立**讓使用者可以使用瀏覽器存取系統，而不需要在乎自身使用何種裝置或身處何地。
- **虛擬化**技術允許伺服器和儲存裝置共享和利用率增加，而應用程式很容易從一個實體伺服器遷移到另一個。
- **多租戶**使眾多使用者共享資源與成本，從而達到基礎設施在某處低成本**集中化管理**（例如：減少房地產、電力等），**峰值負載能力**增加（使用者不需要為可能的最高負載等級進行建造），**利用率與效率**增加（以前系統經常只有 10%到 20%使用率）。

- **可靠性**增加是因為可用雲端良好的設計建置多個備用站點，適合企業永續經營與災難復原。
- **可擴展性和彈性**經由動態（按需求）在一個良好粒度（Fine-grained）上的資源供給，接近於即時的自我服務，無須使用者對峰值進行建造。
- **性能**被監控，且使用網路服務做為系統介面建造一致性與松散耦合架構。
- **安全**被改善，由於資料的集中化，增加了安全資源等等，但對特定敏感資料失去控制，以及缺乏安全的儲存核心需要被持續關注。

雲端服務可分為三種層次，也就是軟體、平台以及基礎設施，如下：

- **軟體即服務（Software as a Service）**：供應商在雲端上安裝並操作應用軟體，而使用者從伺服端存取軟體，不須管理雲端基礎設施和應用程式運行平台，也不須安裝和運行雲端上的應用程式，只要連上網路就可以使用。例如：Google Apps 和 Yahoo 無名小站。
- **平台即服務（Platform as a Service）**：供應商提供雲端平台給使用者，其包含作業系統、程式語言執行環境、資料庫與伺服器，應用程式開發者可以在雲端平台上開發和運行他們的軟體，不須花成本在購買與管理複雜的硬體和軟體層，而有些平台及服務則會自動依照應用程式需求分配底層的計算與儲存資源，讓使用者不需要手動管理。例如：Google App Engine 和 Microsoft Windows Azure。
- **基礎設施即服務（Infrastructure as a Service）**：最基本的雲端服務模型，供應商提供實體電腦或虛擬機器、儲存體、防火牆、負載平衡器和網路。基礎設施即服務供應商依需求從大型資料中心中提供這些資源。使用者安裝作業系統映像檔（Image）以及他們的應用軟體在機器上，並負責維修、保養作業系統和應用程式。例如：Amazon EC2 和 IBM Blue Cloud。

本論文使用 ASP.NET MVC 4 Developer Preview 與 C# 開發系統，並部署到 Windows Azure Platform 與 SQL Azure（基於雲端計算、可擴展版本的 SQL 伺服器）。

4.2 系統流程

推薦系統在這個系統中扮演了關鍵角色，負責檢索並儲存社交圖與開放圖，並依據個別或團體使用者的請求情境，產生賓主盡歡的餐廳推薦結果；推薦系統可分為檢索與推薦兩個模組，系統流程如圖 4.1。

在檢索模組中，首先，個別或團體使用者手動輸入或系統自動發現當前情境，向系統請求建議，系統會呼叫 Graph API 取得本次請求所需使用者和同伴的歷史資料（使用者檔案、餐廳資訊、打卡資料與社交圖），從中解析和處理每次回應的 JSON 格式之資料，將其插入或更新到資料庫中，並另開副線程（Threading）請求所有朋友的歷史資料以豐富資料集；當資料庫取得本次請求所需資料後（副線程可能尚未結束），會先被預處理成可立即使用的資料集，並呼叫推薦模組。

在推薦模組中，會先使用位置與距離參數從資料集中過濾需考慮的餐廳，以增強系統效能，並從資料庫中總結本次請求的社交圖與情境資訊，然後利用章節 3.3.2 所提到的方程式，預測在總結的社交圖與情境資訊下，使用者與同伴對每家餐廳共同的偏好評分，最後產生一個經由預測偏好評分排序的 n 家餐廳清單，使其能夠顯示在行動裝置的小螢幕上，提供當下最佳的建議。

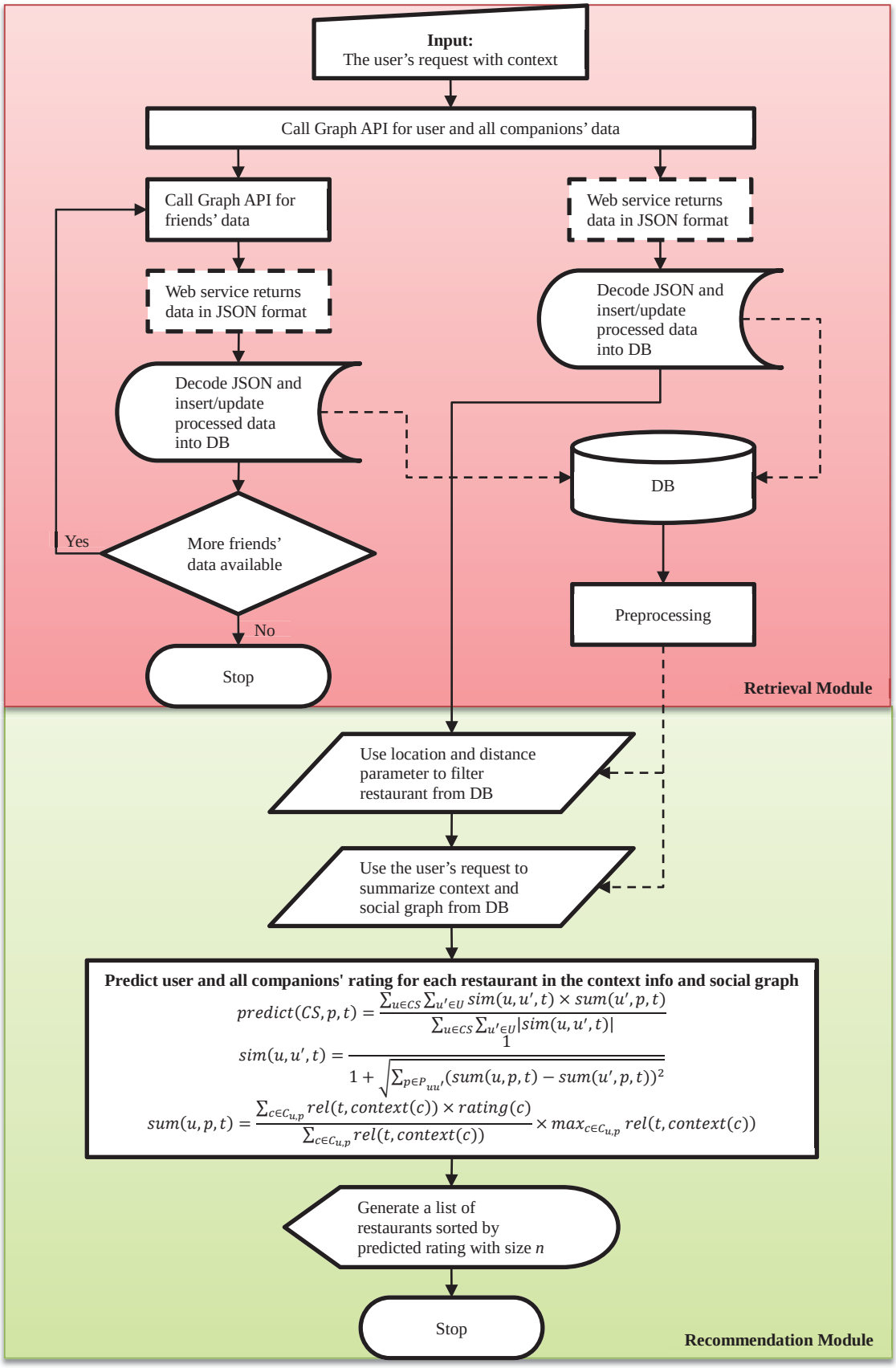

Input:
The user's request with context
Call Graph API for user and all companions' data
Call Graph API for friends' data
Web service returns data in JSON format
Web service returns data in JSON format
Decode JSON and insert/update processed data into DB
Decode JSON and insert/update processed data into DB
DB
Yes
More friends' data available
No
Stop
Preprocessing
Retrieval Module
Use location and distance parameter to filter restaurant from DB
Use the user's request to summarize context and social graph from DB
Predict user and all companions' rating for each restaurant in the context info and social graph
$predict(CS,p,t) = \frac{\sum_{u \in CS} \sum_{u' \in U} sim(u,u',t) \times sum(u',p,t)}{\sum_{u \in CS} \sum_{u' \in U} |sim(u,u',t)|}$
$sim(u,u',t) = \frac{1}{1+\sqrt{\sum_{p \in P_{uu'}} (sum(u,p,t) - sum(u',p,t))^2}}$
$sum(u,p,t) = \frac{\sum_{c \in C_{u,p}} rel(t,context(c)) \times rating(c)}{\sum_{c \in C_{u,p}} rel(t,context(c))} \times max_{c \in C_{u,p}} rel(t,context(c))$
Generate a list of restaurants sorted by predicted rating with size n
Stop
Recommendation Module

圖 4.1 系統流程圖

4.3 使用者介面

本系統之使用者介面與 Google Map 混搭，其提供 JavaScript API 讓開發者整合地圖到適地性服務中；我還使用 JavaScript 與 jQuery 作為客戶端腳本語言增進人機互動介面，並使用 AJAX 技術與 JSON 傳輸，使系統更為迅捷地回應使用者的動作。

在互動式的地圖介面中，系統會透過瀏覽器使用的定位技術自動發現使用者當前的位置並返回相關資訊，使用者可點擊附近餐廳清單中的項目或地圖上的標記查看細節，也可以拖曳地圖去尋找餐廳；或切換至推薦模式，自動設定或手動修改當前的位置與情境，向系統請求建議，如圖 4.2、圖 4.3 和圖 4.4：

圖 4.2 使用 Web 介面瀏覽地圖和尋找餐廳的截圖

圖 4.3 使用 Web 介面請求建議的截圖

圖 4.4 使用 Mobile 介面的截圖

5 實驗與評價

推薦系統最常使用交叉驗證（cross-validation）驗證推薦演算法的性能，例如：Sarwar 等人[23]使用兩組真實世界的資料集驗證幾種推薦演算法的性能，分別為電影推薦網站與大型電子商務的交易紀錄；他隨機抽取 MovieLens 網站評分 20 次以上的資料，得到 943 個使用者對 1682 部電影的 10 萬次評分，以及 Fingerhut 電子商務公司的歷史交易紀錄，包含 6502 位客戶購買 23554 種產品的 97045 次購買紀錄，並應用 5 倍交叉驗證去估計系統的精確度、召回率與 F1 指標。然而對於一個行動推薦系統而言，不能只是單純猜測使用者是否偏好這個項目，而是必須依據使用者偏好的程度排序，並取前幾個項目作為推薦清單，使它適合顯示在行動裝置的小螢幕上。

在本實驗中，我將觀察使用者在推薦清單中實際選擇餐廳的位置，評價系統性能；基本上，平均位置越上面代表性能越高。

5.1 資料收集

本實驗中的資料由 69 名 Facebook 使用者提供，收集區域集中在大台北地區（我的生活圈），收集到 2010/8/25 至 2012/4/30 的打卡資料，表 5.1、表 5.2、表 5.3、圖 5.1 和圖 5.2 描述了這個資料集的特點：

項目	數量
受測者人數	69
使用者人數	25184
曾打卡的使用者人數	7395
地點個數	37305
打卡數量	74283
曾對餐廳打卡的使用者人數	3928
餐廳個數	2691
對餐廳的打卡數量	8264

表 5.1 收集到的各項資料之數量

項目	比率	
受測者男女比(男/女)	1.16	37 / 32
使用者男女比(男/女)	1.13	13309 / 11763
曾對餐廳打卡的使用者男女比(男/女)	0.87	1825 / 2088
曾對餐廳打卡的使用者比率	15.60%	3928 / 25184
種類為餐廳的地點比率	7.21%	2691 / 37305
對餐廳的打卡比率	11.13%	8264 / 74283
公開年齡的使用者比率	60.40%	15212 / 25184
公開性別的使用者比率	99.56%	25072 / 25184
公開家鄉的使用者比率	30.90%	7782 / 25184
公開公司的使用者比率	47.10%	11861 / 25184
公開學校的使用者比率	80.12%	20178 / 25184
公開家人的使用者比率	60.83%	15319 / 25184
公開情人的使用者比率	15.90%	4003 / 25184
餐廳打卡在大台北地區的比率	67.81%	5604 / 8264
餐廳在大台北地區的比率	48.94%	1317 / 2691
曾對大台北地區餐廳打卡的使用者比率	69.27%	2721 / 3928

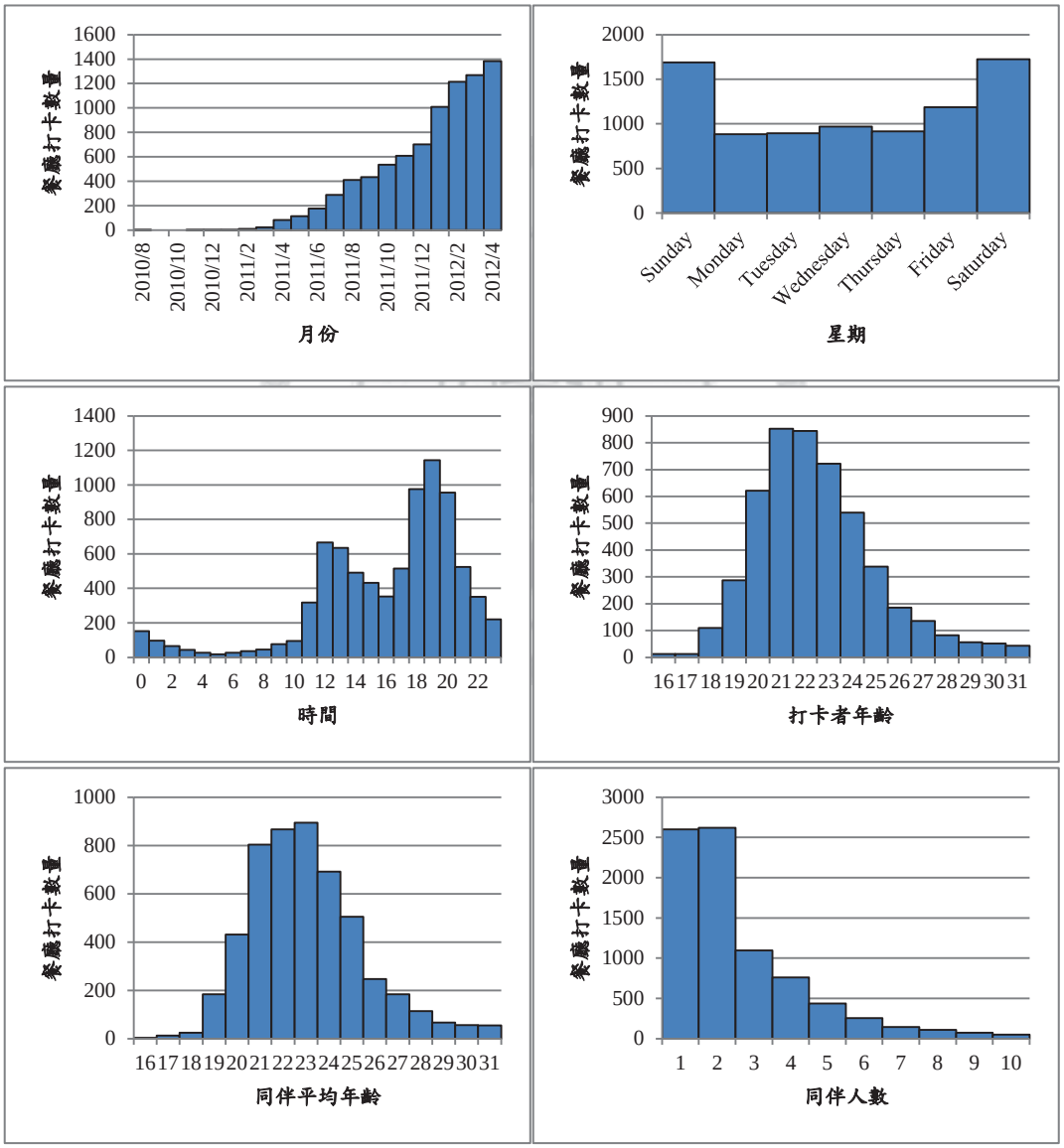
表 5.2 收集到的各項資料之比率

圖 5.1 收集到的所有餐廳之分布圖

項目	平均值	中位數	標準差	最大值	最小值	
受測者朋友數	504.01	423	346.48	2200	34	n=69
受測者年齡	23.28	22	4.99	49	18	n=69
使用者年齡	23.52	22	5.87	107	13	n=25184
曾對餐廳打卡的使用者年齡	22.99	22	4.17	106	15	n=3928
使用者對餐廳的打卡次數	0.34	0	0.99	12	0	n=25184
使用者打卡過的餐廳個數	0.33	0	0.93	10	0	n=25184
餐廳被打卡的次數	3.23	2	6.31	224	1	n=2691
曾對餐廳打卡的使用者對餐廳的打卡次數	2.21	2	1.48	12	1	n=3928
曾對餐廳打卡的使用者打卡過的餐廳個數	2.10	2	1.35	10	1	n=3928
餐廳打卡的餐廳被打卡次數	1.05	1	0.29	10	1	n=8264

餐廳打卡的打卡者年齡	22.96	22	3.97	106	15	n=8264
餐廳打卡的同伴平均年齡	23.55	23	3.91	107	16	n=8264
餐廳打卡的同伴性別指標(女/總數)	0.56	1	0.48	1	0	n=8264
餐廳打卡的同伴人數	2.76	2	2.43	42	1	n=8264

表 5.3 收集到的各項資料之平均值、中位數與標準差



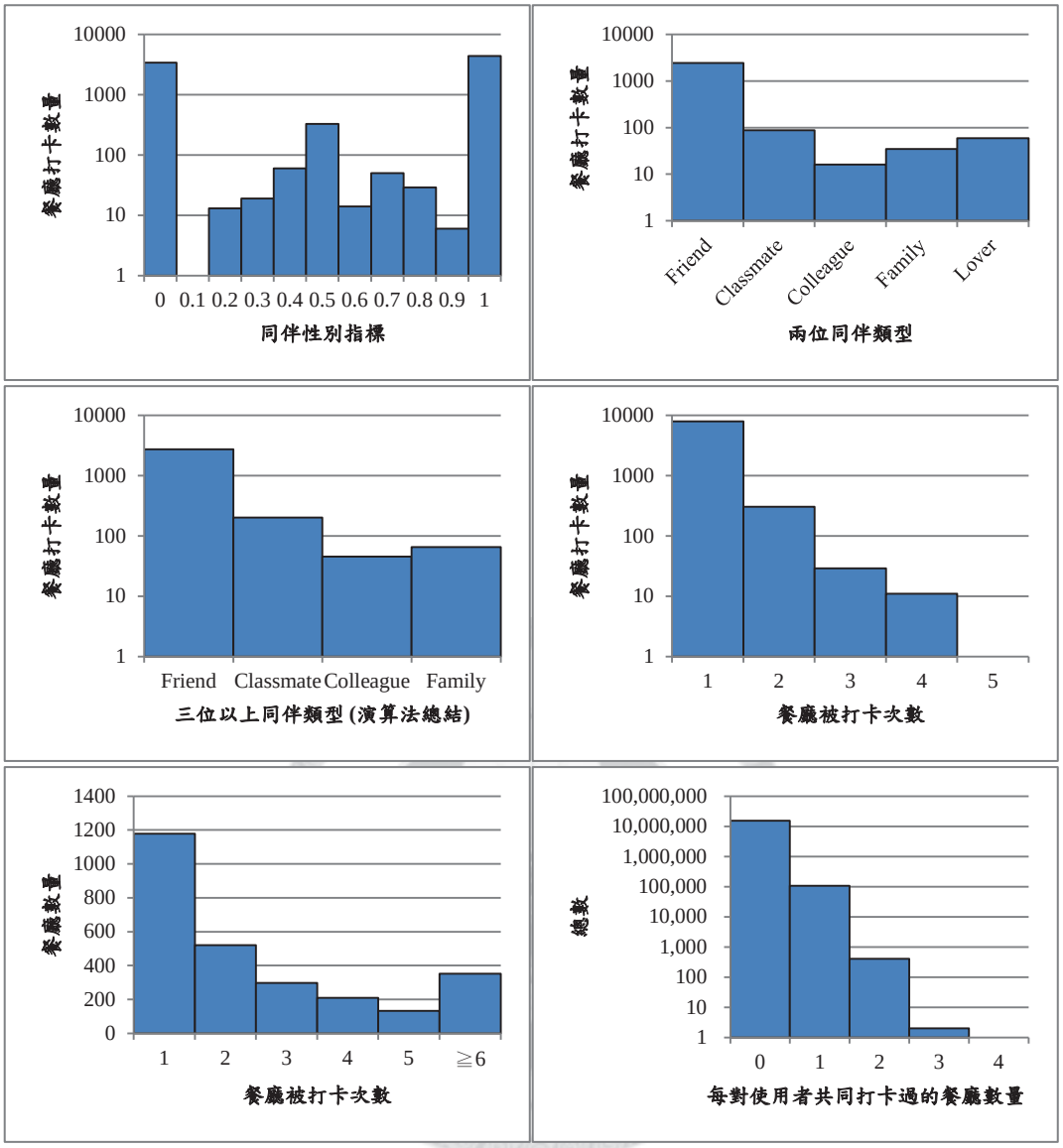


圖 5.2 統計收集到的餐廳打卡資料

此資料集有許多重要的現象，例如：

- 1位使用者約可帶來365位使用者的資料(收集到的使用者人數/受測者人數)，如表 5.1。
- 每 7 個人就有 1 個曾對餐廳打卡，每 14 個地點就有 1 個是餐廳，每 9 次打卡就有 1 次在餐廳，女性較男性熱衷於餐廳打卡上；使用者公開家鄉的比率過低（故本系統未採用此情境因素），如表 5.2。
- 大部分的打卡與餐廳集中在大台北地區，如圖 5.1。

- 受測者平均有 504 個朋友，所有人的平均年齡約 24 歲；每個人平均對餐廳打卡 0.34 次，平均打卡過 0.33 家餐廳，每家餐廳平均被打卡約 3 次，曾對餐廳打卡的人平均對餐廳打卡約 2 次，平均打卡過約 2 家餐廳；餐廳打卡的同伴平均約 3 人，平均年齡約 24 歲，女性比率稍高，如表 5.3。
- 大部分的餐廳打卡發生在近半年，周休二日明顯較多，中餐和晚餐時間呈現峰值，打卡者年齡在 21 歲呈現峰值，同伴年齡在 23 歲呈現峰值，常結伴同行，同伴性別分布均勻，朋友數量是其他類型的 10 倍以上；大多數的餐廳只有被打卡 1 次，相似的使用者只有 1 個共同打卡過的餐廳居多，如圖 5.2。

若轉換此資料集為使用者與餐廳的二元矩陣，此矩陣有 3928 行（曾對餐廳打卡的使用者人數）和 2691 列（餐廳個數），若定義稀疏程度為$\frac{nonzero\ entires}{total\ entries}$，此資料集的稀疏程度為 0.00078。

5.2 選擇相似度測量法

在本系統中，只要相似度測量法得到的結果與一些基本準則保持一致，我們就可以自由選擇能產生最好結果的相似度測量法；回顧章節 3.3.2，我們改寫了三種相似度測量法，使其成為方程式 3.2、方程式 3.3 和方程式 3.4。

在選擇相似度測量法之前，我們可以先固定用於篩選候選餐廳的距離參數，因為人們通常不會考慮太遠的餐廳，各距離參數所涵蓋的範圍如圖 5.3：

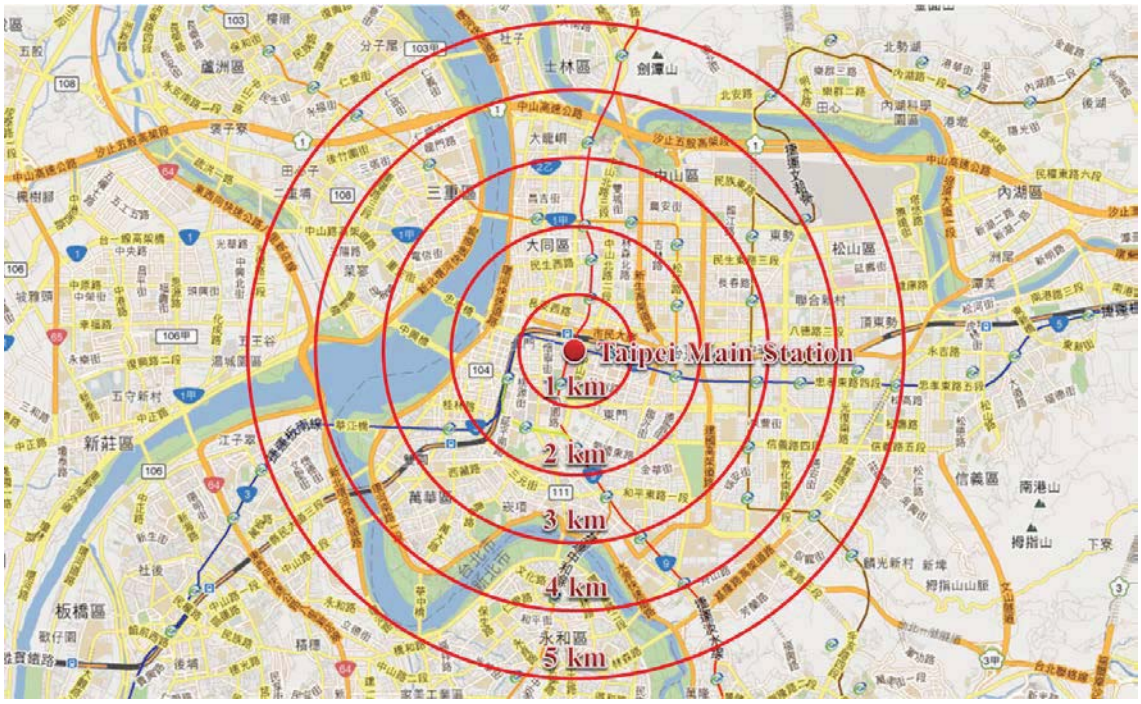


圖 5.3 各距離參數所涵蓋的範圍（以台北車站為中心）

我們可以發現當距離參數設為 5 公里後，其涵蓋的範圍已經達到了整個市區，故本實驗將以此距離參數繼續進行，其餘距離參數將留到最後討論。我們應用 LOOCV（Leave-one-out cross-validation）在此真實世界資料集中評價各種相似度測量法的性能，如圖 5.4。驗證過程中，我們會移除驗證資料之打卡對訓練資料之偏好評分的影響。

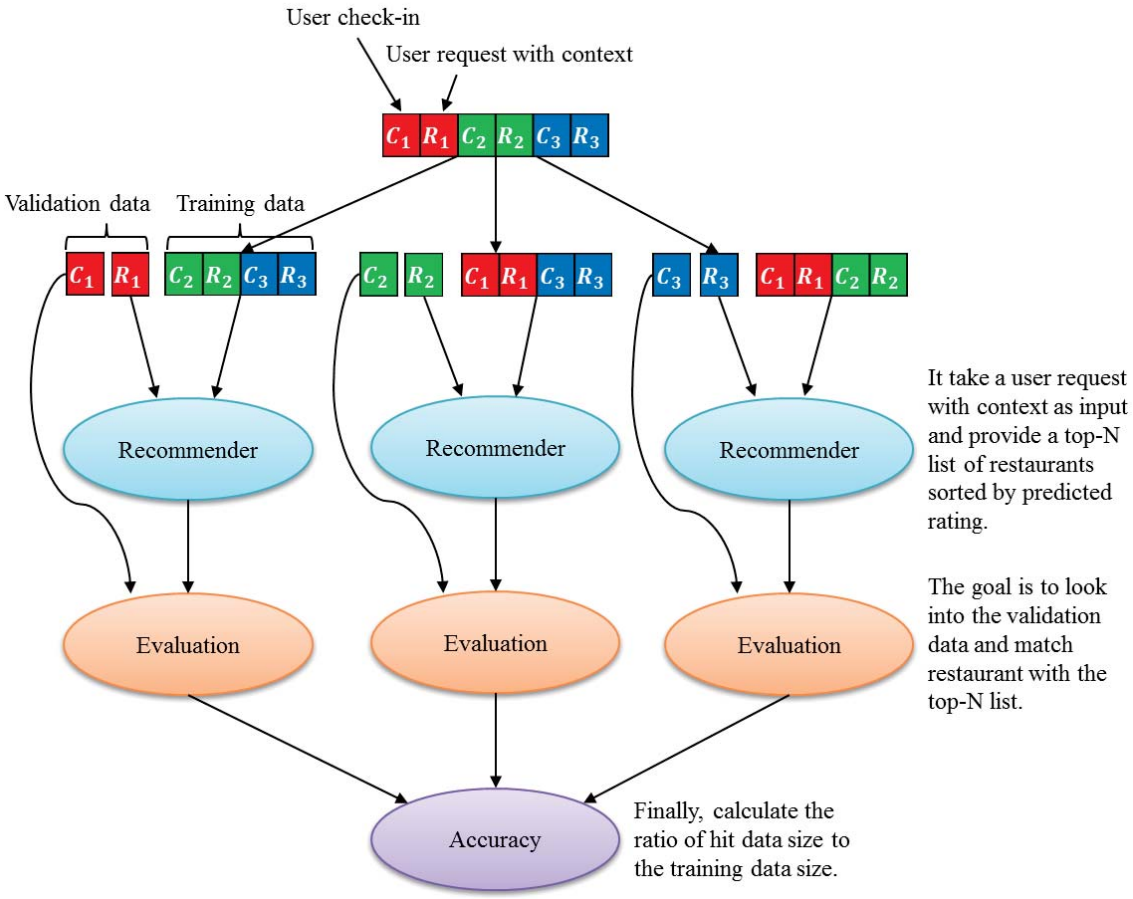
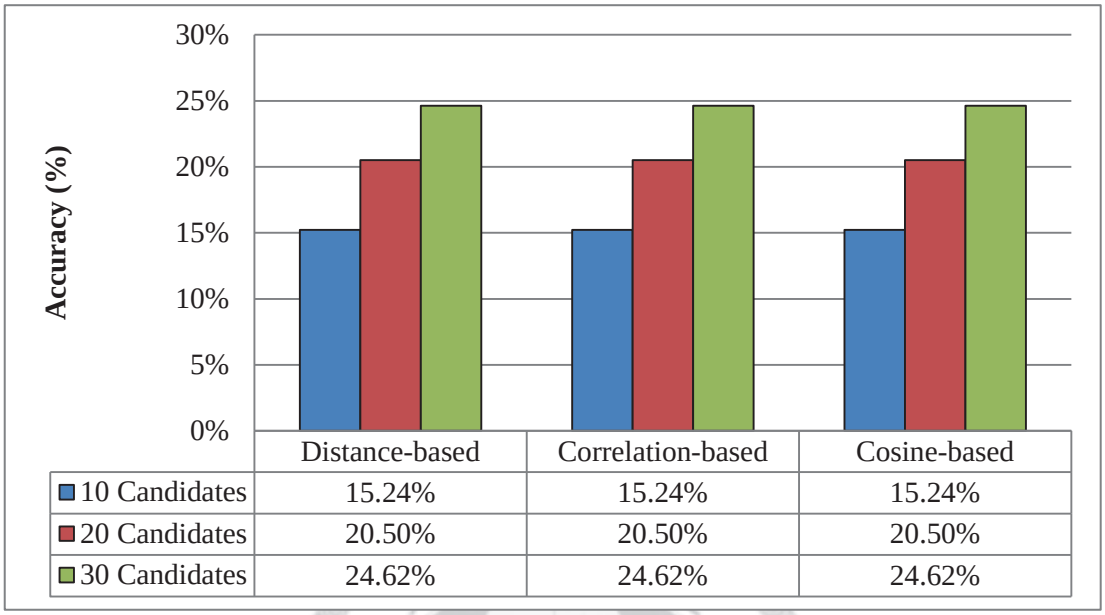


圖 5.4 本論文使用 LOOCV 的驗證過程

相似度測量法的評價結果如圖 5.5：

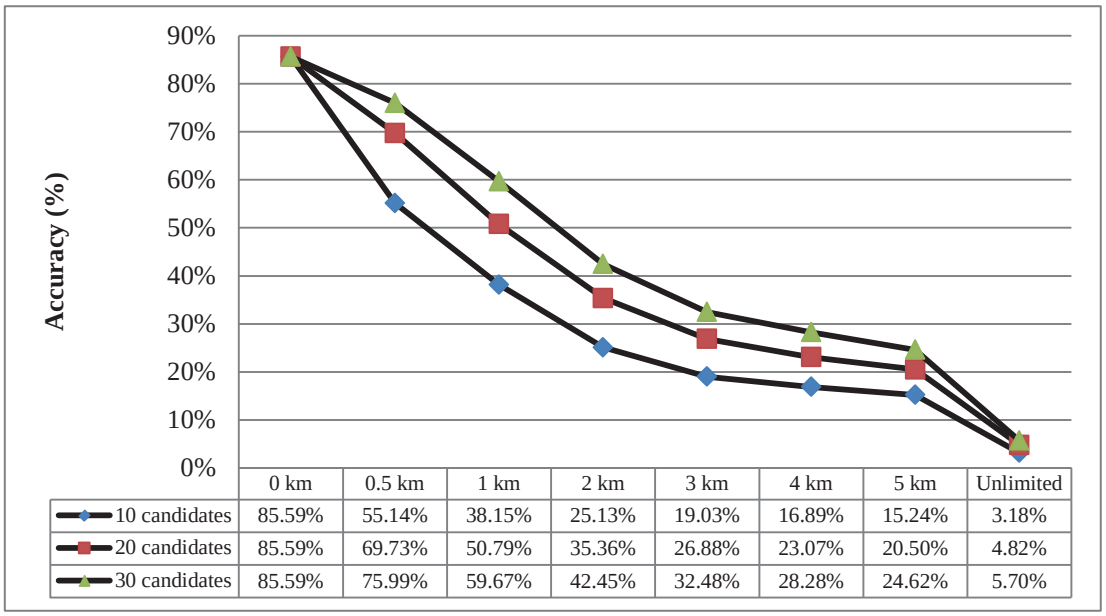


	Distance-based	Correlation-based	Cosine-based
10 Candidates	15.24%	15.24%	15.24%
20 Candidates	20.50%	20.50%	20.50%
30 Candidates	24.62%	24.62%	24.62%

圖 5.5 三種相似度測量法之推薦準確度

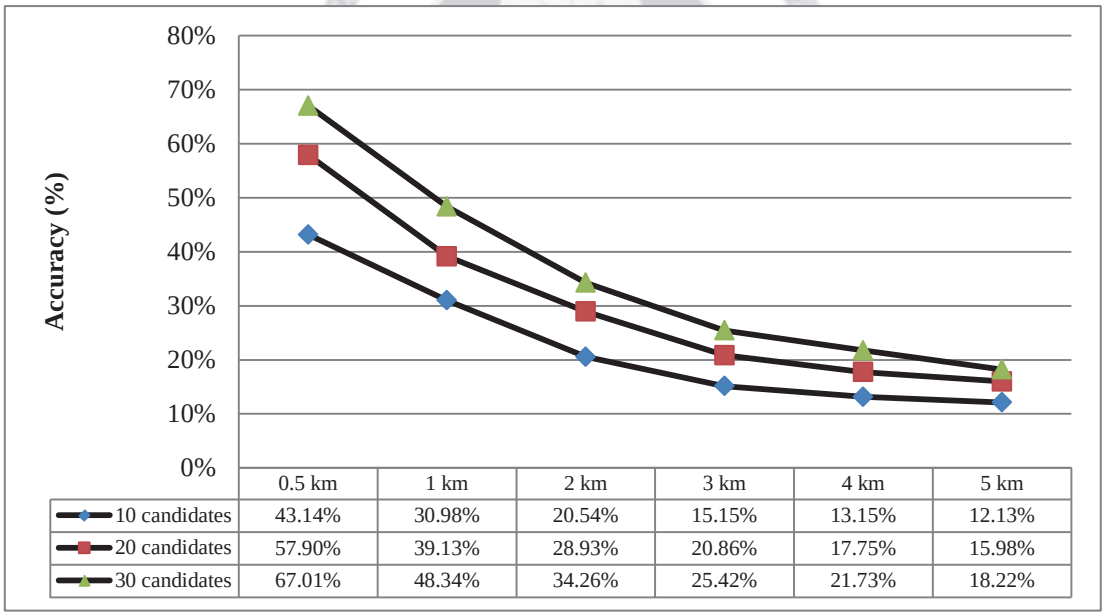
我們可以看到三種相似度測量法在僅參考位置與距離參數下的表現一致，這是因為相似的使用者只有 1 個共同打卡過的餐廳居多，如圖 5.2；故我們選擇簡單的 Distance-based，並期望它對數值敏感的特性，能夠在情境相關性的加權上，得到比較顯著的效果。

當系統接收到使用者請求情境後，會先利用位置與距離參數過濾可能的候選餐廳，我們分別將本系統的推薦數量固定為 10、20 和 30 家候選餐廳，距離參數固定為 500 公尺、1、2、3、4 和 5 公里，對原始結果和篩選推薦數量小於該距離參數之餐廳數量的情況，各別進行僅參考位置與距離參數的初步評價，如圖 5.6 和圖 5.7：



	0 km	0.5 km	1 km	2 km	3 km	4 km	5 km	Unlimited
10 candidates	85.59%	55.14%	38.15%	25.13%	19.03%	16.89%	15.24%	3.18%
20 candidates	85.59%	69.73%	50.79%	35.36%	26.88%	23.07%	20.50%	4.82%
30 candidates	85.59%	75.99%	59.67%	42.45%	32.48%	28.28%	24.62%	5.70%

圖 5.6 僅參考位置與距離參數的各距離參數之準確度



	0.5 km	1 km	2 km	3 km	4 km	5 km
10 candidates	43.14%	30.98%	20.54%	15.15%	13.15%	12.13%
20 candidates	57.90%	39.13%	28.93%	20.86%	17.75%	15.98%
30 candidates	67.01%	48.34%	34.26%	25.42%	21.73%	18.22%

圖 5.7 僅參考位置與距離參數的各距離參數之準確度(篩選推薦數量小於該距離參數之餐廳數量的情況)

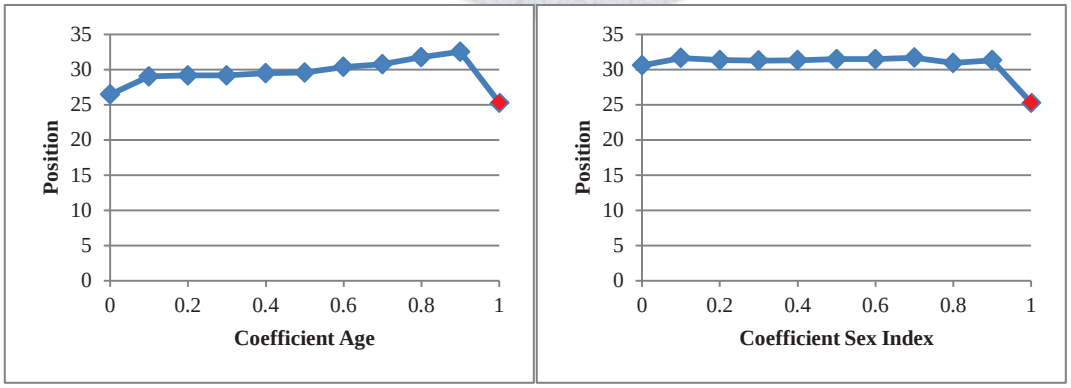
實驗結果顯示，距離參數增大時，推薦準確度遞減；這是因為使用者可選擇的餐廳數量，會隨著距離參數決定的涵蓋範圍擴大，而大幅增加。

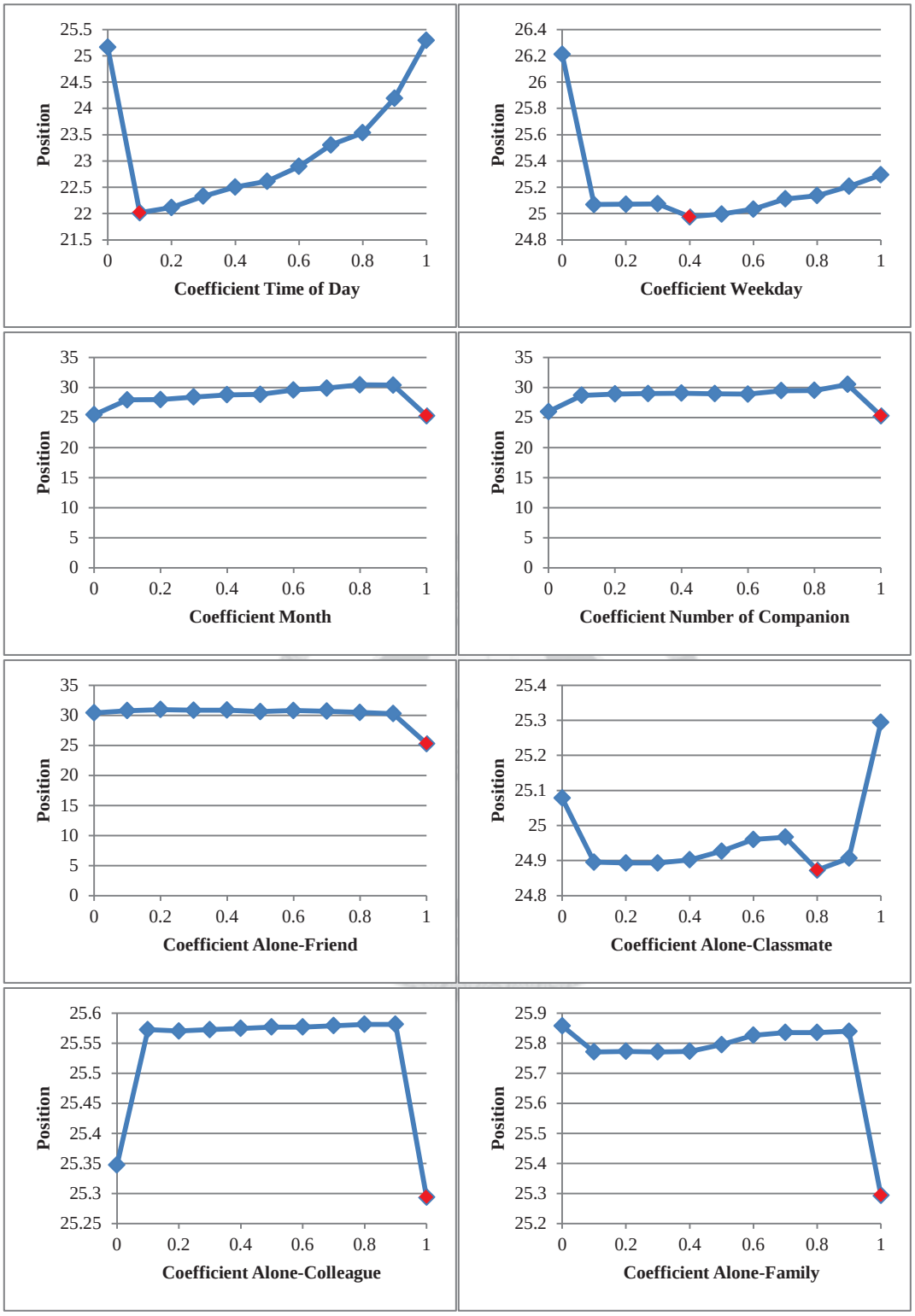
5.3 調整係數

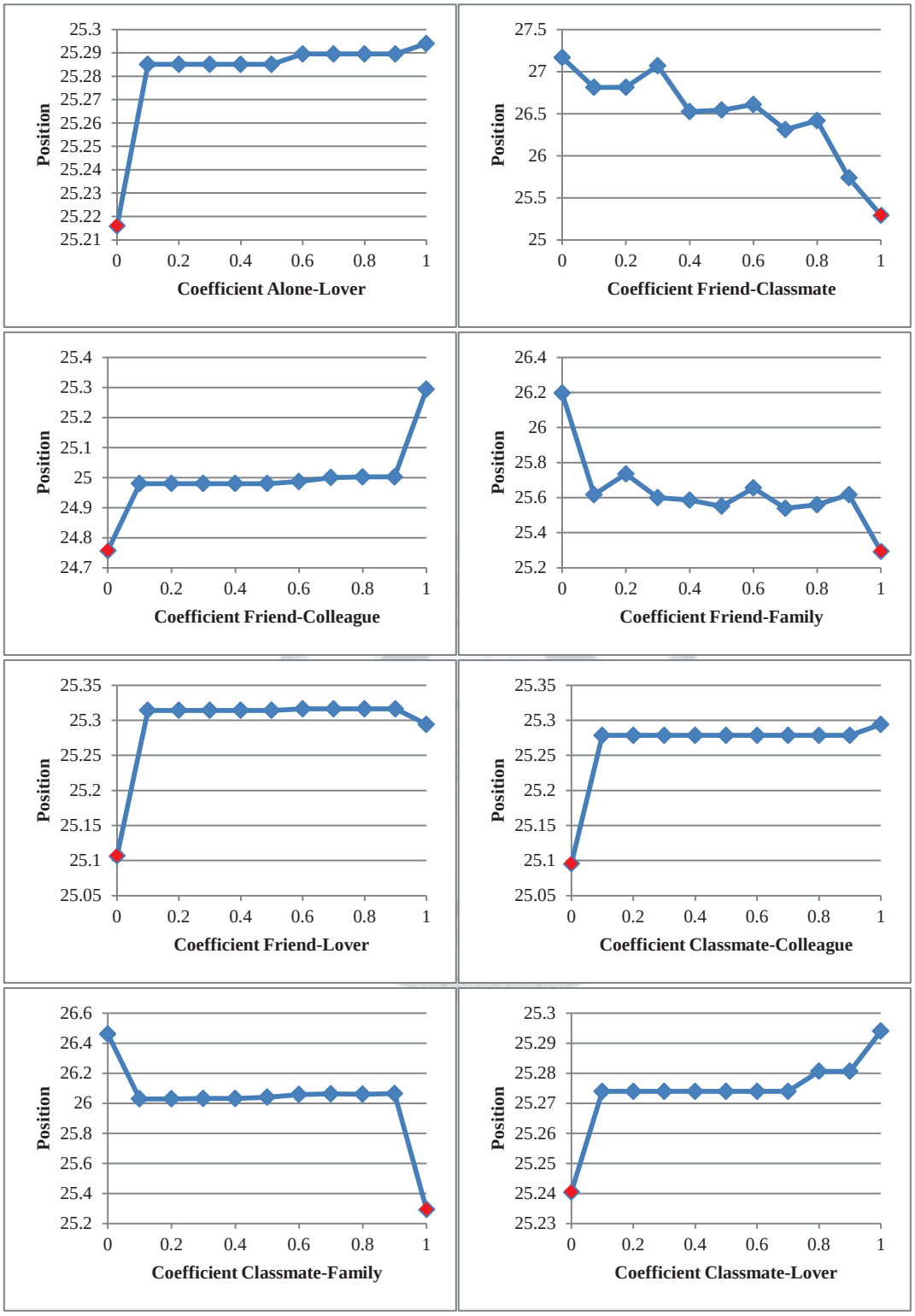
為了取得章節 3.3.3 中，情境相關性測量模型的各項係數，我將利用 LOOCV 與逐步掃描法搜尋最佳係數，讓使用者偏好的餐廳更有可能出現在推薦清單的上方。

一共有 21 個係數需要被調整，我們假設每個係數是獨立的；如果係數是 0，這代表對應之情境維度的值需要分別被考慮（只考慮情境值相同的項目），如果係數是 1，則代表對應之情境維度不影響推薦。

由於 LOOCV 與協同過濾法的時間複雜度相當高，我們僅能將逐步掃描法之階寬調整至合理的範圍，在此我們將階寬設定為 0.1，並針對因個別或團體使用者相似度大於 0 而受情境影響的項目（約占總數 1.76%）進行搜尋，搜尋過程如圖 5.8，搜尋到的最佳係數如表 5.4：




Position
25.5
25
24.5
24
23.5
23
22.5
22
21.5
0 0.2 0.4 0.6 0.8 1
Coefficient Time of Day
Position
26.4
26.2
26
25.8
25.6
25.4
25.2
25
24.8
0 0.2 0.4 0.6 0.8 1
Coefficient Weekday
Position
35
30
25
20
15
10
5
0
0 0.2 0.4 0.6 0.8 1
Coefficient Month
Position
35
30
25
20
15
10
5
0
0 0.2 0.4 0.6 0.8 1
Coefficient Number of Companion
Position
35
30
25
20
15
10
5
0
0 0.2 0.4 0.6 0.8 1
Coefficient Alone-Friend
Position
25.4
25.3
25.2
25.1
25
24.9
24.8
0 0.2 0.4 0.6 0.8 1
Coefficient Alone-Classmate
Position
25.6
25.55
25.5
25.45
25.4
25.35
25.3
25.25
0 0.2 0.4 0.6 0.8 1
Coefficient Alone-Colleague
Position
25.9
25.8
25.7
25.6
25.5
25.4
25.3
25.2
0 0.2 0.4 0.6 0.8 1
Coefficient Alone-Family


Position
Coefficient Alone-Lover
Coefficient Friend-Classmate
Coefficient Friend-Colleague
Coefficient Friend-Family
Coefficient Friend-Lover
Coefficient Classmate-Colleague
Coefficient Classmate-Family
Coefficient Classmate-Lover

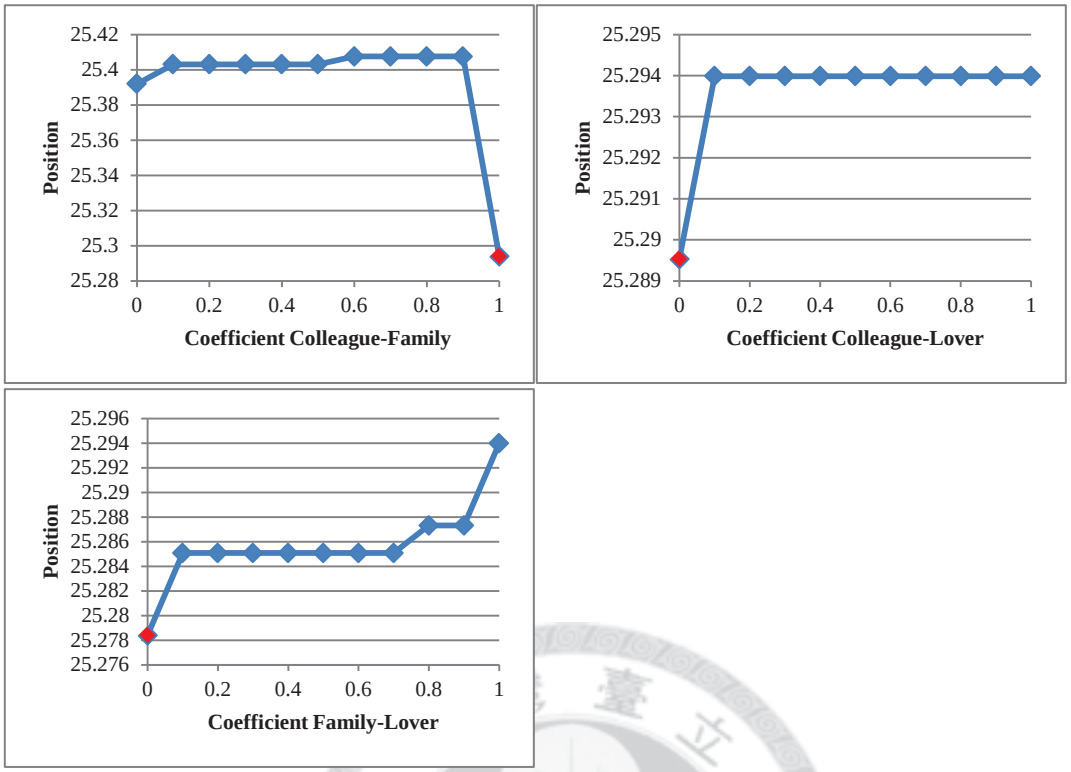


圖 5.8 情境相關性測量模型之各項係數值的平均命中位置（紅點為最佳係數）

$Coef_a$	$Coef_s$	$Coef_t$	$Coef_w$	$Coef_m$	$Coef_{nc}$
1	1	0.1	0.4	1	1

	Alone	Friend	Classmate	Colleague	Family	Lover
Alone	1	1	0.8	1	1	0
Friend	1	1	1	0	1	0
Classmate	0.8	1	1	0	1	0
Colleague	1	0	0	1	1	0
Family	1	1	1	1	1	0
Lover	0	0	0	0	0	1

表 5.4 總結情境相關性測量模型的各項最佳係數

5.4 評價

如果使用者僅對一小部分的項目評分，那麼推薦準確度低不代表系統表現不好；這是因為系統推薦了很多使用者沒有評分的項目，很可能他們原本就非常喜歡這些項目[24]。

準確度高的推薦系統也不能保證使用者對推薦結果滿意；如果系統推薦流行項目給使用者，大幅提高準確度，但使用者很可能早已知道這些資訊，因此使用者不會認為這樣的推薦系統是有價值的；一般而言，系統推薦非流行項目會降低準確度，但使用者反而容易發現一些新奇、自己找不到的偏好項目[24]。

因此本系統以協同過濾法為基礎，從個別使用者偏好總結團體偏好，並藉由收集到的社交圖與情境資訊增進系統性能，期望產生賓主盡歡的餐廳推薦結果；而協同過濾的動機來自於人們通常會從品味相似的人得到最好的建議[25]。

最後我將應用搜尋到的最佳係數和 LOOCV 在真實世界資料集中評價系統性能；為了比較每個情境維度的影響，我移除不影響推薦的情境維度，觀察個別或

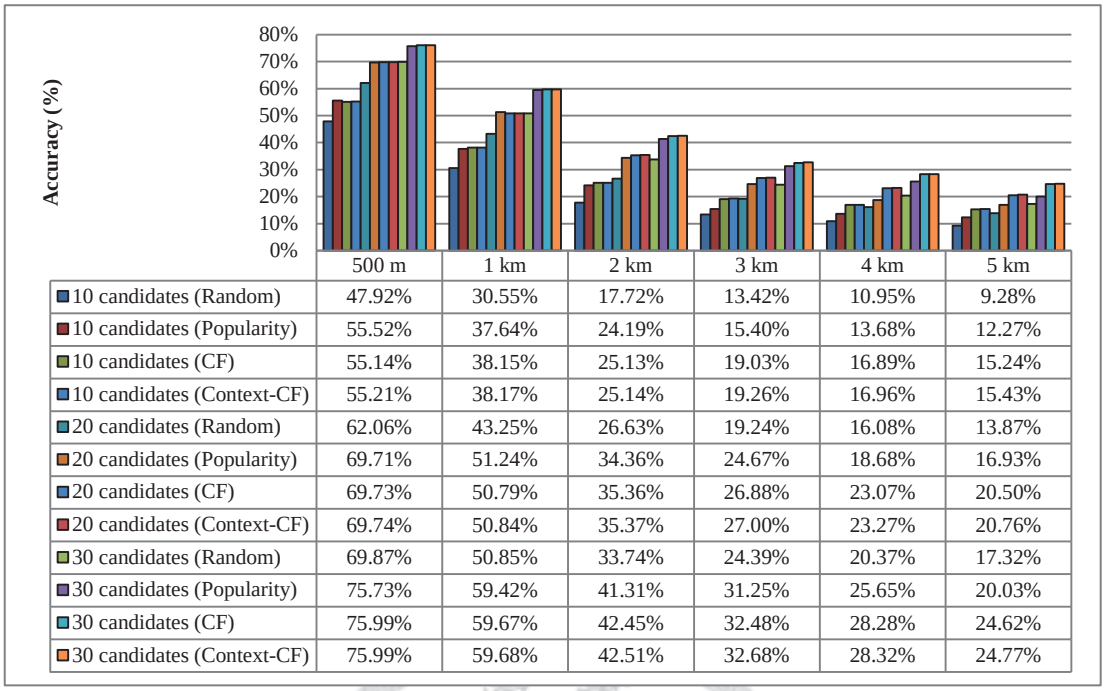
團體使用者相似度大於 0 而受情境影響的項目（約占總數 1.76%），做了 5 輪測試，如表 5.5：

	Average Position	**T-Test (P Value)**
Round 1: Non-contextual information	25.29	1
Round 2: Time of Day	22.02	0.11
Round 3: Weekday	24.98	0.89
Round 4: Type of Companion	24.18	0.62
Round 5: All	21.22	0.04

表 5.5 各情境維度的影響程度

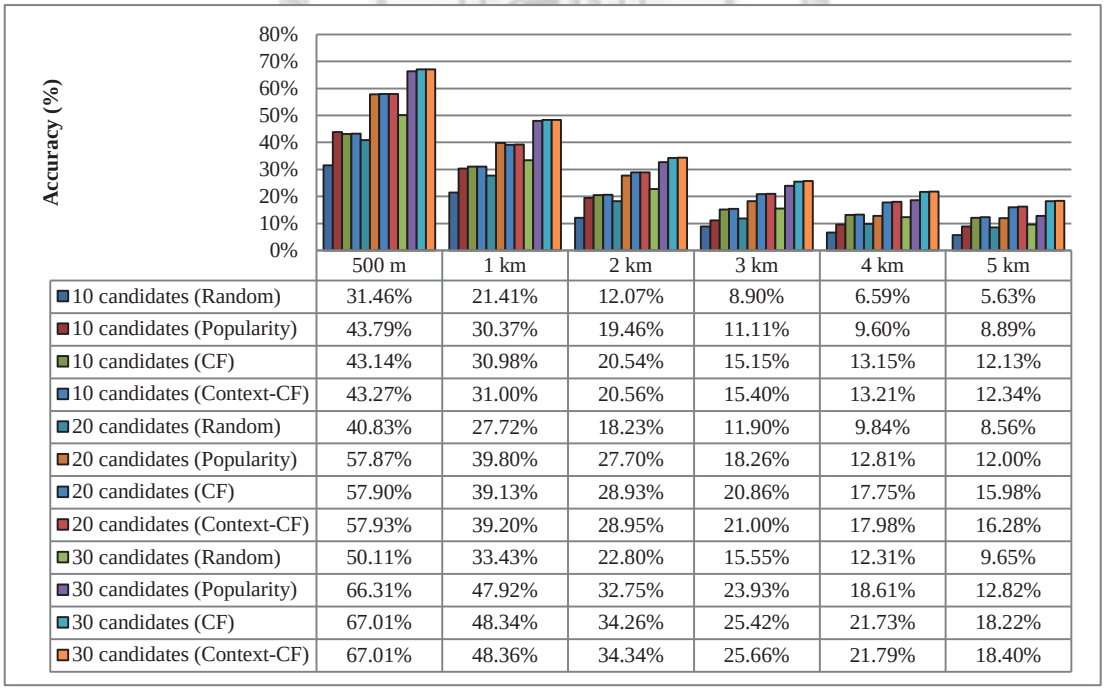
如表 5.5 所示，考慮個別情境雖然看似有所改進，但其 T-Test 之 P Value 卻與 Round 1 差不多；不過當我們進一步考慮所有情境後，即為受情境影響的項目帶來了顯著的改進，T-Test 之 P Value 證實其為顯著水準（Significant Level）。

最後，我們分別將本系統的推薦數量固定為 10、20 和 30 家候選餐廳，距離參數固定為 500 公尺、1、2、3、4 和 5 公里，與基於亂數和基於流行性推薦（被打卡次數多的餐廳優先推薦）的方法進行比較，並對原始結果和篩選推薦數量小於該距離參數之餐廳數量的情況，各別進行評價，如圖 5.9 和圖 5.10。為了更容易比較系統性能，我們計算了本系統對基於流行性推薦的準確度成長率，如圖 5.11。



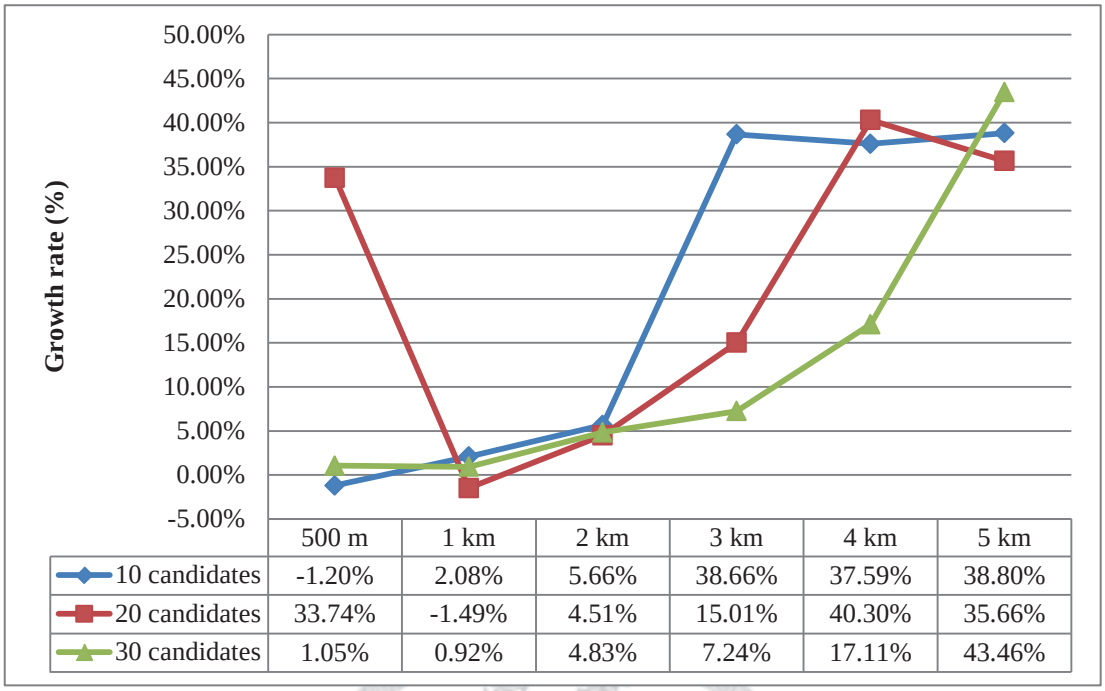
	500 m	1 km	2 km	3 km	4 km	5 km
10 candidates (Random)	47.92%	30.55%	17.72%	13.42%	10.95%	9.28%
10 candidates (Popularity)	55.52%	37.64%	24.19%	15.40%	13.68%	12.27%
10 candidates (CF)	55.14%	38.15%	25.13%	19.03%	16.89%	15.24%
10 candidates (Context-CF)	55.21%	38.17%	25.14%	19.26%	16.96%	15.43%
20 candidates (Random)	62.06%	43.25%	26.63%	19.24%	16.08%	13.87%
20 candidates (Popularity)	69.71%	51.24%	34.36%	24.67%	18.68%	16.93%
20 candidates (CF)	69.73%	50.79%	35.36%	26.88%	23.07%	20.50%
20 candidates (Context-CF)	69.74%	50.84%	35.37%	27.00%	23.27%	20.76%
30 candidates (Random)	69.87%	50.85%	33.74%	24.39%	20.37%	17.32%
30 candidates (Popularity)	75.73%	59.42%	41.31%	31.25%	25.65%	20.03%
30 candidates (CF)	75.99%	59.67%	42.45%	32.48%	28.28%	24.62%
30 candidates (Context-CF)	75.99%	59.68%	42.51%	32.68%	28.32%	24.77%

圖 5.9 各方法與距離參數之推薦準確度



	500 m	1 km	2 km	3 km	4 km	5 km
10 candidates (Random)	31.46%	21.41%	12.07%	8.90%	6.59%	5.63%
10 candidates (Popularity)	43.79%	30.37%	19.46%	11.11%	9.60%	8.89%
10 candidates (CF)	43.14%	30.98%	20.54%	15.15%	13.15%	12.13%
10 candidates (Context-CF)	43.27%	31.00%	20.56%	15.40%	13.21%	12.34%
20 candidates (Random)	40.83%	27.72%	18.23%	11.90%	9.84%	8.56%
20 candidates (Popularity)	57.87%	39.80%	27.70%	18.26%	12.81%	12.00%
20 candidates (CF)	57.90%	39.13%	28.93%	20.86%	17.75%	15.98%
20 candidates (Context-CF)	57.93%	39.20%	28.95%	21.00%	17.98%	16.28%
30 candidates (Random)	50.11%	33.43%	22.80%	15.55%	12.31%	9.65%
30 candidates (Popularity)	66.31%	47.92%	32.75%	23.93%	18.61%	12.82%
30 candidates (CF)	67.01%	48.34%	34.26%	25.42%	21.73%	18.22%
30 candidates (Context-CF)	67.01%	48.36%	34.34%	25.66%	21.79%	18.40%

圖 5.10 各方法與距離參數之推薦準確度（篩選推薦數量小於該距離參數之餐廳數量的情況）



	500 m	1 km	2 km	3 km	4 km	5 km
10 candidates	-1.20%	2.08%	5.66%	38.66%	37.59%	38.80%
20 candidates	33.74%	-1.49%	4.51%	15.01%	40.30%	35.66%
30 candidates	1.05%	0.92%	4.83%	7.24%	17.11%	43.46%

圖 5.11 本系統對基於流行性推薦的準確度成長率（篩選推薦數量小於該距離參數之餐廳數量的情況）

如前所述，一般而言，系統推薦非流行項目會降低準確度；如圖 5.10 和圖 5.11 所示，當本系統推薦 10 家候選餐廳，在距離 500 公尺（步行約 10 分鐘）的情境下，準確度為 43.27%；略低於基於流行性推薦之準確度 43.79%，成長率為-1.2%。在距離 1 公里（步行約 20 分鐘）的情境下，準確度為 31%；開始略高於基於流行性推薦之準確度 30.37%，成長率為 2.08%。在距離 5 公里的情境下（搭公車約 30 分鐘），準確度為 12.34%；高於基於流行性推薦之準確度 8.89%，成長率為 38.8%。這意味著，如果使用者尋找餐廳所設定的範圍比較大，相較於基於流行性推薦，本系統可以產生更好的推薦結果。

6 結論

本論文提出使用社交圖與情境感知之行動餐廳推薦系統；第 3 章我們透過 Graph API 取得社交圖與開放圖中，使用者與其朋友對餐廳的偏好和所有可能影響人們選擇餐廳的因素，我們將這些因素稱為社交圖與情境資訊，並透過系統架構、資料模型、方程式與演算法介紹如何將其整合到推薦系統中；第 4 章敘述實作技術與流程，並展示使用者介面；第 5 章的實驗結果顯示了本系統卓越的性能。

在本章中，章節 6.1 將總結我們所做的工作和得到的成果，章節 6.2 說明系統未來發展與研究方向。

6.1 總結貢獻

本論文利用 Facebook 開放圖（Open Graph）的打卡資料（Check-ins）設計一個行動餐廳推薦系統，並實作到 Windows Azure Platform 上。系統以協同過濾法（Collaborative Filtering）為基礎，從個別使用者偏好總結團體偏好，實現團體推薦服務；並考慮社交圖（Social Graph）與情境資訊（Contextual Information）提升推薦品質

一般而言，系統推薦非流行項目會降低準確度；實驗結果顯示，本系統在中、長距離（3 到 5 公里）的情境下，準確度相較於基於流行性推薦有顯著成長，成長率約 38%。這意味著，如果使用者尋找餐廳所設定的範圍比較大，相較於基於流行性推薦，本系統可以產生更好的推薦結果。

6.2 未來工作

如表 5.1 和表 5.3 所示，1 位使用者平均有 504 位朋友，扣除重複的對象，約可帶來 365 位使用者的資料（收集到的使用者人數/受測者人數），因此本系統隨著使用者增加，可參考的相似使用者數量會大幅成長，將有助於進一步提升推薦品質，但其計算量也將成線性加大；對於行動裝置來說，回應速度是影響使用者體驗最重要的因素之一，因此如何改善其可擴展性（Scalability）以及如何維護個人隱私（Privacy）是值得深入研究的議題。

參考文獻

1. 刘建国，周涛, and 汪秉宏，*个性化推荐系统的研究进展*. 自然科学进展, 2009. **19**(001): p. 1-15.
2. Schilit, B.N. and M.M. Theimer, *Disseminating active map information to mobile hosts.* Network, IEEE, 1994. **8**(5): p. 22-32.
3. Dey, A.K., *Understanding and using context.* Personal and ubiquitous computing, 2001. **5**(1): p. 4-7.
4. Woerndl, W. and J. Schlichter. *Introducing context into recommender systems*. in *Proceedings of AAAI 2007 Workshop on Recommender Systems in e-Commerce*. 2007.
5. Wikipedia contributors. *Social graph*. Available from: http://en.wikipedia.org/w/index.php?title=Social_graph&oldid=495500805.
6. Facebook. *Open Graph*. Available from: http://developers.facebook.com/docs/opengraph/.
7. Wikipedia contributors. *Check-in*. Available from: http://en.wikipedia.org/w/index.php?title=Check-in&oldid=495397467.
8. Facebook. *Graph API*. Available from: http://developers.facebook.com/docs/reference/api/.
9. Goldberg, D., et al., *Using collaborative filtering to weave an information tapestry.* Communications of the ACM, 1992. **35**(12): p. 61-70.
10. Konstan, J.A., et al., *GroupLens: applying collaborative filtering to Usenet news.* Communications of the ACM, 1997. **40**(3): p. 77-87.
11. Adomavicius, G. and A. Tuzhilin, *Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions.* Knowledge and Data Engineering, IEEE Transactions on, 2005. **17**(6): p. 734-749.
12. Sarwar, B., et al. *Item-based collaborative filtering recommendation algorithms*. in *Proceedings of the 10th international conference on World Wide Web*. 2001. ACM.
13. Claypool, M., et al. *Combining content-based and collaborative filters in an online newspaper*. in *Proceedings of ACM SIGIR Workshop on Recommender Systems*. 1999. Citeseer.
14. Pazzani, M.J., *A framework for collaborative, content-based and demographic filtering.* Artificial Intelligence Review, 1999. **13**(5): p. 393-408.
15. Balabanović, M. and Y. Shoham, *Fab: content-based, collaborative recommendation.* Communications of the ACM, 1997. **40**(3): p. 66-72.

16. Adomavicius, G., et al., *Incorporating contextual information in recommender systems using a multidimensional approach.* ACM Transactions on Information Systems (TOIS), 2005. **23**(1): p. 103-145.
17. Chen, A., *Context-aware collaborative filtering system: Predicting the user's preference in the ubiquitous computing environment.* Location-and Context-Awareness, 2005: p. 75-81.
18. Nguyen, Q.N. and F. Ricci. *Long-term and session-specific user preferences in a mobile recommender system*. in *Proceedings of the 13th international conference on Intelligent user interfaces*. 2008. ACM.
19. Sadeh, N., E. Chan, and L. Van. *MyCampus: an agent-based environment for context-aware mobile services*. in *Proceedings of Workshop on Ubiquitous Agents on embedded, wearable and mobile devices*. 2002.
20. *黃啟嘉, 情境資訊對智慧型裝置上餐廳推薦系統的影響分析*, in *臺灣大學資訊工程學研究所學位論文* 2009, *臺灣大學*.
21. Park, M.H., H.S. Park, and S.B. Cho. *Restaurant recommendation for group of people in mobile environments using probabilistic multi-criteria decision making*. in *Proceedings of the 8th Asia-Pacific conference on Computer-Human Interaction*. 2008. Springer.
22. Wikipedia contributors. *Cloud computing*. Available from: http://en.wikipedia.org/w/index.php?title=Cloud_computing&oldid=499416499.
23. Sarwar, B., et al. *Analysis of recommendation algorithms for e-commerce*. in *Proceedings of the 2nd ACM conference on Electronic commerce*. 2000. ACM.
24. *刘建国*, et al., *个性化推荐系统评价方法综述*. *复杂系统与复杂性科学*, 2009. **6**(003): p. 1-10.
25. Wikipedia contributors. *Collaborative filtering*. Available from: http://en.wikipedia.org/w/index.php?title=Collaborative_filtering&oldid=495504334.

附錄

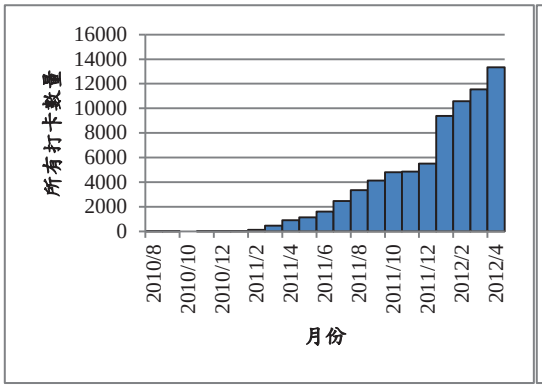
項目	比率	
曾打卡的使用者男女比(男/女)	1.02	3719 / 3654
曾打卡的使用者比率	29.36%	7395 / 25184
打卡在大台北地區的比率	63.64%	47277 / 74283
地點在大台北地區的比率	49.74%	18556 / 37305
曾對大台北地區地點打卡的使用者比率	76.50%	5657 / 7395

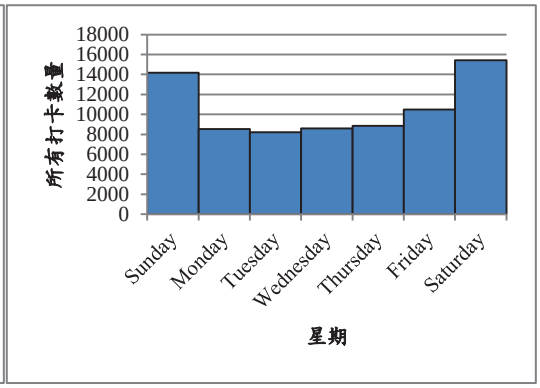
附錄 1 收集到的各項資料之比率（所有地點）

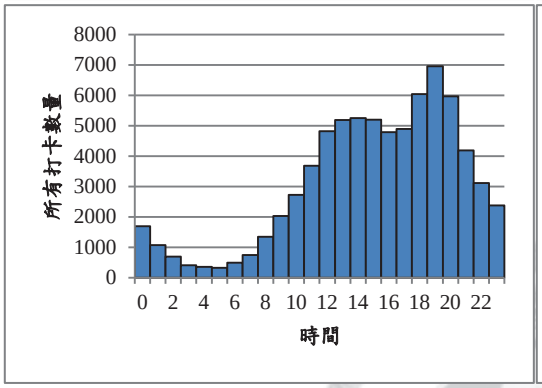
附錄 2 收集到的所有地點之分布圖

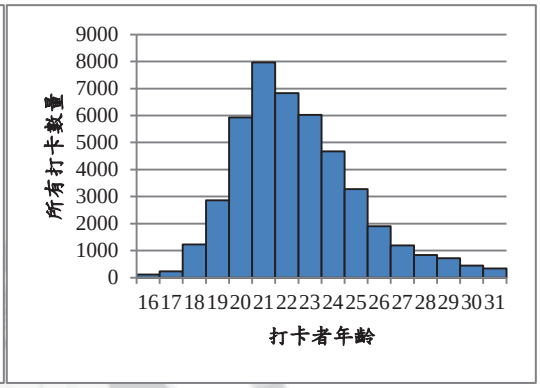
項目	平均值	中位數	標準差	最大值	最小值	
曾打卡的使用者年齡	22.91	22	4.56	106	13	n=7395
使用者的打卡次數	3.27	0	6.61	40	0	n=25184
使用者打卡過的地點個數	2.95	0	5.99	39	0	n=25184
地點被打卡的次數	2.21	1	6.00	256	1	n=37305
曾打卡的使用者打卡次數	11.14	11	7.83	40	1	n=7395
曾打卡的使用者打卡過的地點個數	10.05	10	7.13	39	1	n=7395
所有打卡的打卡者年齡	23.05	22	4.50	106	13	n=74283
所有打卡的地點被打卡次數	1.11	1	0.53	19	1	n=74283
所有打卡的同伴平均年齡	23.68	23	4.58	107	14	n=74283
所有打卡的同伴性別指標(女/總數)	0.51	0.5	0.49	1	0	n=74283
所有打卡的同伴人數	2.36	2	2.42	51	1	n=74283

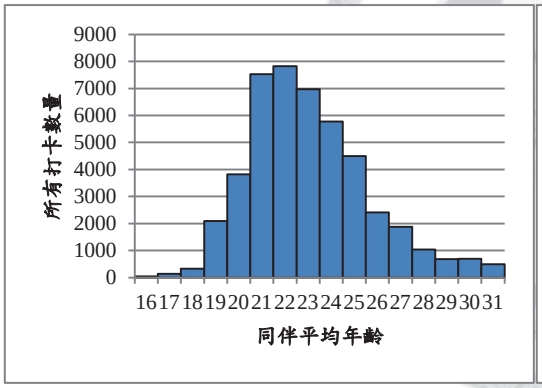
附錄 3 收集到的各項資料之平均值、中位數與標準差（所有地點）

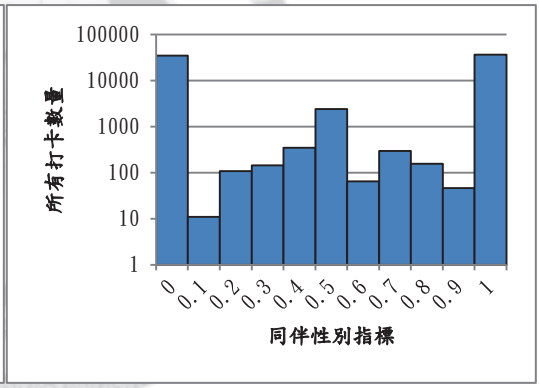

所有打卡數量
16000
14000
12000
10000
8000
6000
4000
2000
0
2010/8
2010/10
2010/12
2011/2
2011/4
2011/6
2011/8
2011/10
2011/12
2012/2
2012/4
月份

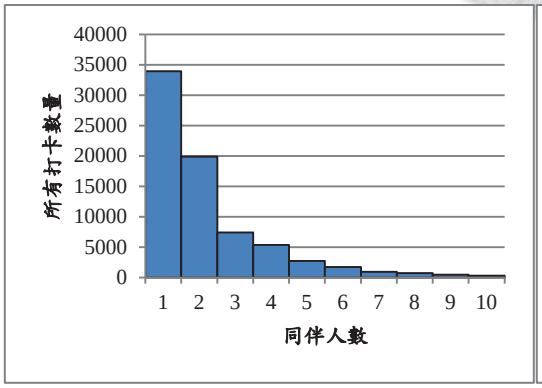

所有打卡數量
18000
16000
14000
12000
10000
8000
6000
4000
2000
0
Sunday
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
星期

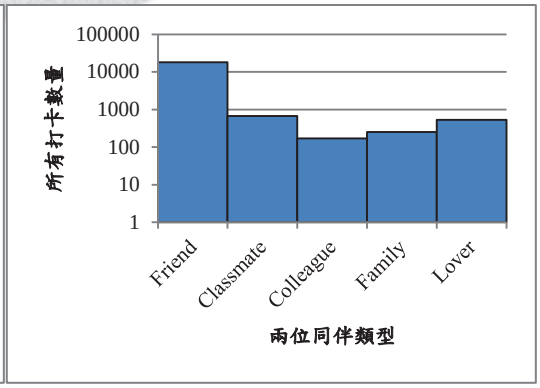

所有打卡數量
8000
7000
6000
5000
4000
3000
2000
1000
0
0 2 4 6 8 10 12 14 16 18 20 22
時間

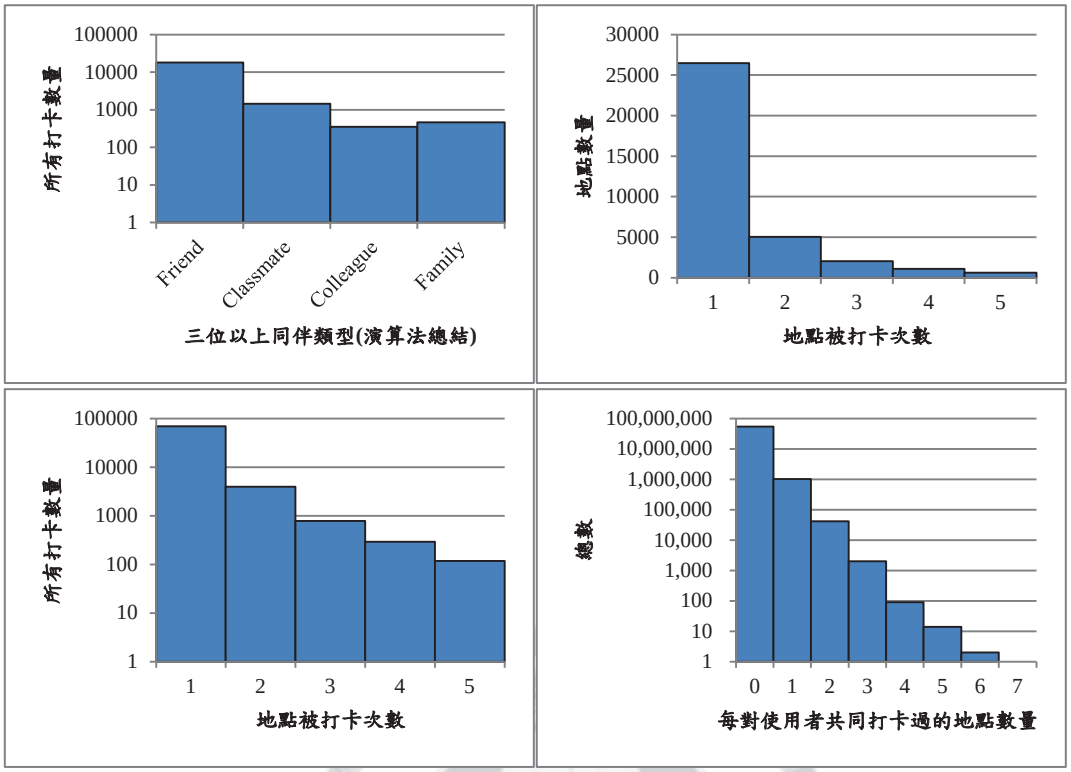

所有打卡數量
9000
8000
7000
6000
5000
4000
3000
2000
1000
0
16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
打卡者年齡


所有打卡數量
9000
8000
7000
6000
5000
4000
3000
2000
1000
0
16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
同伴平均年齡


所有打卡數量
100000
10000
1000
100
10
1
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
同伴性別指標


所有打卡數量
40000
35000
30000
25000
20000
15000
10000
5000
0
1 2 3 4 5 6 7 8 9 10
同伴人數


所有打卡數量
100000
10000
1000
100
10
1
Friend
Classmate
Colleague
Family
Lover
兩位同伴類型



附錄 4 統計收集到的所有打卡資料