基於行事曆的行動應用程式語意推薦方法
Semantic Recommendation for Mobile Apps: A Calendar-Based Approach

林睿祥、莊晏、藍孟彬、馬尚彬
國立臺灣海洋大學
10357027@ mail.ntou.edu.tw, 00157004@mail.ntou.edu.tw,
00257112@mail.ntou.edu.tw, albert@ntou.edu.tw

摘　要

現今，行動應用程式(Mobile App 或簡稱 App)已被廣泛的使用，因應使用者多元的需求，多樣且大量的各式 App 持續被開發出來，於此同時，情境資訊推薦系統 (Context-Aware Recommender Systems 簡稱 CARS) 越來越被業界與學界重視，可用以推薦符合使用者情境之資訊。本研究利用行事曆(Calendar)作為情境資訊來源，提出一套行動應用程式之語意推薦方法，我們稱之為 SRMA (Semantic Recommendation for Mobile Apps)，本研究主要的三大特點包含：(1)利用語意相似度方法計算出行事曆與 App 的相關程度；(2)利用語意相似度方法進行 App 之分類；以及(3)根據使用者行事曆之資訊，推薦出符合使用者日常生活需求之 App。此外，利用實驗證實 SRMA 方法可實際有效地推薦符合使用者預期之 App。

關鍵字：App Recommendation, Google Calendar, CARS (Context-Aware Recommender Systems), WordNet, Information Retrieval

一、緒論

由於行動應用程式(Mobile App 或簡稱 App)廣為大眾所使用，市面上可被下載的各種類 App 數量與日俱增[1, 2]，App 已成為了人們生活中不可或缺的存在，亦成為重要的軟體遞送模式[3]。2015 年時，Google Play 平台中就發佈了超過 160 萬支 Android App [4]，因此，如何有效率的搜尋、推薦、管理 App 已成為一個重要的技術議題。

近年來，情境資訊推薦系統(CARS)已是業界與學界中廣被重視及研究的方向[5]。CARS 因考量使用者在特定情境中的因素，故可有效地使推薦準確度提升，一般 CARS 分為以下三種型態：contextual prefiltering、contextual postfiltering、contextual modeling [6]，三者並無優劣之分，並可根據不同的需求條件應用較合適 CARS。除此之外，一般 CARS 通常會需要長時間的監控與存取使用者資料，這很有可能會侵犯到使用者的隱私。

根據上述背景，本研究主要採用 contextual prefiltering 之概念，提出了行動應用程式的語意推薦方法：Semantic Recommendation for Mobile Apps (SRMA)。SRMA 的核心為基於辭彙語意(lexical semantics)的相似度計算方法，用以分析使用者的行事曆與 App 之間的相關程度。SRMA 也提供一套 App 分類機制，可將相似的 App 進行分類，以利於後續的過濾與搜尋程序。最後，SRMA 提供一套基於行事曆的推薦方法，可找出與使用者日常生活需求相關之 App。為了避免長期的背景資料監控與不必要的隱私侵犯，SRMA 僅運用使用者主動提供之行事曆資料來擷取情境資訊，無須持續監控使用者的 App 使用歷程或瀏覽資料。

剩下的章節將依下列順序論述：第二章將介紹相關研究、第三章將敘述本方法的詳細內容、第四章將舉出使用者案例與評估方法的實驗、最後一章將對本方法之效益與特點進行總結。

二、相關研究

在本章節中，我們將回顧與比較幾個與 SRMA 相關之研究。

Datta [7, 8]利用公開的資訊來搜尋 App，而 App 搜尋的排名分數是以該 App 在其主要分類的排名，與該 App 開發者的所有 App 的平均評分計算得之。Appolicious [9]為與 Facebook 服務結合的 App 搜尋引擎，其利用使用者的好友及使用者喜好的社群資訊來提供個人化的推薦。MW (Mobilewalla) [10]也是 App 搜尋引擎，其利用一套計算 App 分數的系統，稱為 MWS (Mobilewalla Score)來生成“Top App”推薦列表。Yang et al. 利用 “Small-Crowd” model [11]透過分析社群網站產生個人化的 App 推薦，該方法概念為：安裝相同的 App 使用者通常擁有相同的嗜好。“Slope One” algorithm [12]為推薦系統常用的技術之一，其可辨別使用者有興趣的項目，並應用 CF (Collaborative Filtering) 來預測各個項目的評分。

雖然以上這些方法在某些觀點上具有不錯的效果，但皆未考慮使用者的情境資訊。一般 CARS (Context-aware Recommendation systems) 通常以三種型態被實現：contextual prefiltering、contextual postfiltering、contextual modeling [13]，此三種型態間沒有優劣之分，可根據不同需求條件運用不同的 CARS，此外，一般的 CARS 通常需要存取使用者的私有資料並長期監控其使用狀況[13]，相較於一般 CARS，SRMA 較貼近使用者的日常生活，且分

析使用者主動提供的行事曆做為情境資訊，可降低對其隱私資料的需求，以增加使用者的使用意願。

三、方法描述

在此章節中，我們將詳細的描述 SRMA 方法的六個核心程序：(1)收集 App 與行事曆資料；(2)分析 App 與行事曆資料；(3)決定 App 的關鍵字(keywords)與擷取行事曆的關鍵字(queries)；(4)以二階層類別結構分類 App；(5)根據擷取出的 queries 進行相似度分數計算；以及(6)推薦 App 給使用者。

圖 1 為 SRMA 之系統架構圖，使用者可透過 SRMA 的使用者介面登入系統，並提供其行事曆資料與其曾經安裝過的 App 資訊，即可獲得其個人化 App 推薦列表。其提供的行事曆資料與 App 資訊會被存入 User & App 資料庫，行事曆資料將會被 Calendar Data Handler 分析，為了藉由 WordNet 搜尋 App，Calendar Data Handler 需由行事曆中擷取 queries。App Data Handler 將存於資料庫中支 App 與其在 Google Play 相對應的網頁資訊分析出該 App 的關鍵字，並將每支 App 分類到我們提出的二階層結構類別中，且會定時更新所有 App 資訊。App Scoring Handler 利用行事曆擷取出的 queries，計算 queries 與 App 之間的 QS 分數(Queries Score)，Recommendation Generator 透過以 QS 找到的 App 列表產生出針對使用者的個人化 App 推薦列表，並透過使用者介面呈現給使用者。

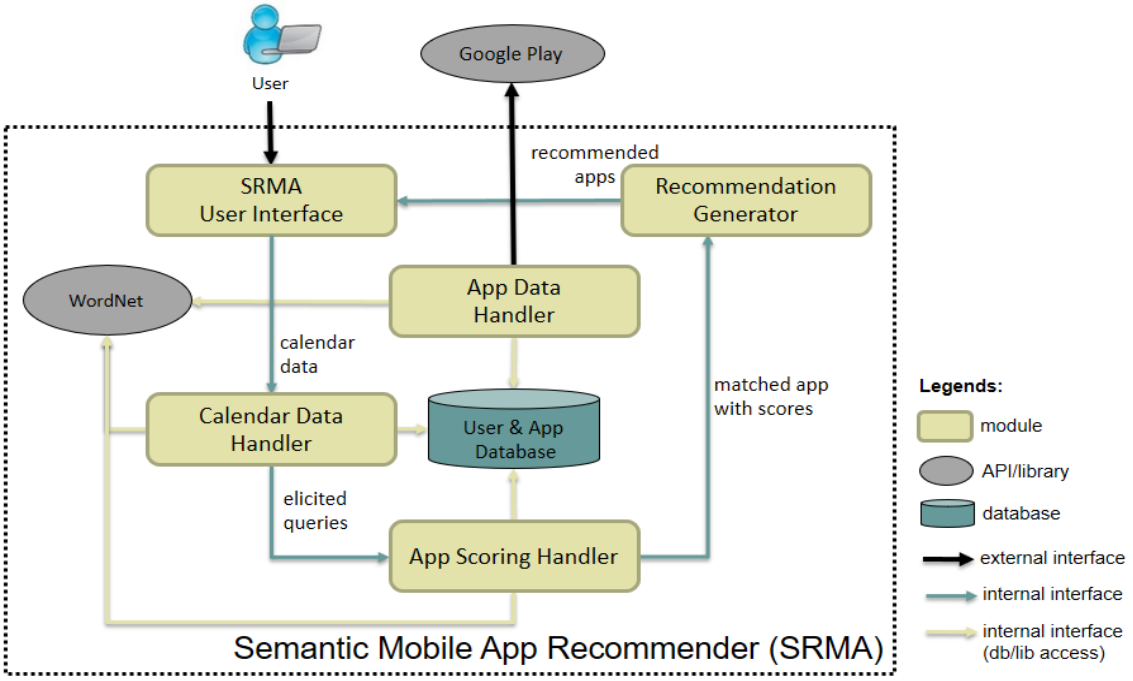


圖 1　SRMA 架構圖

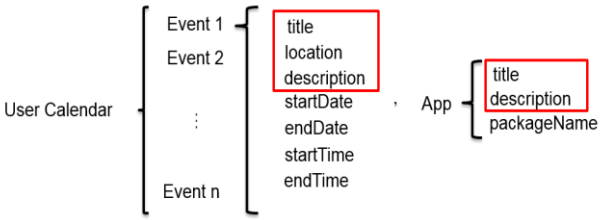
(1)　收集 App 與行事曆資料

在開始使用 SRMA 系統之前，使用者需使用其 Google 帳號登入系統，SRMA 會先進行帳號驗證，並將驗證過的使用者帳號與其曾安裝過的 App 資訊存入資料庫中的 *userApp* 資料表中，如表 1。使用者通過驗證後即可上傳其 Google 行事曆匯出的 ICS 格式檔案(i.e. the iCalendar format)，以進行後續的字詞分析。

表 1　*userApp* 資料表

user	*packageName*
john1234	com.facebook.katana
john1234	tw. edu .ntou
mary999	tw.edu.ntou
mary999	com.instagram.android

(2)　分析 App 與行事曆資料



在推薦 App 之前需將 App 與行事曆中較重要的訊息萃取出來，因此，我們透過以下 4 步驟的字詞分析機制取出具代表性的關鍵字詞：原始資料擷取、斷詞、字詞分類、英文字詞處理。

圖 2 原始資料文字結構

在第一步驟中，為了擷取出使用者行事曆原始資料與各 App 網頁中有用的資料，我們設計了如圖 2 中所示之原始資料文字結構。

表 2 以兩支 App 為例之 *AppInfo* 資料表

appName	*packageName*	*description*	*keyword*	*type*
Language Translator	laclaveganadora.translator	This Application is perfect to translate texts…	[{"count":2, "keyword":"translator", "pos":"FW"},…	tool
RemindMe	com.aspartame.RemindMe	RemindMe is a lightweight App to create...	[{"count":1, "keyword":"remindme", "pos":"Title"}]	note

在 App 部分，SRMA 分別將 App 的標題(title)、內容說明(description)、及特有的套件名稱(packageName)存入 *AppInfo* 資料表中，如表 2。因為每支 App 的 packageName 都不相同，故將其做為資料表中的主鍵(primary key)，並透過 jsoup [14]從對應的 Google Play 網頁中抓取 App 的其他屬性資料。圖中紅色框中的資料為下個步驟所需的資料，其中包括行事曆的 title、locations、description 與 App 的 title 和 description。

第二步驟的處理程序為將文字資料斷詞後取出有意義的字詞，透過中研院提供的 CKIP 斷詞服務[15]將原始的一段文字資料分割成個別的字詞，經過斷詞後，SRMA 將 App 與行事曆 title 中的字詞都保留下來，而剩下部分中的字詞只保留名詞，舉例來說：“LinkedIn Job Search”為 App 的 title，斷詞後則保留“linkedin”、“job”、“search”等字詞；“RemindMe is a lightweight App”為 App description 中的一個句子，其斷詞後只保留了 3 個名詞字詞“remindme”、“lightweight”、 “app”。

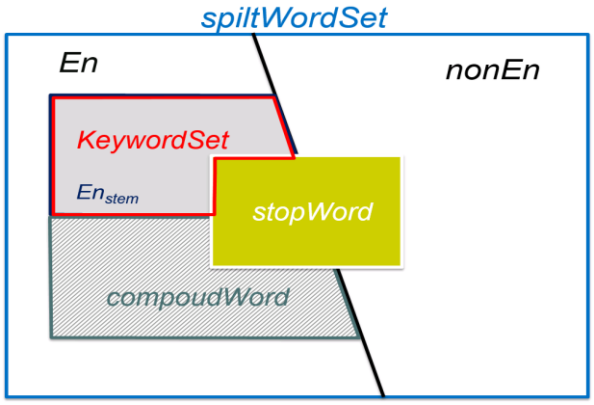


圖 3 *spiltWordSet* 集合的內部集合關係圖

第三步驟為將字詞進行分類，以形成如圖 3 的多個集合。*splitWordSet* 集合代表所有第二步驟斷詞後的字詞集合，其又分為 *En* 集合與 *nonEn* 集合，*En* 為所有英文字詞集合，*nonEn* 為所有非英文的字詞集合，在 *nonEn* 的所有字詞皆會經過 Microsoft 詞彙服務 API [16]翻譯成英文，並將 *nonEn* 中可被翻譯成英文字詞的集合加入 *En* 中。

第四步驟為 *En* 中英文字詞的處理，SRMA 利用 WordNet [17]提供的字詞形態還原(lemma)功能，將英文字詞還原成最原始的型態，而被還原過的字詞集合被記為 En_{stem}，同時，我們將英文複合字詞集合(*compoundWord*)刪除來避免複合字衍生的問題，如：“night market”不論是以“night”或是“market”都與原意不合，可能導致最後推薦不相關的 App，如：“night camera”。除此之外，也必須將停止字集合(*stopWord*)去除，如：“get”、“set”“thing”、“stuff”、“good”、“useful”等等來避免不必要的雜訊產生。將 En_{stem} 集合中與 *compondWord* 和 *stopWord* 交集部份字詞的去除，剩下的字詞集合(*KeywordSet*)為最後我們萃取出的 App 與行事曆的關鍵詞之字詞集合。

(3) 決定 App 的 keywords 與擷取行事曆的 queries

在此小節將詳細說明如何以 IR (Information Retrieval) [18]技術，利用分析後的字詞集合來決定 App 的 keywords 以及擷取行事曆 queries 的方法，如表 2 所示，App 的各個屬性皆已存於資料庫中的 *AppInfo* 表中，為了決定某 App 的 keywords 屬性，SRMA 以某 App 之 *KeywordSet* 中的每個字詞作為 query，透過 Google Play 的搜尋功能來記錄每個能搜尋到該 App 的 query，而這些 queries 即為能夠代表該 App 之字詞，故以這些 queries 為某 App 的 keywords，此外，為了避免某些 App 無法透過上述方法被搜尋到，導致其沒有任何的 keywords，故在決定 keywords 時，將 *AppName* 也當作一個 keyword，最後將這些決定好的 keywords，以易於處理的 JSON [19] (JavaScript Object Nation)格式存入 *AppInfo* 表的 *keyword* 屬性中。

行事曆 *KeywordSet* 的字詞中抽出較重要的字詞作為行事曆的 queries，SRMA 將利用行事曆抽出的 queries 作為推薦使用者 App 的依據，為了要決定這些 queries 中每個字詞的重要程度，我們設定了以下規則：1)以整份行事曆視為文集(corpus)，每個事件視為文件(document)；2)計算每個事件中各字詞的 TF-IDF (term frequency-inverse document frequency)值[20]；3)計算所有字詞的平均 TF-IDF 值；4)將低於平均 TF-IDF 值的字詞刪除；5)將每個事件中剩下的字詞作為該行事曆抽出的 queries。若經過上述規則抽出的 queries 數量過少，無法計算出相似的 App 時，我們利用放寬機制來調整刪除 queries 的數量，以平均 TF-IDF 值為門檻值(threshold)，每次放寬將門檻值減半直到平均 TF-IDF 十分之一為止，即最多放寬 5 次。

(4) 以二階層類別結構分類 App

將推薦結果分類為普遍推薦系統主要技術之一，因為使用者可能會不想看到與其預期推薦類別不相關的推薦結果，故我們將 SRMA 上存在的所有 App 分類，使用者可利用過濾類別機制調整他們的推薦列表，我們參考 Zhu et al.的二階層類別結構[21]，提出調整 Google Play 上的類別種類後，如表 3 的二階層類別結構。在分類結構中有 7 個第一階層的類別與"others"類別，用來歸類無法即時被適當分類的 App。

在每個第一階層下各有 3 個第二階層的類別與 1 個如第一階層的"others"類別，每個類別中都有各自的 type corpus，這些 type corpus 皆由各自類別中 App 的 keywords 組成。當有新的 App 被分類入某類別時，該 App 的 keywords 亦會加入某類別中，以擴充其 type corpus。

表 3 二階層分類結構

Level-1	***Level-2***			
entertainment	reference	Shopping	food	others
multimedia	music	Video	photography	others
utility	tool	management	personal	others
life	weather	Sport	note	others
communication	sociality	message	phone	others
business	news	finance	education	others
navigation	travel	Map	transportation	others
others	If the App cannot be classified, it can be classified by manually			

我們將計算 App 與各類別之間的相似度來作為分類的依據，相似度計算的規則包括：1)計算每個類別的 type corpus 中各個 keywords 的 TF-IDF 值；2)利用如公式 (1)基於 WordNet 的 *Wu & Palmer* 方法[22]來計算 App a_i 中的 keyword k_m 與 type corpus t_j 中的 keywords w_n 之字詞相似度 *WuS* 值；3)計算 t_j 中所有 w_n 與某 k_m 的 *WuS* 值並乘上 w_n 的 TF-IDF 值作為該 k_m 相似度值；4)找出與某 k_m 有最大相似度值的 w_n，作為該 k_m 與 t_j 的相似度；5)將每個 k_m 與 t_j 的相似度平均，作為 a_i 對 t_j 的類別相似度分數；6)找出類別相似度分數最高的 t_j 並將 a_i 分類到該類別。圖 4 為利用目前 SRMA 中所有使用者曾經安裝過的 1,006 支 App 進行分類後的的分類分佈圖。

$$WuS = \frac{2*depth(LCS)}{(depth(s1)+depth(s2))}, 0 <= WuS <= 1, \quad if \quad s1 = s2 \quad then \quad WuS = 1 \tag{1}$$

其中 *s1* 與 *s2* 為欲計算相似度的兩字串，*depth* 為 WordNet 類別(taxomony)中節點(synset)的深度，*LCS* 為距離兩 synset 最接近的父節點(ancestor)。

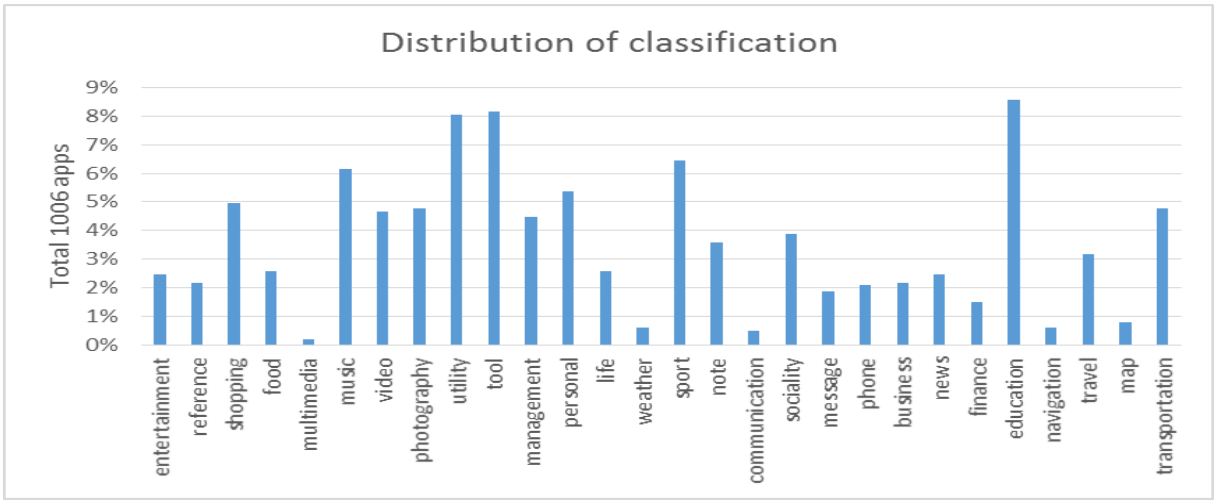


圖 4 App 類別分佈圖

在 SRMA 中並不考慮“game”類別的 App，原因在於遊戲類 App 過於多元，其模糊的 keywords 將造成非遊戲類 App 在分類時中被誤分入“game”類別中，舉例來說：“Virtual City”或“Crime City” 其 keywords 有 “city” 、 “building” 、 “airport” 、 “railway”，若這兩支 App 被分類到“game”中，其 keywords 將被加入“game”的 corpus 中造成雜訊，可能導致其他應該被分類到 “travel” 或 “transportation”類別的 App 被分到“game”中。

(5) 以擷取出的 queries 算分

為了推薦合適的 App 給使用者，SRMA 需計算 App 與行事曆間的分數，利用行事曆擷取出的 queries 搜尋出相關的 App，並將搜尋到的 App 排名後推薦給使用者，搜尋 App 的算分規則與分類的算分規則十分類似，包括：1)以所有 App 作為一個 corpus，每支 App 作為 document，計算各個 document 中 keywords 之 TF-IDF 值；2) 基於 WordNet 的 *Wu & Palmer* 方法來計算行事曆 c_i 中的 queries q_m 與 App a_j 中的 keywords k_n 之字詞相似度 *WuS* 值；3)計算 a_j 中所有 k_n 與某 q_m 的 *WuS* 值並乘上 k_n 的 TF-IDF 值作為該 q_m 相似度值；4)找出與某 q_m 有最大相似度值的 k_n，作為該 q_m 與 a_j 的相似度；5)將每個 q_m 與 a_j 的相似度平均，作為 c_i 對 a_j 的相似度分數，即為 Queries Score (QS)；6)找出 QS 最高的 App a_j 作為搜尋到的 App。所有搜尋到的 App 作為候選 App，再經由下章節的步驟推薦給使用者。

(6) 推薦 App 給使用者

本章節最後的核心程序即為推薦 App 於使用者，首先，SRMA 根據 QS 將搜尋到的候選 App 排名；接著，挑選出前 10 名當作推薦給提供行事曆的使用者的 App；最後，使用者可藉由類別過濾器，依其喜好選擇想被推薦的 App 類別，將非此類別的候選 App 過濾掉。該類別過濾器可讓使用者選擇兩個階層的類別，若選擇第二階層類別，則可過濾其他未被選擇的類別；若選擇第一階層的類別，即等於選擇了在其類別之下的 4 個第二階層類別，過濾器會將其他未被選擇的類別之 App 過濾掉。若推薦出的 App 中有使用者目前正安裝的 App 亦會標示出告知。

四、使用者案例與實驗

此章節將舉例說明使用者如何使用 SRMA 來獲得其 App 推薦列表，與如何將從新使用者曾安裝過的 App 資訊收集到的 App 分類到 SRMA 中的二階層類別，以及評估 SRMA 的使用者對其個別的 App 推薦滿意度。

(1) 使用者案例

使用者為本學期的應屆畢業生，這是他首次使用 SRMA ，在成功登入系統後，他順利的提供了 Google 行事曆匯出的 ICS 檔案，表 4 為其行事曆事件列表的片段。當系統接收到其上傳的行事曆資料後，透過分析 App 與行事曆程序分析出其行事曆中的關鍵字，如：“workout”、“job”、 “birthday”、“photo”、“swim”、“restaurant”、“interview”等等。接著將這些關鍵字作為 queries 計算每支 App 的 QS 搜尋出候選 App，如表 5。

SRMA 最後依據 QS 選出前 10 名的 App 推薦給使用者，如表 6 所示。同時，SRMA 依據其推薦列表產生了類別過濾器如表 7，因為使用者想要找與工作相關的 App，所以他選擇了“business”類別，於是他得到了只有“business”類別 App 的新推薦列表，如表 8，現在使用者的新推薦列表中所有的 App 皆與其需求相符了。

表 4 使用者的片段行事曆之事件列表

Event Title
Go to workout
Find job
Mom’s birthday
Take passport photo
Go to workout
Mary’s birthday
Swim
Find job
Write a letter to Mary
Phat restaurant
Prepare the interview
Interview
Find job

表 5 使用者的候選 App 列表

App Name
LinkedIn Job Search
7 Minute Workout
Birthdays for Android
Birthday Reminder
Interview Question and Answers
Birthday Wishes
HR Interview Preparation Guide
Total Fitness - Gym Workouts
Swimming & Triathlon Trainer
WhatsApp Messenger
GYM Trainer
AndroMoney (Expense Track)
Swim.com
State Bank Anywhere
Restaurant Finder
⋮

表 6　使用者的 App 推薦列表

	App Name
Top1	LinkedIn Job Search
Top2	Birthdays for Android
Top3	Birthday Reminder
Top4	Interview Question and Answers
Top5	Swimming & Triathlon Trainer
Top6	Restaurant Finder
Top7	AndroMoney (Expense Track)
Top8	7 Minute Workout
Top9	Job Search
Top10	HR Interview Preparation Guide

表 7　使用者的類別過濾器

Level-1	Level-2	
business		
life	sport	note
entertainment	food	
utility	tool	

表 8　使用者經過濾後的 App 推薦列表

	App Name in "business"
Top1	LinkedIn Job Search
Top2	Interview Question and Answers
Top3	Job Search
Top4	HR Interview Preparation Guide
Top5	Freelancer - Hire & Find Jobs

(2)　實驗評估

為了證實 SRMA 可有效的提供符合使用者需求的 App 推薦列表，我們設計了此實驗來作驗證。在實驗中，我們邀請 8 位普通的使用者，這些使用者中有不同年級的大學生，各自皆有不同的生活背景，藉由提供他們的行事曆資料，並透過 SRMA 產生基於他們各自的行事曆之前 10 名 App 推薦列表，並請他們為這 10 支 App 分別評 1~5 分。

該 SRMA 的環境設定為桌上型電腦：Intel i7-950 3.07GHz with 8G RAM, 500G hard disk, and Windows 7 (64bit)。實驗測量的準確度方法如下列公式 (2)所示：

$$Accuracy(Q_i) = \frac{|\mathrm{Re}\,l(Q_i) \cap \mathrm{Re}\,c(Q_i)|}{|\mathrm{Re}\,c(Q_i)|} \qquad (2)$$

其中 $Rel(Q_i)$為與使用者 i 相關的 App 集合；相關 App 的定義為使用者評分為 3 以上的 App；$Rec(Q_i)$為系統推薦給使用者 i 的 App 集合。

另外，利用 SRMA 算出在 8 位使用者行事曆中 TF-IDF 值最高的前 5 個字詞，若有同分的字詞則詢問該使用者要挑選哪一個字詞作為前 5 字詞，利用各字詞直接至 Google Play 上搜尋其前 10 名的 App，湊成最多 50 支 App 的推薦列表，並請使用者對這些 App 評分，作為 SRMA 推薦列表的實驗對照組，試說明若使用者直接至 Google Play 下關鍵字尋找 App 的準確度。

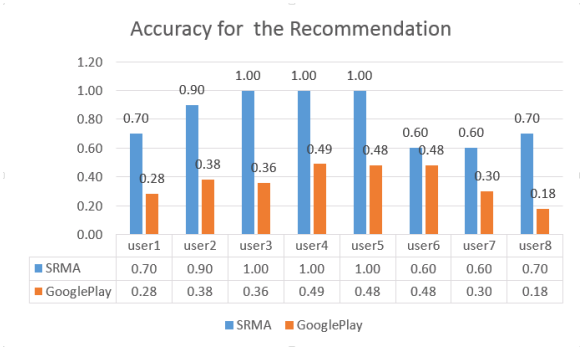


圖 4　8 位使用者的 SRMA 推薦準確率

實驗結果如圖 4 所示，透過 SRMA 的推薦，所有使用者的推薦準確率皆落在 60%至 100%中，其平均準確率為 81.25%；然而透過 Google Play 推薦的準確度為 18%至 49%，其平均準確度僅有 36.86%，此結果顯示：1)使用者在利用其認為不錯的關鍵字之情況下不容易在 Google Play 上搜尋到令其滿意的 App 的機率並不高；2)從平均準確率說明 SRMA 的普遍推薦 App 皆能符合使用者的預期；3)SRMA 仍有改善的空間，可加入其他資訊來提高準確率的穩定性。

五、結論

本研究提出了一個方法，稱為 SRMA (Semantic Recommendation for Mobile Apps)，來實現以情境感知之 App 推薦系統。SRMA 的主要特色包含：

- 提供一套以語意相似度計算方法，可度量使用者的行事曆與 App 的相似度
- 提供一套利用語意相似度方法的 App 分類機制，可將相似的 App 分類到相同的類別
- 提供一套基於行事曆的推薦方法，可找出與使用者的行事曆中記錄的日常生活需求相關聯之 App。

此外，實驗結果證實 SRMA 方法可實際推薦出符合使用者預期之 App，其推薦準確率高於 80%。

參考文獻

[1] S.-P. Ma, J.-S. Jiang, and W.-T. Lee, "Service Brick Composition Framework for Smartphones," in *20th Asia-Pacific Software Engineering Conference (APSEC 2013)*, 2013, pp. 459-466.

[2] S.-P. Ma, S.-J. Lee, W.-T. Lee, J.-H. Lin, and J.-H. Lin, "Mobile Application Search: A QoS-Aware and Tag-Based Approach," *EAI Endorsed Transactions on Industrial*

Networks and Intelligent Systems, vol. 15, p. e6, 2015.

[3] S. Hyrynsalmi, T. Mäkilä, A. Järvi, A. Suominen, M. Seppänen, and T. Knuutila, "App store, marketplace, play! an analysis of multi-homing in mobile software ecosystems," *Jansen, Slinger*, pp. 59-72, 2012.

[4] Statista. (2015). *Number of apps available in leading app stores as of July 2015*. Available: http://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/

[5] G. Adomavicius and D. Jannach, "Preface to the special issue on context-aware recommender systems," *User Modeling and User-Adapted Interaction*, vol. 24, p. 1, 2014.

[6] G. Adomavicius and A. Tuzhilin, "Context-aware recommender systems," in *Recommender systems handbook*, ed: Springer, 2011, pp. 217-253.

[7] A. Datta, K. Dutta, S. Kajanan, and N. Pervin, "Mobilewalla: A Mobile Application Search Engine," in *Mobile Computing, Applications, and Services*. vol. 95, J. Zhang, J. Wilkiewicz, and A. Nahapetian, Eds., ed: Springer Berlin Heidelberg, 2012, pp. 172-187.

[8] A. Datta, S. Kajanan, and N. Pervin, "A Mobile App Search Engine," *Mobile Networks and Applications*, vol. 18, pp. 42-59, 2013.

[9] *Appolicious*. Available: http://www.appolicious.com/

[10] A. Datta, K. Dutta, S. Kajanan, and N. Pervin, "Mobilewalla: A mobile application search engine," in *Mobile Computing, Applications, and Services*, ed: Springer, 2011, pp. 172-187.

[11] C. Yang, T. Wang, G. Yin, H. Wang, M. Wu, and M. Xiao, "Personalized mobile application discovery," in *Proceedings of the 1st International Workshop on Crowd-based Software Development Methods and Technologies*, 2014, pp. 49-54.

[12] D. Lemire and A. Maclachlan, "Slope One Predictors for Online Rating-Based Collaborative Filtering," in *SDM*, 2005, pp. 1-5.

[13] P. Yin, P. Luo, W.-C. Lee, and M. Wang, "App recommendation: a contest between satisfaction and temptation," presented at the Proceedings of the sixth ACM international conference on Web search and data mining, Rome, Italy, 2013.

[14] J. Hedley, "jsoup: Java html parser," ed, 2010.

[15] W.-Y. Ma and K.-J. Chen, "Introduction to CKIP Chinese word segmentation system for the first international Chinese Word Segmentation Bakeoff," in *Proceedings of the second SIGHAN workshop on Chinese language processing-Volume 17*, 2003, pp. 168-171.

[16] *Microsoft-Terminology-API*. Available: https://www.microsoft.com/Language/zh-tw/Microsoft-Terminology-API.aspx

[17] G. A. Miller, "WordNet: a lexical database for English," *Communications of the ACM*, vol. 38, pp. 39-41, 1995.

[18] R. Baeza-Yates and B. Ribeiro-Neto, *Modern information retrieval* vol. 463: ACM press New York, 1999.

[19] D. Crockford, "Introducing json," *Available: json. org*, 2009.

[20] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to information retrieval* vol. 1: Cambridge university press Cambridge, 2008.

[21] H. Zhu, E. Chen, H. Xiong, H. Cao, and J. Tian, "Mobile app classification with enriched contextual information," *Mobile Computing, IEEE Transactions on*, vol. 13, pp. 1550-1563, 2014.

[22] Z. Wu and M. Palmer, "Verbs semantics and lexical selection," in *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, 1994, pp. 133-138.