國立政治大學資訊科學系

Department of Computer Science
National Chengchi University

碩士論文

Master's Thesis

一個使用雙分群演算法進行智慧型手機應用程式推薦之框架

A Framework for Using Co-Clustering Algorithms to Recommend Smartphone Apps

研究生： 葉思妤 Szu-Yu Yeh
指導教授： 徐國偉 Kuo-Wei Hsu

中華民國一百零三年七月
July 2014

一個使用雙分群演算法進行智慧型手機應用程式推薦之框架

A Framework for Using Co-Clustering Algorithms to Recommend Smartphone Apps

研 究 生：葉思妤　　Student：Szu-Yu Yeh
指導教授：徐國偉　　Advisor：Kuo-Wei Hsu

國立政治大學

資訊科學系

碩士論文

A Thesis
submitted to Department of Computer Science
National Chengchi University
in partial fulfillment of the Requirements
for the degree of
Master
in
Computer Science

中華民國一百零三年七月
July 2014

摘要

近年來，智慧型手機(Smartphone)的銷量超過其他型式手機。智慧型手機具有更先進、更開放的行動作業系統，可允許使用者自行安裝應用程式軟體(Application)來擴充手機功能。目前市面上的應用程式數量非常龐大，在眾多的應用程式和有限的時間下，使用者不太可能將所有的應用程式下載試用，所以對使用者而言，找出自己所想要和需要的應用程式，是個困難的問題。推薦系統可依照使用者的喜好，或是準備推薦項目的相似程度來做推薦，讓使用者能較快得到想要的資訊，目前主要的方式有協同過濾(Collaborative Filtering, CF)、內容過濾(Content-Based Filtering, CBF)，還有結合前述兩種方式的混和式推薦(Hybrid Approach)。

本研究所使用的資料集是由政治大學資訊科學系所開發的實驗平台蒐集而來。資料以側錄的方式，將使用者實際操作手機應用程式的狀況記錄下來，其中包含了 25 位使用者和 1125 個應用程式。我們將原始資料集以三種方式整理成三個資料集：一、是否使用應用程式；二、使用應用程式的次數；三、使用應用程式的頻率，其值表示使用者在該應用程式的使用狀況。我們並將資料分成前段與後段時間兩部分，以前段時間的資料當作基準，推薦最多同群使用者使用的應用程式、同群使用者使用次數最多的應用程式，以及同群使用者最常使用的應用程式，然後以後段時間的資料做驗證，計算推薦結果的準確率與召回率加以比較。

我們使用知名的 Information Theoretic Co-Clustering Algorithm 和兩種基於 Minimum Squared Residue Co-Clustering Algorithm 的演算法將使用者與應用程式分群，利用分群結果做計算，推薦應用程式給使用者。實驗發

現三種演算法在第一個資料集的準確率與召回率表現最好，此資料集以 0 和 1 的值，來紀錄使用者在各應用程式的使用狀況。實驗比較三個演算法的結果，在大部分的情況之下，一個基於 Minimum Squared Residue Co-Clustering Algorithm 的演算法，給出的結果較好。

此外，我們也發現應用程式開發者將應用程式上架提供下載時，以個人主觀想法對該應用程式定義其分類，與我們利用雙分群方法，以使用者實際操作的情況將應用程式分類的結果有些差異，或許在 Google Play 的分類上可做調整。

本研究提出推薦系統的框架具有彈性，未來可以使用不同的雙分群演算法做分群，也能套用其他的推薦方式。

關鍵字：雙分群、智慧型手機應用程式、推薦系統

國立政治大學
National Chengchi University

Abstract

With the rapid evolution of smartphone devices, tens of thousands applications have been supplied on online stores such as App Store (operated by Apple Inc.) and Google Play (operated by Google Inc.). Since there are many applications, recommending applications to users becomes an important topic. In this thesis, we present a framework for using a co-clustering algorithm to recommend applications to users. Recommendations are a part of everyday life. People usually rely on some external knowledge to make informed decisions about a particular artifact or action. Using recommender systems is one of general approaches that help people make decisions. There are three common types of recommender systems, namely collaborative filtering, content-based filtering, and hybrid recommender systems.

In this thesis, we use the dataset that was collected by a tool developed by the Department of Computer Science at the National Chengchi University. It recorded the users' behavior when they were using their smartphones. We transform the original dataset into three types of datasets: 1) indicating whether a user used an application; 2) indicating the number of uses made by a user for an application; 3) indicating the frequency of uses made by a user for an application. Furthermore, we divide each dataset into two parts: The first part containing data for the early time period is used as the recommending base, and the second part containing data for the late time period is used for verifying the results. We utilize three famous co-clustering algorithms, which are the Information Theoretic Co-Clustering Algorithm and two algorithms based on

國立政治大學 National Chengchi University

the Minimum Squared Residue Co-Clustering Algorithm, in the proposed framework.

According to the clusters given by a co-clustering algorithm, we recommend top five applications to each user by referring to the maximum number of users, the maximum number of uses, and the most frequently used applications that are in the same cluster. We calculate the precision and recall values to compare the results. From the experimental results, we find that the best result corresponds to the first type of dataset and also that one of the algorithms based on the Minimum Squared Residue Co-Clustering Algorithm is better than the other two algorithms in terms of the precision and recall values.

From the clusters of applications, we obtain some interesting insights into the categories of applications. The categories of applications are set by their developers, but the users may not totally agree with the settings. There might be space for improvement for the categories of applications on the online store.

In the future, we can utilize different co-clustering algorithms and other recommended methods in the proposed framework.

Keyword: Co-clustering、Smartphone、Mobile Application、Recommender System

國立政治大學 National Chengchi University

致謝

本論文可以順利完成，首先要感謝我的家人，讓我在研究所學習的這段期間，不用擔心其他事務，能放心的將學業完成，親愛的爸爸、媽媽總是相信我的決定，給我很大的支持與幫助。

感謝我的好朋友，辰輝、佳妤、曉吟、嬴爵、鈺婷，在我壓力大時總是傾聽我的想法，適時的提供建議，並和我一起討論和解決問題。感謝趨勢科技的同事們，總是默默的關心著我，幫忙我工作上的困難，特別是在怡、語婷，常陪我聊天。感謝和平高中的好朋友淇秝、嘉玲，一直以來陪伴著我，聽我訴苦。

最要感謝的是我的指導教授徐國偉老師，可以成為老師的學生真的非常幸運，每次在討論論文時，給予我許多寶貴的意見與信心，也非常有耐心的給我很多指導以及鼓勵，讓我能夠按照進度至最後完成論文。

最後，感謝所有關心、幫助我的人。

葉思妤 謹誌

國立政治大學

資訊科學學系研究所

民國 103 年 7 月

國立政治大學 National Chengchi University

目錄

第一章 緒論 1

1.1 前言 1
1.2 研究動機 1
1.3 研究目的 4
1.4 論文架構 5

第二章 文獻探討 6

2.1 推薦系統 6
2.1.1 協同過濾 6
2.1.2 內容過濾 7
2.1.3 混合式推薦系統 7
2.2 CO-CLUSTERING (雙分群) 8
2.2.1 Information Theoretic Co-Clustering Algorithm 9
2.2.2 Minimum Squared Residue Co-Clustering Algorithm 9
2.3 如何使用 CO-CLUSTERING 做推薦系統 10
2.4 總結 10
2.4.1 電影 11
2.4.2 景點/餐廳 13
2.4.3 電子商務 14
2.4.4 文本資料 15
2.4.5 健保 16
2.4.6 社群網路服務 16
2.4.7 手機應用程式 16

第三章 基於雙分群的推薦框架 18

3.1 總覽 18
3.2 實際運作範例 20
3.2.1 以使用者是否使用應用程式做推薦 20
3.2.2 以使用應用程式之次數做推薦 22
3.2.3 以使用者使用應用程式的頻率做推薦 24

第四章 實驗 27

4.1 環境與流程 27
4.1.1 實驗環境 27
4.1.2 實驗流程 27
4.2 資料集 34
4.3 實驗結果(一) 39

國立政治大學 National Chengchi University

4.3.1 以使用者是否使用應用程式做推薦(YN) 41
4.3.2 以使用者使用應用程式的次數做推薦(Count) 42
4.3.3 以使用者使用應用程式的頻率做推薦(Frequency) 45
4.3.4 小結 48
4.4 實驗結果(二) 55
4.4.1 以使用者是否使用應用程式做應用程式分群(YN) 56
4.4.2 以使用者使用應用程式的次數做應用程式分群(Count) 58
4.4.3 以使用者使用應用程式的頻率做應用程式分群(Frequency) 60
4.4.4 小結 62
第五章 結論與未來可能研究方向 63
5.1 結論 63
5.2 未來可能研究方向 64
參考文獻 65

國立政治大學
National Chengchi University

表目錄

表 1 推薦系統文獻整理.....11

表 2 本實驗使用之參數說明.....33

表 3 使用者操作行為紀錄(範例).....34

表 4 手機應用程式分類說明(範例).....35

表 5 實驗使用之資料集及其結果說明.....37

表 6 稀疏矩陣檔案說明.....39

表 7 Confusion Matrix 說明.....39

表 8 使用者只在後段時間所使用的應用程式個數.....40

表 9 實驗 4.3.1，以使用者是否使用應用程式資料做推薦，準確率、召回率及 F-measure 比較表.....41

表 10 實驗 4.3.1，平均準確率及召回率.....42

表 11 實驗 4.3.2，以使用者使用應用程式的次數資料，推薦使用人數最多的 App，準確率、召回率及 F-measure 比較表.....43

表 12 實驗 4.3.2，推薦使用人數最多，App 的平均準確率及召回率.....43

表 13 實驗 4.3.2，以使用者使用應用程式的次數資料，推薦使用次數最高的 App，準確率、召回率及 F-measure 比較表.....44

表 14 實驗 4.3.2，推薦使用次數最多，App 的平均準確率及召回率.....45

表 15 實驗 4.3.3，以使用者使用應用程式的頻率資料，推薦使用人數最多的 App，準確率、召回率及 F-measure 比較表.....46

表 16 實驗 4.3.3，推薦使用人數最多的 App，平均準確率及召回率.....46

表 17 實驗 4.3.3，以使用者使用應用程式的頻率資料，推薦使用頻率最高的 App，準確率、召回率及 F-measure 比較表.....47

表 18 實驗 4.3.3，推薦使用頻率最高的 App，平均準確率及召回率.....48

表 19 以最多人使用的應用程式排名，使用是否使用應用程式資料集中做推薦.....49

表 20 以最多使用次數的應用程式排名，以使用應用程式次數資料集做推薦.....50

國立政治大學 National Chengchi University

表 21 以使用頻率最高的應用程式排名，以使用應用程式頻率資料集做推薦.....51
表 22 以最多人使用的應用程式排名，使用是否使用應用程式資料集中做推薦.....52
表 23 以最多人使用的應用程式排名，以使用應用程式次數資料集做推薦.....52
表 24 以最多使用次數的應用程式排名，以使用應用程式次數資料集做推薦.....53
表 25 以最多人使用的應用程式排名，以使用應用程式頻率資料集做推薦.....53
表 26 以使用頻率最高的應用程式排名，以使用應用程式頻率資料集做推薦.....54
表 27 準確率平均值比較表.....55
表 28 應用程式群集數量.....56
表 29 Sudoku 相關應用程式資料.....56
表 30 旅遊及交通相關應用程式.....56
表 31 各群集中，應用程式數量.....57
表 32 各演算法中，四個 Sudoku 相關 App 的分類情形.....57
表 33 各演算法中，五個旅遊與交通相關 App 的分類情形.....58
表 34 各群集中，應用程式數量.....59
表 35 各演算法中，四個 Sudoku 相關 App 的分類情形.....59
表 36 各演算法中，五個旅遊與交通相關 App 的分類情形.....60
表 37 各群集中，應用程式數量.....61
表 38 各演算法中，四個 Sudoku 相關 App 的分類情形.....61
表 39 各演算法中，五個旅遊與交通相關 App 的分類情形.....62

國立政治大學 National Chengchi University

圖目錄

圖 1 本研究推薦流程 18

圖 2 實際運作範例中應用程式被使用者使用情形 21

圖 3 範例一，使用 Information Theoretic Co-Clustering Algorithm 21

圖 4 範例一，使用 Minimum Squared Residue I Co-Clustering Algorithm 21

圖 5 範例一，使用 Minimum Squared Residue II Co-Clustering Algorithm 22

圖 6 使用者使用應用程式的次數示意圖 23

圖 7 範例二，使用 Information Theoretic Co-Clustering Algorithm 23

圖 8 範例二，使用 Minimum Squared Residue I Co-Clustering Algorithm 23

圖 9 範例二，使用 Minimum Squared Residue II Co-Clustering Algorithm 24

圖 10 使用者使用應用程式的頻率示意圖 25

圖 11 範例三，使用 Information Theoretic Co-Clustering Algorithm 25

圖 12 範例三，使用 Minimum Squared Residue I Co-Clustering Algorithm 25

圖 13 範例三，使用 Minimum Squared Residue II Co-Clustering Algorithm 26

圖 18 實驗流程圖 29

圖 19 使用者使用應用程式個數 35

圖 20 應用程式在各分類的分布 36

圖 21 各使用者使用應用程式的總次數 38

國立政治大學 National Chengchi University

第一章 緒論

1.1 前言

近年來，智慧型手機(Smartphone)的銷量已經超過了市場上的其他手機。智慧型手機具有獨立的行動作業系統，可透過安裝應用軟體(Application，簡稱 App)、遊戲等程式來擴充手機功能。目前市面上的應用程式數量非常龐大，使用者可能在不清楚應用程式的情況下，下載了不適用的應用軟體，往往浪費許多時間和金錢。在此，我們使用雙分群(Co-Clustering)方法來建置推薦系統(Recommender System)，讓使用者可以在較短的時間內，找到自己所需的應用程式。

1.2 研究動機

全球的科技持續且迅速的發展，人們在日常生活中已經跟手機有著密不可分的關係。伴隨著智慧型手機科技的發展，手機的功能也越來越多，其中最熱門的討論話題，莫過於手機上使用的應用程式軟體。聯合新聞網報導，根據 Google 調查，台灣智慧型手機使用者平均安裝 30 個應用程式，超過美國和中國，顯示民眾對行動裝置相當依賴。當 Apple 的 App Store 與 Google 的 Play Market(之前稱為 Android Market)陸續登場，我們感受到了 App 的狂潮。在短短四年內，在這兩個最主要市集上累積的 App 數量，已經跨越了 100 萬的規模門檻[1]。

Android 是一個以 Linux 為基礎的半開放原始碼作業系統，主要用於行動設備，由 Google 成立的 Open Handset Alliance 持續領導與開發中。Android 系統由安迪·魯賓(Andy Rubin)開發製作，最初主要支援手機，於 2005 年 8 月被 Google 收購。2007 年 11 月，

[1] 資料來源：http://mag.udn.com/mag/digital/storypage.jsp?f_ART_ID=422828

Google 與多家硬體製造商、軟體開發商及電信營運商成立開放手持設備聯盟來共同研發改良 Android 系統，隨後，Google 以 Apache 免費開源許可證的授權方式，發佈了 Android 的原始碼，讓生產商推出搭載 Android 的智慧型手機，而 Android 作業系統後來更逐漸拓展到平板電腦及其他領域上。目前使用 Android 系統的手機的大廠，依字母順序排列，包含 Acer、Asus、Google、HTC、Samsung、Sony 等；平板電腦則有 Acer、Asus、Cisco、Dell、HTC、Samsung、Sony 等[2]。Google Play 是由 Google 為 Android 所開發的應用程式發行平台，於 2008 年 8 月 28 日公開發表並於 2008 年 10 月 22 日正式上線，截止目前，應用程式中的付費服務共支援過一百個國家[3]，Google Play 上頭有超過 80 萬個 App，而總下載次數已超越 250 億次之多[4]。

App Store，是蘋果公司為其 iPhone、iPod Touch 以及 iPad 等產品創建和維護的數位應用發佈平台，允許使用者從 iTunes Store 瀏覽和下載一些由 iOS SDK 或者 Mac SDK 開發的應用程式。根據應用程式發佈的情況，使用者可以付費或者免費將應用程式直接下載到 iOS 設備，也可以透過 Mac OS X 或者 Windows 平台下的 iTunes 下載到電腦中。蘋果的 App Store 於 2008 年 7 月份上線，第三方開發者將 App 提供給蘋果公司審核，發佈後供使用者下載。截至 2012 年底，App Store 服務於全球 155 個國家，總體 App 數量超過 80 萬個，適於 iPad 的 App 30 萬個，下載次數突破 400 億次5。2013 年，App 數量更是超過 90 萬個，並且下載次數超過 500 億次[6]。

[2]資料來源：http://zh.wikipedia.org/wiki/Android

[3]資料來源： http://zh.wikipedia.org/wiki/Google_Play

[4]資料來源：http://www.bnext.com.tw/article/view/cid/0/id/26012

[5]資料來源：http://technews.tw/2013/01/08/ios/

[6]資料來源： http://en.wikipedia.org/wiki/App_Store_(iOS)

2010 年 10 月 11 日，微軟(Microsoft)正式發佈了智慧型手機作業系統 Windows Phone。2011 年 2 月，諾基亞(Nokia)與微軟合作研發。目前使用 Windows Phone 系統的行動裝置，除了 Windows Phone，還有 Acer、Nokia 等品牌。Windows Phone Store(前身為 Windows Phone Marketplace)是由微軟提供的一個針對 Windows Phone 平台的服務，允許使用者能夠下載由第三方開發的應用軟體。Windows Phone Store 與 Windows Phone 於 2010 年 10 月上線。截至 2012 年 12 月 31 日，Windows Phone Store 擁有超過 15 萬項應用程式可供使用者使用[7]。

BlackBerry World 是一個由 BlackBerry 公司為 BlackBerry OS 創建的服務，於 2009 年 4 月 1 日創立，允許使用者下載一些由第三方開發商開發的應用程式。BlackBerry World 於 2009 年 4 月上線，目前有超過 10 萬個應用程式可供使用者使用[8]。於 2013 年 1 月底發佈的 BlackBerry 10 手機系統平台是開放的，支援以不同方式寫 App 。BlackBerry 10 的開發工具當中，有一個特別的功能叫 Android Runtime，可以直接把 Android 上的 .apk 轉為 BlackBerry 10 上使用的 .bar 檔，簡單來說就是把整個 APP 完全移植到 BlackBerry 10 平台上[9]。此功能的開發或許可以促使 BlackBerry 的應用程式數量更快速地增加。

目前應用程式軟體，除了可以在智慧型手機上應用之外，在智慧電視(Smart TV)上也可以下載使用。以往對於大多數使用者在使用電視時，最常使用的功能一定是觀賞電視節目，或是接上影音周邊設備來觀看 DVD 或是藍光光碟。在網路普及化的時代來臨之後，智慧電視也增添了許多智慧連網的功能。智慧電視可以執行完整的作業系統，並提供一個軟體平台，供應用軟體開發者，開發他們自己的軟體，並在智慧電視之上執行，讓使用者除了使用電視原有的功能之外，又多了上網瀏覽網頁、下載應用程式、連結臉書社群網站

[7]資料來源：http://zh.wikipedia.org/wiki/Windows_Phone_Store

[8]資料來源：http://en.wikipedia.org/wiki/BlackBerry_World

[9]資料來源： http://www.eprice.com.hk/mobile/talk/3475/49038/

等附加功能。各 Smart TV 也有相對應的 App 可以下載使用，例如 Samsung 有特別針對電視設計的應用程式。

應用程式的數量持續不斷的增加，使用者的選擇越來越多，如何在有限的時間和預算中，找到最適合自己的 App，成了值得關注的課題。

1.3 研究目的

目前推薦 App 的網站有許多，像是德國的 AndroidPIT 網站(www.androidpit.com)，就是一個推薦使用者 Android 作業系統 App 的網站。使用者可以在這裡尋找適合的應用程式，應用程式開發者也可以在此發佈自己的 App。當使用者尚未輸入任何資訊時，網站會依照目前最熱門的 App 做推薦，也可以針對使用者自訂的條件，例如評分(Ratings)、時間(Time)、下載次數(Installations)等篩選條件做推薦。

台灣 APPGuru 網站(www.Appguru.com.tw)的目標，是讓每個人都能找到適合自己用的 App，此網站以收集完整、實用的 App 情報為宗旨。推薦方式以作業系統區分為 Android 及 iOS 兩種，並將 App 做分類，分成工具、遊戲、旅遊、攝影等項目，方便使用者尋找。

在眾多的 App 推薦網站下，目前沒有發現依據使用者的喜好及使用習慣做為推薦的依據。因此，我們想利用使用者平時使用 App 的習慣，推薦使用者想要的 App。我們利用 Co-Clustering 可以同時對資料矩陣的行與列作分群的特性，提出一個彈性的雙分群框架，對不同的資料集來做分群，並且以分群的結果對使用者做推薦。我們探討不同 Co-Clustering 演算法在各個不同設定的資料集的推薦結果，計算準確率、召回率以及 F 度量作分析及比較。本研究所使用的資料，並非來自傳統的問卷調查，而是由側錄方式直接取得的資料，所以，可以確保使用者一定有下載並且使用有出現在資料集內的 App。收集資料的對象是台灣大學校園內使用智慧型手機的族群，而他們所使用的是 Android 作業系統的手機。

1.4 論文架構

本論文分為五個部分，第一章為本篇論文的緒論，包含研究動機、研究目的及論文架構。在第二章為文獻探討，內容涵蓋本研究相關技術，包含推薦系統與技術、Co-Clustering，及其相關應用。第三章為本研究之技術探討。實驗結果整理於第四章，用以驗證本論文中所提出的作法的可行性，而第四章也包含針對實驗結果的討論。第五章對本研究做一個結論，並探討未來可能的研究方向。

國立政治大學
National Chengchi University

第二章 文獻探討

2.1 推薦系統

在現實生活中每個人對每件事會有不同的觀點，這樣的觀點有時很抽象。使用者無法使用一串關鍵字來正確表達它，推薦系統(Recommender System)就是一個針對個人化問題發展出來的智慧型系統[3]，它能幫助我們快速找尋到我們有興趣的資料或商品，若消費者覺得購買的商品很符合自己的習性，以後便會增加對系統的信賴度，更常使用系統來消費，而系統能準確預測出消費者對於購買商品的偏好，便能促進電子商品的交易買賣還有增加公司的銷售量。可以達到使用者和系統雙贏的目標。對於資訊過量的問題，也有研究[3]指出可以用推薦技術能解決。

推薦系統目前主要的方式有幾類，其中以協同過濾(Collaborative Filtering，簡稱為 CF)和內容過濾(Content-Based Filtering，簡稱為 CBF)為主，前者是利用前人的經驗所給予的推薦，而後者是針對個人對某些商品的特徵點計算出這個人對各特徵點的重視程度，進而做出預測。另外也有混合這兩者的混合式推薦系統(Hybrid Recommender Systems)。

2.1.1 協同過濾

協同過濾(Collaborative Filtering)推薦系統，是利用有共同經驗之群體的喜好，來推薦使用者可能會感興趣的資訊，而每個使用者透過合作機制，給予所推薦的資訊相當程度的回應(如評分)，進而幫助別人篩選資訊。協同過濾的原理來自於使用者會喜歡參考有相同品味或想法的人的建議，透過找出與某使用者行為相似的一群人，然後用這群人過去的記錄，預測該使用者對於新商品(推薦項目)的喜好程度，進而推薦喜好程度最高的商品給該使用者。以 Amazon 為例，該公司掌握了客戶過去的歷史購買資訊，累積資訊的結果可以發現使用者購買商品之相關性，這樣的相關性就成為他推薦的主要依據[4]。

國立政治大學 National Chengchi University

協同過濾又可分為評比(Rating)或者群體過濾(Social Filtering)。其後成為電子商務當中很重要的一環，即根據某顧客以往的購買行為以及從具有相似購買行為的顧客群的購買行為去推薦這個顧客可能喜歡的品項，也就是藉由社群的喜好提供個人化的資訊、商品等的推薦服務[10]。然而協同式推薦系統還是有他的侷限性在，例如像是稀疏性(Sparsity)的問題，當商品數量超出使用者，兩者的比例十分懸殊時，系統就無法找到適合的推薦者來進行推薦。另外可能有擴充性(Scalability)的問題，當使用者及物品數量越來越大時，可能會影響到運算速度。

2.1.2 內容過濾

內容過濾(Content-Based Filtering)推薦系統，依據的基礎是對物品內容的分析，而不是使用者所給予的評價。內容過濾推薦系統會預測使用者對內容的喜愛程度，再找出使用者可能會感興趣的特徵，進而找出使用者喜歡的物品。因為內容過濾推薦系統是針對物品內容去做分析，進而推薦使用者，而它的缺點，主要是較難處理聲音、圖片、藝術品、影像等不容易被找出特徵的內容。另外，由於內容過濾僅對內容做處理，使用者只能看到與過去經歷類似的推薦項目，這樣可能會失去許多的潛在推薦。

2.1.3 混合式推薦系統

混合式推薦系統(Hybrid Recommender Systems)是使用兩種以上推薦系統，它可以針對問題的特性，結合不同推薦系統的優點，以彌補各推薦系統在特定問題的不足。

[10] 資料來源：http://en.wikipedia.org/wiki/Collaborative_filtering

國立政治大學 National Chengchi University

2.2 Co-Clustering (雙分群)

分群(Clustering)指的是按照資料之間的差異性，將資料分組。一筆資料只屬於某一組，相似資料歸類於同一組，每一組稱作一個群集(Cluster)。Co-Clustering，又稱為 Biclustering 或 Two-Mode Clustering[11]。Co-Clustering 是一種資料探勘的技術，它將資料儲存在矩陣的行和列中，並且進行排序。目前有數種 Co-Clustering 演算法，像是 Block Clustering、The Plaid Model、Coupled Two-Way Clustering、 Interrelated Two-Way Clustering[12]、Information Theoretic Co-Clustering Algorithm[7]、Minimum Squared Residue Co-Clustering Algorithm[8]。

一般資料探勘的過程中，通常在同一時間下，只會考慮資料矩陣的行或列其中之一，Co-Clustering 則是將資料矩陣的行與列同時考慮，但取決於所使用的特定演算法，可能會產生不同的結果。

最早將 Co-Clustering 公式化的研究，是由 Hartigan 提出的，在[4]中稱為直接群集(Direct Clustering)。Hartigan 提出多種 Co-Clustering 的方法和模型，並在階層化的 Co-Clustering 模型中，使用一個貪婪算法(Greedy Algorithm)，使其中的行和列分區可以以分層的方式，用樹(Tree)加以說明。分群問題的研究如[5]，該文研究文件與文字分群的問題，提出了 Spectral Algorithm，使用 Co-Clustering 技術，同時考慮了文件和文字，將文件和文字分群。研究[aa55]提出基因表達資料的雙分群方法，他們使用一個考慮平方差的方法去衡量分群的品質。[7]提出了一種基於資訊理論的雙分群演算法，該研究利用遞減函數將兩個互相影響的行與列同時做文字的分群，並使用資訊理論解決受限於行和列群集數目的問題，[7]的實驗結果顯示該演算法在稀疏與高維的資料集中有良好的運作。

[11]資料來源：http://en.wikipedia.org/wiki/Biclustering

[12] 資料來源：http://www.wisegeek.com/what-is-biclustering.htm

大多數現有的推薦系統中使用任何內容或基於協同過濾的方法，而研究[9]考慮到使用者和推薦項目的個人偏見，提出一種新的協同過濾方法來做推薦。

以下介紹本研究所使用的 Co-Clustering 演算法。

2.2.1 Information Theoretic Co-Clustering Algorithm

Information Theoretic 是應用數學、電氣工程、電腦科學中涉及訊息進行量化的一個分支，由 Claude E. Shannon 開發，目前應用在許多的領域，包含統計、自然語言處理、密碼學、神經生物學等。Information Theoretic Co-Clustering Algorithm 由 Inderjit S. Dhillon 等人提出，[7]提出一個雙分群演算法，類似 K-Means 和 EM 的精神，使用康乃爾大學提供的新聞資料(NG20)，利用遞減函數將兩個互相影響的行與列同時做文字的分群，實驗證明了該演算法在稀疏與高維的資料集中有良好的運作，該演算法流程為：輸入矩陣資料集欲取得的行與列 Cluster 數量，將 Matrix 隨機切區塊，當作 Row Cluster 的雛型，使用 K-L 距離計算並分配 Row 為最近 Cluster，利用分配過的 Row 資料，將 Matrix 隨機切區塊，計算 Column Cluster 的雛形，再使用 K-L 距離計算並分配 Column 為最近 Cluster，之後重複計算產生新的 Row 和 Column 的 Cluster，直到收斂為止，最後輸出行與列的分群。

國立政治大學 National Chengchi University

2.2.2 Minimum Squared Residue Co-Clustering Algorithm

Co-Clustering Software 所使用的 Minimum Squared Residue Co-Clustering Algorithm[8]主要是由 Hyuk Cho 等人提出的方法，研究中表明該雙分群演算法在基因分群的實驗是有效的。使用 Iterative Algorithms 解決剩餘最小矩陣的 NP-hard 問題。提出兩種計算方式來評估 Co-Cluster 的同質性，分別為一、計算群集中每個項目與其平均值的差值之平方和。二、計算群集中每個項目與其對應的行與列的差值之平方和，群集的平均值需保有對稱性。

本研究也使用了 Hyuk Cho 等人所提出的兩種計算方式進行實驗。使用該演算法需要先輸入矩陣資料集欲取得的行與列 Cluster 數量，使用 Batch Iterative Algorithms，利用所提的兩種方法，先計算 Column 的 Cluster，再計算 Row 的 Cluster，直到收斂為止，最後輸出行與列的分群。

2.3 如何使用 Co-Clustering 做推薦系統

現今，Co-Clustering 在演算法的開發及應用上受到關注，研究範圍相當廣，其中較廣為人知的，是在癌症基因上的研究。除了在醫學方面的研究外，Co-Clustering 也被使用在其他的研究，例如推薦系統。研究[9]就是以加權 Co-Clustering 演算法，在做推薦預測時，同時考慮到使用者和推薦項目的個人偏見，該研究並提出一個新的協同過濾推薦方法。而[10]則是結合了協同定位建議(Collaborative Location Recommendation)的框架做推薦。研究[a55]運用協同過濾推薦系統，同時結合了使用者喜好進行推薦演算法、廣告的受歡迎程度進行推薦演算法與考慮擴充矩陣的雙分群演算法，顯示出在缺乏已知資料的議題上能藉由同時應用多重矩陣之間的資訊而能夠產生更佳的結果。

國立政治大學 National Chengchi University

2.4 總結

總結上述討論，整理了表 1。如下，橫軸表示推薦系統的應用，縱軸則呈現其使用技術，依照類別在以下各小節做說明。

表 1 推薦系統文獻整理

Recommender Systems	電影	景點/餐廳	電子商務	文本資料	健保	社群網路服務	手機應用程式
協同過濾 Collaborative Filtering	[11] [12] [13] [14] [15] [16] [17] [18] [19]	[10] [26]	[9] [30] [31] [32] [33] [34]	[39]		[45] [46]	[48] [49] [50] [51]
內容過濾 Content-Based Filtering		[27]	[35] [36]		[43]	[47]	
混合式推薦 Hybrid	[20] [21] [23] [24] [19]	[28] [29]	[37] [38]	[40] [41] [42]			
其他 Misc	[25]				[44]		

2.4.1 電影

在電影相關研究中，有許多是以 MovieLens 或 NetFlix 的資料來做實驗。MovieLens(http://grouplens.org/datasets/movielens/)是由美國明尼蘇達大學電腦科學與工程學系的 GroupLens 研究小組，針對研究用途所提供之網站的電影評分資料。NetFlix 公司是一家網路串流媒體的供應商，在 2007 年舉辦了一個 Netflix Prize 的競賽(http://www.netflixprize.com/)，利用客戶的資料，希望能找出最好的影片推薦方法，2009 年九月，由 BellKor's Pragmatic Chaos 團隊拿下了首獎。然而，由於隱私權問題備受爭議，NetFlix 公司已在 2010 年取消這個比賽[13]。

以協同過濾方法做研究，使用 MovieLens 資料的文獻如[11]，對 Apache Mahout 實作，使用的 Mahout Distributed Item Base Recommendation System 進行分析，修改的該實作中花費較多執行時間的 Similarity Matrix 計算部分進行修改，實作出兩個 Distributed

[13]資料來源：http://en.wikipedia.org/wiki/Netflix

Collaborative-Based Recommendation Systems，經實驗證實該論文所實作的系統在 Performance 和 Accuracy 的表現都優於原系統。[12]提出一個新的方法於動態協同過濾法(Dynamic Similarity Collaborative Filtering，DSCF)上，衰減人與人之間的相似度，並比較預測值與實際值的結果，讓每個使用者在不同時間點，都有適合個人的相似度衰減值。實驗證明該方法更符合人們的行為，且在執行時間上，有很大的改進。[13]和[14]在群體推薦演算法中，分別以基因演算法及皮爾森係數(Pearson Correlation Coefficient)和餘弦相似性(Cosine Similarity)做實驗，發現群體推薦結果能作為群體決策時的一個參考，更能滿足一般使用者在從事群體活動時的需求，也證實該方法可以提高偏好預測時的準確度，且不會犧牲預測的涵蓋範圍。[15]提出整合分群與包含啟發式搜尋的特徵選取法，可根據資料分群結果動態地搜尋最佳特徵子集，用以改善分群結果。實驗使用基因演算法做計算，結果顯示該研究提出之特徵選取法確實能有效提高推薦系統之效能。[16]使用統計特徵距離(Statistical Attribute Distance，SAD)的 Multi-Tier Granule Mining Algorithm 方法，提出一個回饋推薦機制架構在協同過濾器下的推薦系統。[18]是在 CF 框架下，透過實驗，在不同的組合中發現，使用結合個人的資訊和社群的意見，能產生更好的推薦。[17]分析了不同的基於項目(Item-Based)的推薦演算法，透過實驗使用不同的技術(Basic Cosine/Adjusted Cosine/Correlation)來計算項目項相似之處並做評估，與基本的 K-Nearest Neighbor 方法比較，發現基於物品(Item-Based)的算法比基於用戶(User-Based)的算法，有更好的性能。[20]則是使用 Netflix 的資料，提出一個考量多種類型多樣性的混合型推薦系統，會預先算好所有類型項目的多樣性權重並將這些多樣性因素結合混合式推薦系統，達到各種多樣性類型的效果，實驗結果表示該系統可改善並考量到多樣性來滿足使用者的興趣。

混合式推薦系統中，以 MovieLens 的資料做實驗的研究，如[21]，使用一個混合式的推薦系統，來整合以使用者為基礎的協同過濾演算法，加上雲模型(Cloud Model)協同過濾演算法，來預測結果。實驗證實混合式的推薦系統可改善稀疏性的問題。在[23]當中，實驗使用 Bayesian Learning 和 Decision Tree Learning 方法，發現基於案例推理(Case-Based)能夠提供比由專家使用以規則基礎(Rule-Based)的預測更加準確，並且可以擴充系統。[24]

則是使用 Compaq Systems Research Center 提供的 EachMovie1 資料，採用基於內容的預測，使用 Neighborhood-Based Algorithm 以加強現有的使用者資料，然後透過協同過濾，使用 Bag-Of-Words Naive Bayesian Text Classifier，提供個性化建議。

[19]使用了 MovieLens 的資料，提出兩個方法，一為考慮個別商品內容差異性的協同過濾推薦技術，解決傳統協同過濾技術不考慮商品異質性的缺點，研究結果顯示，可以提高準確度，另一個則是結合傳統協同過濾技術的混合方法，其實驗顯示該方法可以達到與傳統協同過濾技術相同的可預測涵蓋範圍，且有較佳的推薦效果。

[25]提出了一種歸納學習方法(Inductive Learning)，即能夠同時使用評比的資訊和其他形式的資訊，預測用戶的偏好。

2.4.2 景點/餐廳

推薦系統文獻中，也有以地點相關做的研究。例如[26]使用協同過濾方法，採用公式計算使用者之間的興趣相似程度，配合各相對年齡層做不同權重值分數評價調整，以花蓮為主要推薦旅遊景點，並且嵌入 Google Map API 作為旅途路徑規劃，是一套景點查詢及規劃旅遊行程之整合型推薦系統。[10]提出了協同定位建議(Collaborative Location Recommendation(CLR))的框架做推薦。在推薦過程中考慮了活動(例如：喜好)和不同的使用者類別(例如：一般使用者或旅行中的使用者)，CLR 能夠產生更精確和完善的建議。此外，CLR 採用動態分群演算法 CADC，當新的 GPS 軌跡資料產生時，可以有效的將資料做分類，該研究也證明的 CLR 的方法較一般方法更能提出精確的推薦。

在內容過濾的基礎下，[27]使用 Facebook 建構了一個即時餐飲商店推薦系統，以機器學習法，在多為空間中以分群法，推薦適合的餐飲商店。

[28]提出一個以標籤為基礎之混合式推薦系統，資料採用於文化資產與歷史古蹟資料庫，建議適合使用者參訪的景點和觀覽路徑。[29]使用一種新的方法 EntreeC，結合了基於知識(Knowledge-Based)推薦和協同過濾(Collaborative Filtering)，使用 KDD 資料庫中的 Entrée 資料，做餐館的推薦。

2.4.3 電子商務

電子商務方面，使用協同過濾方法的研究如[30]，該研究以一著名的網路音樂 CD 販售網站 Joy Audio (http://www.joyaudio.com.tw/)為例，先將音樂進行屬性建置成模糊語意知識庫，收集買方每次的音樂採購偏好，開發出音樂推薦系統的雛形。研究使用歐幾里德距離(Euclidean Distance)公式計算離異度，結果顯示，使用模糊語意法進行產品推薦之動作，可提供使用者在音樂產品上做良好的選擇。[31]根據一個音樂推薦系統 RINGO(A Personalized Music Recommendation System)，利用其網站使用者資料，並使用四個演算法進行實驗，發現 The Constrained Pearson R Algorithm 表現最好。[32]使用自動產生評比的機制，利用使用者的網頁瀏覽紀錄，由實驗觀察，發現該研究所使用隱藏式評比(Implicit Rating)之協同過濾機制所得之結果較前人研究使用明顯是評比(Explicit Rating)的結果佳。[33]不同於傳統兌換商品的系統，將網頁伺服器及資料庫和網路作結合，並加入協同過濾推薦機制及個人化的技術，透過分析每位消費者察看及兌換商品的紀錄，找出其個人的偏好，並為其建立一個專屬的個人化推薦機制。[34]探討 Amazon 網站使用 Item-To-Item Collaborative Filtering 推薦的方法與技術。

使用內容過濾法的研究如[35]，該研究探索電子商務模型，透過問卷調查方式了解線上消費者的行為，運用資料探勘決策樹方法，去探索蒐集到的問卷資料，並將商品推薦分為「一般商品推薦」及「互補商品推薦」計算模型，計算消費者對商品的喜好，進而推薦商品給使用者。[36]探索基於內容過濾的書籍，透過應用自動化的文本分類方法，從網上

摘錄的半結構化文本，使用 LIBRA，提取網頁在 Amazon.com 的圖書資料庫，使用標題內容的 Meta-Data，而不是圖書本身的實際文本，使用 Bayesian Learning 法，產生的最值得推薦的結果。

混合式推薦的研究，則有[37]，該文提出了一個混合推薦系統，結合了協同過濾和 Knowledge-Based 的方法，說明可以結合各種知識庫來做推薦系統。[38]則是結合協同過濾和基於內容的方法，提出一種新的混合推薦系統，使用 Music Machines 的資料，透過實驗證明新的方法比使用基於內容的建議或協同過濾推薦算法，更準確的預測和更適當的建議。

2.4.4 文本資料

推薦系統的文獻中，也有與文本資料相關的研究，[39]以協同過濾方法，使用銘傳大學網路虛擬教室網站上的專家系統課程學生瀏覽記錄，提出階段式向前序列與興趣向量來做為隱性評比的方式，並使用多種常見的相似度計算方式做實驗，比較其對推薦準確率的效能。

使用混合式推薦方法的研究，如[40]，該研究提出一個混合式的文件推薦方法，使用 Digg.com 的資料，結合了共同興趣使用者的喜好與使用者原本的喜好來做推薦，第一階段將使用者原本的喜好藉由協同過濾來拓展使用者的喜好，第二階段則是從拓展的喜好建立使用者對文件字詞的喜好，再利用潛在語意檢索提高推薦結果的準確率，結果顯示該研究所提的方法在實務上更具實用性。[41]提出一套個人化的學習系統，將學習測驗結果，應用電腦適性化中試題反應理論，與學習歷程結果及問卷調查的個人偏好作為參考依據，來做英文文章的推薦，幫助學習者增進興趣及學習效率。在[42]當中，使用 Fab System(Stanford University Digital Library Project)，自行蒐集使用者看過的文件當作測試資料，推薦其感興趣的文件。

國立政治大學 National Chengchi University

2.4.5 健保

推薦系統研究，也有與健保相關的論文，如[43]，該研究以內容過濾方法，結合資料探勘的技術，使用決策樹做分類，開發營養資訊輔助推薦系統，將結果交由執行系統決策的營養師，進行評估做決定，並結合雲端運算的技術，讓使用者可在任何時間、地點使用。

而[44]則是藉由患者知識本體的架構建立患者項目，利用多準則方法，將重要性的熵量度運算將資料的用藥程度作出計算，使用藥品知識本體座結合排序並推薦適用的藥品類型。

2.4.6 社群網路服務

社群網路服務方面，利用協同過濾方式的研究，[45]結合 Facebook API 自行收集實驗資料，取得 135 使用者樣本資料，並分析使用者的社會網絡，經由分析社會網絡中成員的偏好，運用正規概念分析(Formal concept analysis,FCA)關聯矩陣理論進行資料分析，預測使用者的潛在興趣。[46]使用協同過濾推薦系統，結合基於使用者的喜好、及基於廣告受歡迎程度，並考慮擴充矩陣的共分群演算法，該研究實驗的資料來源取自一個 Financial Social Web-Site - Ad$Mart，結果顯示出在缺乏已知資料的議題上能產生更佳的處理結果。

[47]使用 Facebook 的資料，結合 Co-occurrences Relationship 和 Mutual Friend Relationship，提出一個新穎的方法來對人臉標註做推薦，有別於其他方法中單純分析照片上傳者在社群網站中與其他人的關係，該研究考慮了欲被標註的那群人臉之間的群組關係。

2.4.7 手機應用程式

在智慧型手機發展下，手機應用程式的推薦研究，大多以協同過濾為基礎。如[48]提出一個 AppFunnel，利用 App 安裝的次數、使用者是否使用過此 App、使用 App 的時間(早/晚)、使用時是否正在移動及 App 的類型等資料，以 Appazaar Recommender System 的資料做實驗，研究發現在 Short-Term Usage 上使用 Context-Aware Recommendation Engine 以及在 Long-Term Usage 使用 Non-Context-Aware Engine 效果較好。[49]提出了一個 AppJoy 程式，自行蒐集使用者資料，透過分析使用者如何實際使用所安裝的應用程序，以 Item-Based Collaborative Filtering Algorithm 對個人做推薦。[50]提出一種基於 SPSO(Set-Based Particle Swarm Optimization)的新的方法-CF-SPSO，使用 Google Play Market 中 AppBrain 及 AppAware 網站的資料。[51]提出一個 EigenApp 的方法，它是一個改善 Memory-Based 模型，該研究將四個模型(Non-Personalized (POP)、Memory-Based (MEM)、PureSVD、EigenApp)做比較，使用 GetJar 的資料做實驗。

國立政治大學
National Chengchi University

第三章 基於雙分群的推薦框架

3.1 總覽

一般的推薦方式往往只考慮其中一種因素，根據使用者的相似度，或是項目的相似度進行推薦。本實驗利用使用者及應用程式的資料，以 Co-Clustering 方法，分別將使用者及應用程式做分群，再以結果進行推薦，流程如下：

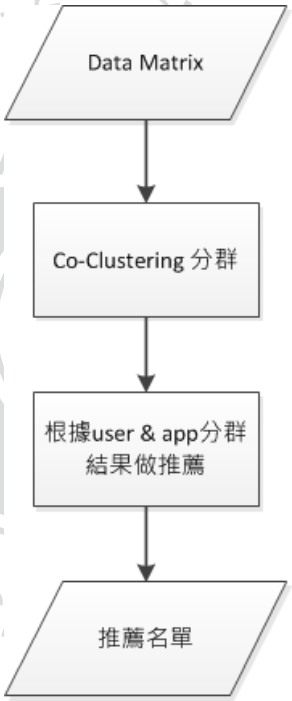

圖 1 本研究推薦流程

目前許多研究使用了 NMF(Non-Negative Matrix Factorization[14])的方法對矩陣做處理。NMF 先將原本的矩陣拆解成兩個或兩個以上低維度的矩陣相乘，使用低維度矩陣做運算，最後以重建的結果矩陣來做推薦。

[14]資料來源：http://en.wikipedia.org/wiki/Non-negative_matrix_factorization

正規劃矩陣分解模型在推薦系統的應用上，已知能產生高品質的預測結果，但是在計算之後產生的模型是靜態的，[52]建立了一個規範化的矩陣分解，當新使用者或新的項目加入時更新矩陣，然而，最後輸出的結果矩陣，在重建與原先的矩陣比較之後，才能做推薦。研究[53]是使用 MovieLens 的資料來做實驗，它提出方法改進原始的 NMF，實驗最後輸出了兩個結果矩陣，必須將結果矩陣相乘，再與原先的矩陣做比較，才能抓取欲推薦的結果。[54]提出了兩階段的評分預測算法，先以 Co-Clustering 演算法對原始的矩陣進行分類，再以改進過的 NMF 方法來預測未知的評分，改進的方法為當有新的使用者對已存在的物品進行評分，或是新的物品被既有的使用者評分之後，使用重新預測的結果重建矩陣，以此進行推薦。[55]為解決 Missing Value Prediction 的問題，使用協同過濾推薦系統做實驗，採用 Co-Clustering 方法同時計算使用者和項目的群集，嘗試恢復資料的原始結構，使得重建矩陣可以被用來預測結果。[56]將矩陣壓縮成一個 Codebook，提出了一個演算法來重建目標矩陣，實驗結果顯示，與其他方法比較，該方法可有效地解決資料稀疏性問題。

上述的研究中，產生的結果矩陣數量，可能為兩個或兩個以上，必須將矩陣重建，與原本的矩陣比對之後，才能找出欲推薦的項目。重建矩陣是利用矩陣相乘的方式，來得到目標矩陣。矩陣相乘由大量運算產生，這是非常耗費成本的。矩陣相乘時，若輸入兩個 $n\times n$ 的矩陣，會輸出一個 $n\times n$ 的矩陣，以演算法 Schoolbook Matrix Multiplication 來說，它的時間複雜度為 $O(n^3)$；以演算法 Strassen Algorithm 來說，它的時間複雜度為 $O(n^{2.807})$；以演算法 Coppersmith–Winograd Algorithm 來說，它的時間複雜度為 $O(n^{2.807})$；以演算法 Williams Algorithm 來說，它的時間複雜度為 $O(n^{2.373})$，當輸入的矩陣為一個 $n\times m$ 和一個 $m\times p$ 的矩陣做相乘時，會輸出一個的 $n\times p$ 矩陣，以演算法 Schoolbook Matrix Multiplication 來說，時間複雜度則為 $O(nmp)$。[15]

本研究所使用的方法，不需做重建矩陣的動作就可以對使用者做應用程式的推薦。

[15]資料來源：http://en.wikipedia.org/wiki/Computational_complexity_of_mathematical_operations#Matrix_algebra

3.2 實際運作範例

在此，我們使用部分資料做範例說明。我們將資料用四種方法做整理，分別在 3.2.1、3.2.2、3.2.3、3.2.4 章節呈現實驗結果，其中資料包含 5 個使用者，以及 21 個應用程式。以下以 ITCC 代表 Information Theoretic Co-Clustering Algorithm，以 MSRICC 代表 Minimum Squared Residue I Co-Clustering Algorithm，以 MSRIICC 代表 Minimum Squared Residue II Co-Clustering Algorithm。

3.2.1 以使用者是否使用應用程式做推薦

此範例中，我們以使用者是否使用應用程式的標準來整理資料，若使用者有使用該應用程式，則將其值設定為 1；若使用者未使用該應用程式，則將其值設定為 0。圖 2 呈現各應用程式是否被使用者使用的情形。其中 X 軸代表應用程式，Y 軸代表被使用的次數，例如 App0006 在此範例中被 User_004 和 User_006 使用，而 App001、App002、App012、App013、App014、App018、App019、App028、App047 沒有被這 5 個使用者所使用。

國立政治大學 National Chengchi University

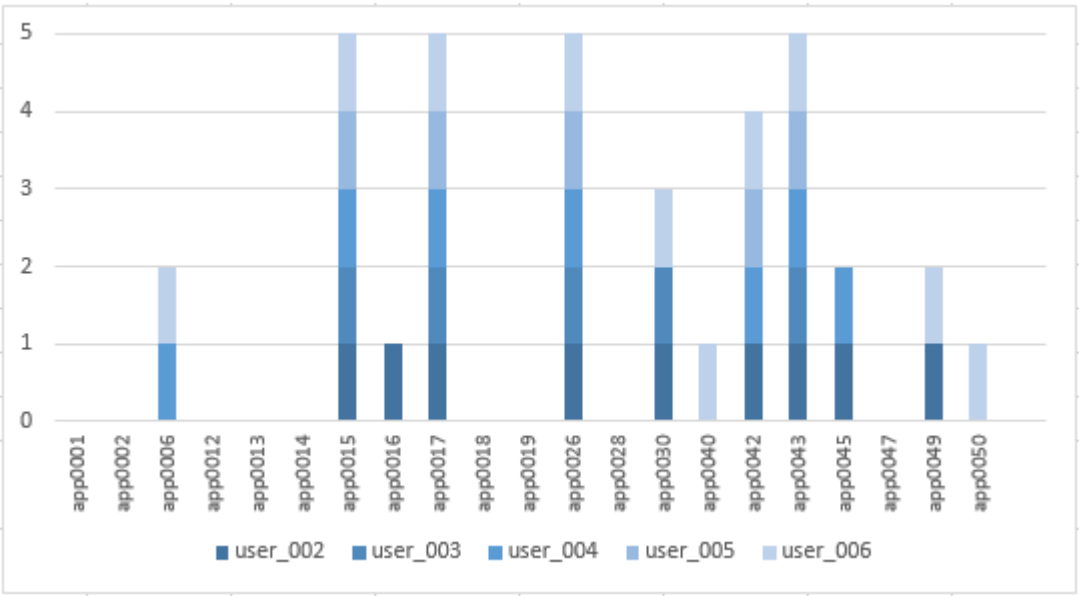

圖 2 實際運作範例中應用程式被使用者使用情形

以下為 5 個使用者和 21 個應用程式分別以三種演算法做 Co-Clustering 分群的結果。分別將使用者與應用程式照分群排序，並聚集了同群的使用者和同群應用程式聚集，我們也把應用程式在 Google Play 中的分類於 App Category 欄位標示，使用者及應用程式以分類後的結果分別排序，並將非 0 值以紅底呈現。

Algorithm	App Category	Tools	Business	Tools	Tools	Tools	Business	Tools	Tools	Productivity	Tools	Tools	Tools	Productivity	Tools	Tools	Productivity	Travel & Local	Lifestyle	Lifestyle	Tools	Travel & Local
ITCC	App Cluster	3	3	3	2	2	2	4	4	4	4	4	4	1	1	1	1	1	1	1	1	1
User Cluster	User Id/App Id	app0001	app0018	app0019	app0006	app0015	app0026	app0002	app0012	app0013	app0014	app0016	app0017	app0028	app0030	app0040	app0042	app0043	app0045	app0047	app0049	app0050
1	user_003	0	0	0	0	1	1	0	0	0	0	0	1	0	1	0	0	1	0	0	0	0
1	user_005	0	0	0	0	1	1	0	0	0	0	0	1	0	0	0	1	1	0	0	0	0
2	user_002	0	0	0	0	1	1	0	0	0	0	1	1	0	1	0	1	1	1	0	1	0
2	user_004	0	0	0	1	1	1	0	0	0	0	0	1	0	0	0	1	1	1	0	0	0
2	user_006	0	0	0	1	1	1	0	0	0	0	0	1	0	1	1	1	1	0	0	1	1

圖 3 範例一，使用 Information Theoretic Co-Clustering Algorithm

Algorithm	App Category	Tools	Business	Tools	Tools	Tools	Business	Tools	Tools	Productivity	Tools	Tools	Tools	Productivity	Tools	Tools	Productivity	Travel & Local	Lifestyle	Lifestyle	Tools	Travel & Local
MSRICC	App Cluster	4	4	4	1	1	1	2	2	2	2	2	2	3	3	3	3	3	3	3	3	3
User Cluster	User Id/App Id	app0001	app0018	app0019	app0006	app0015	app0026	app0002	app0012	app0013	app0014	app0016	app0017	app0028	app0030	app0040	app0042	app0043	app0045	app0047	app0049	app0050
1	user_003	0	0	0	0	1	1	0	0	0	0	0	1	0	1	0	0	1	0	0	0	0
1	user_005	0	0	0	0	1	1	0	0	0	0	0	1	0	0	0	1	1	0	0	0	0
2	user_002	0	0	0	0	1	1	0	0	0	0	1	1	0	1	0	1	1	1	0	1	0
2	user_004	0	0	0	1	1	1	0	0	0	0	0	1	0	0	0	1	1	1	0	0	0
2	user_006	0	0	0	1	1	1	0	0	0	0	0	1	0	1	1	1	1	0	0	1	1

圖 4 範例一，使用 Minimum Squared Residue I Co-Clustering Algorithm

Algorithm	App Category	Tools	Tools	Business	Tools	Tools	Business	Tools	Tools	Tools	Tools	Productivity	Tools	Productivity	Tools	Tools	Productivity	Travel & Local	Lifestyle	Lifestyle	Tools	Travel & Local
MSRIICC	App Cluster	2	2	2	4	4	4	4	1	5	5	5	5	3	3	3	3	3	3	3	3	3
User Cluster	User Id/App Id	app0001	app0015	app0026	app0006	app0014	app0018	app0019	app0016	app0002	app0012	app0013	app0017	app0028	app0030	app0040	app0042	app0043	app0045	app0047	app0049	app0050
2	user_004	0	1	1	1	0	0	0	0	0	0	0	1	0	0	0	1	1	1	0	0	0
2	user_006	0	1	1	1	0	0	0	0	0	0	0	1	0	1	1	1	1	0	0	1	1
1	user_002	0	1	1	0	0	0	0	1	0	0	0	1	0	1	0	1	1	1	0	1	0
1	user_003	0	1	1	0	0	0	0	0	0	0	0	1	0	1	0	0	1	0	0	0	0
1	user_005	0	1	1	0	0	0	0	0	0	0	0	1	0	0	0	1	1	0	0	0	0

圖 5 範例一，使用 Minimum Squared Residue II Co-Clustering Algorithm

由以上結果，在使用者分群的部份，我們發現使用 ITCC 和 MSRICC 所做的分群結果相同，使用者 User_003 和 User_005 都被分在同一個群集，使用者 User_002、User_004、User_006 被分在第二群，而 MSRIICC 則是將 User_002、User_003 和 User_005 分在同一個群集，User_004、User_006 分在第二群。在應用程式分群的部份，使用 ITCC 和 MSRICC 所做的分群結果也相同，而 MSRIICC 則是有較不同的分法。

以 ITCC 和 MSRICC 的結果分析，在使用者群組 2 中，所有的使用者只有 User_002 並未使用 App0006，所以我們會將 App0006 推薦給 User_002；同理，我們會將 App0030 推薦給 User_004、User_005，將 App0042 推薦給 User_003，將 App0045 推薦給 User_006，將 App0049 推薦給 User_004。而 MSRIICC 的部分，我們會將 App0030 推薦給 User_004、User_005，將 App0040 推薦給 User_004，將 App0042 推薦給 User_003，將 App0045 推薦給 User_006，將 App0049、App0050 推薦給 User_004。

3.2.2 以使用應用程式之次數做推薦

此範例中，我們以使用者在資料集中，使用該應用程式的次數做為依據，其值表示使用次數。圖 6 呈現各使用者在每個應用程式的使用次數統計，其中 X 軸代表應用程式及使用者，Y 軸代表被使用的次數，以 User_004 使用 App0006 次數最多，為 2007 次。App001、App002、App012、App013、App014、App018、App019、App028、App047 沒有被這 5 個使用者所使用。

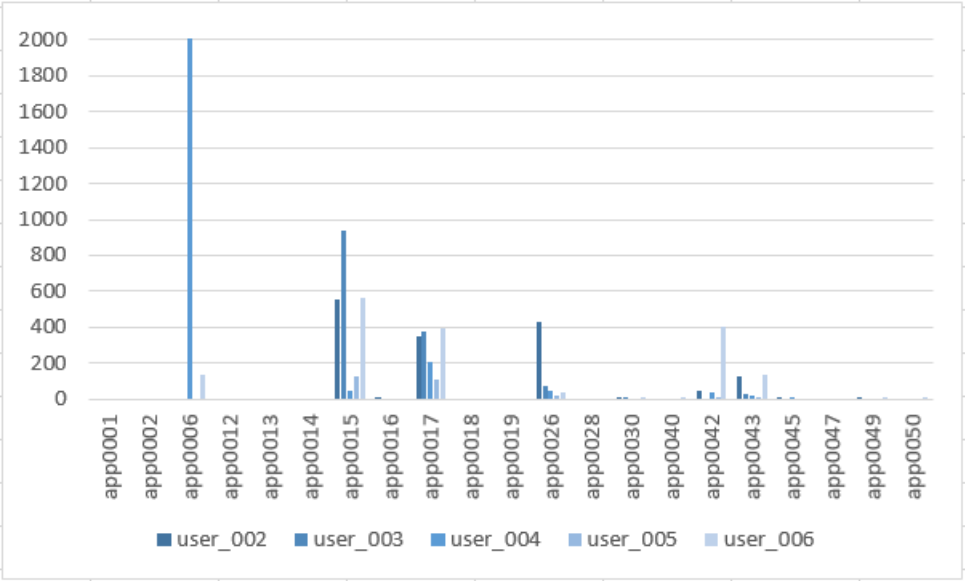

圖 6 使用者使用應用程式的次數示意圖

以下為 5 個使用者和 21 個應用程式分別以三種演算法做 Co-Clustering 分群的結果。分別將使用者與應用程式照分群結果排序，並且聚集了同群的使用者和同群的應用程式，我們也把應用程式在 Google Play 中的分類於 App Category 欄位標示，使用者及應用程式以分類後的結果分別排序，並將非 0 值以紅底呈現。

Algorithm	App Category	Tools	Tools	Tools	Tools	Tools	Tools	Business	Tools	Business	Productivity	Tools	Tools	Productivity	Travel & Local	Lifestyle	Lifestyle	Tools	Travel & Local	Productivity	Tools	Tools
ITCC	App Cluster	1	3	3	2	2	4	4	4	4	4	4	4	4	4	4	4	4	4	5	5	5
User Cluster	User Id/App Id	app0001	app0002	app0012	app0006	app0014	app0015	app0018	app0019	app0026	app0028	app0030	app0040	app0042	app0043	app0045	app0047	app0049	app0050	app0013	app0016	app0017
1	user_002	0	0	0	0	0	556	0	0	431	0	10	0	46	129	4	0	2	0	0	13	346
1	user_003	0	0	0	0	0	939	0	0	70	0	4	0	0	29	0	0	0	0	0	0	379
1	user_005	0	0	0	0	0	127	0	0	17	0	0	0	1	13	0	0	0	0	0	0	108
1	user_006	0	0	0	139	0	563	0	0	39	0	6	3	402	135	0	0	1	2	0	0	397
2	user_004	0	0	0	2007	0	48	0	0	42	0	0	0	34	21	1	0	0	0	0	0	208

圖 7 範例二，使用 Information Theoretic Co-Clustering Algorithm

Algorithm	App Category	Tools	Tools	Tools	Tools	Tools	Tools	Business	Tools	Business	Productivity	Tools	Tools	Productivity	Travel & Local	Lifestyle	Lifestyle	Tools	Travel & Local	Productivity	Tools	Tools
MSRICC	App Cluster	1	2	2	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	4	4	4
User Cluster	User Id/App Id	app0001	app0002	app0012	app0006	app0014	app0015	app0018	app0019	app0026	app0028	app0030	app0040	app0042	app0043	app0045	app0047	app0049	app0050	app0013	app0016	app0017
2	user_002	0	0	0	0	0	556	0	0	431	0	10	0	46	129	4	0	2	0	0	13	346
2	user_003	0	0	0	0	0	939	0	0	70	0	4	0	0	29	0	0	0	0	0	0	379
2	user_005	0	0	0	0	0	127	0	0	17	0	0	0	1	13	0	0	0	0	0	0	108
2	user_006	0	0	0	139	0	563	0	0	39	0	6	3	402	135	0	0	1	2	0	0	397
1	user_004	0	0	0	2007	0	48	0	0	42	0	0	0	34	21	1	0	0	0	0	0	208

圖 8 範例二，使用 Minimum Squared Residue I Co-Clustering Algorithm

Algorithm	App Category	Tools	Tools	Tools	Tools	Productivity	Tools	Tools	Tools	Tools	Business	Tools	Business	Productivity	Tools	Tools	Productivity	Travel & Local	Lifestyle	Lifestyle	Tools	Travel & Local
MSRIICC	App Cluster	3	2	2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
User Cluster	User Id/App Id	app0001	app0002	app0012	app0006	app0013	app0014	app0015	app0016	app0017	app0018	app0019	app0026	app0028	app0030	app0040	app0042	app0043	app0045	app0047	app0049	app0050
1	user_002	0	0	0	0	0	0	556	13	346	0	0	431	0	10	0	46	129	4	0	2	0
1	user_003	0	0	0	0	0	0	939	0	379	0	0	70	0	4	0	0	29	0	0	0	0
1	user_005	0	0	0	0	0	0	127	0	108	0	0	17	0	0	0	1	13	0	0	0	0
1	user_006	0	0	0	139	0	0	563	0	397	0	0	39	0	6	3	402	135	0	0	1	2
2	user_004	0	0	0	2007	0	0	48	0	208	0	0	42	0	0	0	34	21	1	0	0	0

圖 9 範例二，使用 Minimum Squared Residue II Co-Clustering Algorithm

由以上結果，在使用者分群的部份，我們發現三種演算法都將 User_004 分成一個群集，另外四位使用者分在第二個群集。在應用程式分群的部份，使用 ITCC 和 MSRICC 所做的分群結果相同，而 MSRIICC 則是較不同。

以 ITCC 和 MSRICC 的結果分析，在使用者群組 1 中，所有的使用者只有 User_005 並未使用 App0030，所以我們會將 App0030 推薦給 User_005；同理，我們會將 App0042 推薦給 User_003。而 MSRIICC 的部分，我們會將 App0030 推薦給 User_005，將 App0042 推薦給 User_003。

從 3.2.1 和 3.2.2 兩個範例中，我們發現 ITCC 和 MSRICC 兩個演算法，在使用者和應用程式的分群，皆得到相同的結果，而 MSRIICC 則是有不同的分群結果。

3.2.3 以使用者使用應用程式的頻率做推薦

此範例中，我們以使用者在資料集中，使用該應用程式的頻率做為依據，其值表示使用頻率，例如 User_004 總共使用了 2361 次的應用程式，其中 App0006 使用了 2007 次，表示 App0006 佔總使用的百分之八十五，其值以 0.85 表示。圖 10 呈現各使用者在每個應用程式的使用頻率統計，其中 X 軸代表應用程式及使用者，Y 軸代表被使用的頻率，以 User_004 使用 App0006 頻率最高，為 0.85。App001、App002、App012、App013、App014、App018、App019、App028、App047 沒有被這 5 個使用者所使用。

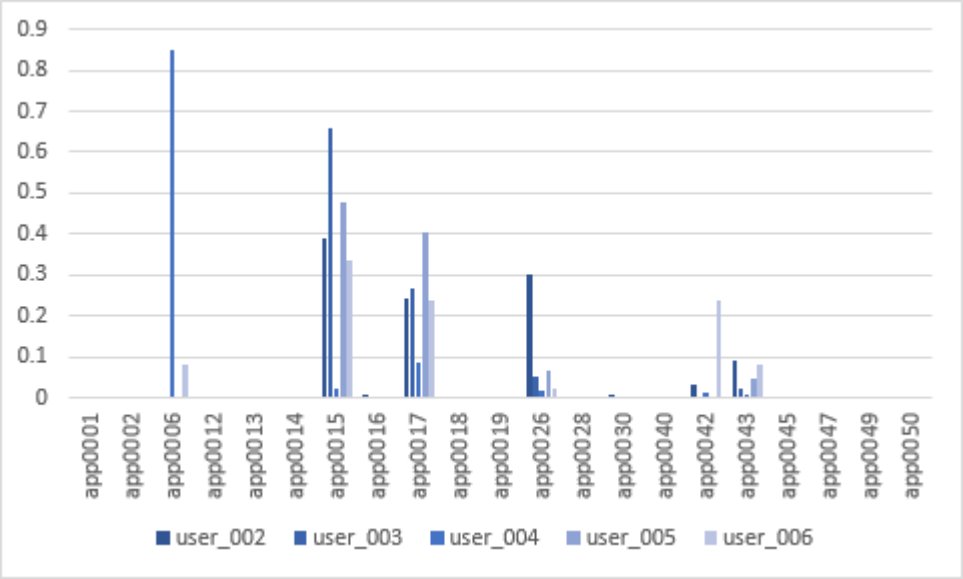

圖 10 使用者使用應用程式的頻率示意圖

以下為 5 個使用者和 21 個應用程式分別以三種演算法做 Co-Clustering 分群的結果。分別將使用者與應用程式照分群結果排序，並且聚集了同群的使用者和同群的應用程式，我們也把應用程式在 Google Play 中的分類於 App Category 欄位標示，使用者及應用程式以分類後的結果分別排序，並將非 0 值以紅底呈現。

Algorithm	App Category	Tools	Tools	Tools	Tools	Tools	Business	Tools	Tools	Business	Productivity	Tools	Tools	Productivity	Travel & Local	Lifestyle	Lifestyle	Tools	Travel & Local	Productivity	Tools	Tools
ITCC	App Cluster	3	1	1	2	2	2	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
User Cluster	User Id/App Id	app0001	app0002	app0012	app0013	app0016	app0017	app0006	app0014	app0015	app0018	app0019	app0026	app0028	app0030	app0040	app0042	app0043	app0045	app0047	app0049	app0050
1	user_002	0	0	0	0	0.01	0.24	0	0	0.39	0	0	0.3	0	0.01	0	0.03	0.09	0	0	0	0
1	user_003	0	0	0	0	0	0.27	0	0	0.66	0	0	0.05	0	0	0	0	0.02	0	0	0	0
1	user_005	0	0	0	0	0	0.41	0	0	0.48	0	0	0.06	0	0	0	0	0.05	0	0	0	0
1	user_006	0	0	0	0	0	0.24	0.08	0	0.33	0	0	0.02	0	0	0	0.24	0.08	0	0	0	0
2	user_004	0	0	0	0	0	0.09	0.85	0	0.02	0	0	0.02	0	0	0	0.01	0.01	0	0	0	0

圖 11 範例三，使用 Information Theoretic Co-Clustering Algorithm

Algorithm	App Category	Tools	Tools	Tools	Tools	Tools	Business	Tools	Tools	Business	Productivity	Tools	Tools	Productivity	Travel & Local	Lifestyle	Lifestyle	Tools	Travel & Local	Productivity	Tools	Tools
MSRICC	App Cluster	1	1	1	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
User Cluster	User Id/App Id	app0001	app0002	app0012	app0013	app0016	app0017	app0006	app0014	app0015	app0018	app0019	app0026	app0028	app0030	app0040	app0042	app0043	app0045	app0047	app0049	app0050
1	user_002	0	0	0	0	0.01	0.24	0	0	0.39	0	0	0.3	0	0.01	0	0.03	0.09	0	0	0	0
1	user_006	0	0	0	0	0	0.24	0.08	0	0.33	0	0	0.02	0	0	0	0.24	0.08	0	0	0	0
2	user_003	0	0	0	0	0	0.27	0	0	0.66	0	0	0.05	0	0	0	0	0.02	0	0	0	0
2	user_004	0	0	0	0	0	0.09	0.85	0	0.02	0	0	0.02	0	0	0	0.01	0.01	0	0	0	0
2	user_005	0	0	0	0	0	0.41	0	0	0.48	0	0	0.06	0	0	0	0	0.05	0	0	0	0

圖 12 範例三，使用 Minimum Squared Residue I Co-Clustering Algorithm

Algorithm	App Category	Tools	Tools	Tools	Tools	Tools	Business	Tools	Tools	Business	Productivity	Tools	Tools	Productivity	Travel & Local	Lifestyle	Lifestyle	Tools	Travel & Local	Productivity	Tools	Tools
MSRIICC	App Cluster	1	2	2	3	3	3	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
User Cluster	User Id/App Id	app0001	app0002	app0012	app0013	app0016	app0017	app0006	app0014	app0015	app0018	app0019	app0026	app0028	app0030	app0040	app0042	app0043	app0045	app0047	app0049	app0050
2	user_002	0	0	0	0	0.01	0.24	0	0	0.39	0	0	0.3	0	0.01	0	0.03	0.09	0	0	0	0
2	user_003	0	0	0	0	0	0.27	0	0	0.66	0	0	0.05	0	0	0	0	0.02	0	0	0	0
2	user_005	0	0	0	0	0	0.41	0	0	0.48	0	0	0.06	0	0	0	0	0.05	0	0	0	0
2	user_006	0	0	0	0	0	0.24	0.08	0	0.33	0	0	0.02	0	0	0	0.24	0.08	0	0	0	0
1	user_004	0	0	0	0	0	0.09	0.85	0	0.02	0	0	0.02	0	0	0	0.01	0.01	0	0	0	0

圖 13 範例三，使用 Minimum Squared Residue II Co-Clustering Algorithm

由以上結果，在使用者分群的部份，我們發現 ITCC 和 MSRIICC 兩種演算法都將 User_004 分成一個群集，另外四位使用者分在第二個群集。在應用程式分群的部份，使用 ITCC 和 MSRIICC 所做的分群結果相同，而 MSRICC 則是較不同。

以 ITCC 和 MSRICC 的結果分析，在第一個使用者群組中，有一半的使用者使用了 App0042，所以我們會將 App0042 推薦給 User_003 和 User_005。而 MSRIICC 的部分，我們會將 App0016 推薦給 User_006，將 App0006 推薦給 User_002，將 App0030 推薦給 User_006。

由上述三個範例，我們發現各演算法在 App 分類時，不一定會把相同類別的 App 分在同一個 Cluster。Google Play 的分類使由程式開發者主觀定義的，或許可以反應使用者的操作方式，稍做調整。

第三章範例主要解釋本研究如何做推薦，並非比較演算法之間的差異。詳見第四章更大規模的實驗。

國立政治大學 National Chengchi University

第四章 實驗

相較於第三章的實驗，本章實驗使用較大規模的資料集。資料集為 2010 年 9 月至 2011 年 3 月的使用紀錄，共 26 萬多筆資料，其中包含有 30 個使用者、1125 個手機應用程式。

4.1 環境與流程

4.1.1 實驗環境

本研究進行實驗時所使用的系統，硬體為一般個人電腦(單機)，軟體環境則是簡述如下：

- 作業系統，使用 Ubuntu v12.04
- 程式語言，使用 C++、Python
- 關聯式資料庫使用 MySQL (Workbench CE 6.0.8)
- 另外，本研究使用由德克薩斯州大學資訊科學學系所開發的 Co-clustering Software (Version 1.1)[16]進行雙分群

4.1.2 實驗流程

此實驗為一個基於離線(Off-Line)及模型基礎(Model-Base)的架構。我們將資料集整理成 CCS(Compressed Column Storage)檔，使用 Co-Clustering Software 三種演算法對使用者

[16]資料來源：http://www.cs.utexas.edu/Users/dml/Software/cocluster.html

及應用程式做分群，將三種演算法輸出的結果，分別以 Python[17]處理，轉換格式成 CSV(Comma Separated Value)檔並匯入資料庫，對每個 User，做不同的推薦。首先，我們找出與此 User 相同群組的 User 名單，在各應用程式分類的群組中，使用三種方法計算同群 User 的使用狀況，決定是否推薦此應用程式。

第一種方法，我們找出每個 App Cluster 中，應用程式被最多同群 User 使用的最優先推薦。例如：App Cluster 是 1 的 App0050，在同群的 User 中有 5 位使用，他的分數就是 5；App Cluster 是 1 的 App0100，在同群的 User 中有 8 位使用，他的分數就是 8，所以我們會優先推薦 App0100。若是遇到分數相同的情況下，我們會將兩個應用程式同時推薦給使用者。

第二種方法，我們將資料分別整理成以使用次數及使用頻率計算，推薦使用次數或頻率最高的應用程式。使用次數的部分，我們找出每個 App Cluster 中，應用程式被同群 User 使用次數最多的優先推薦。例如：App Cluster 是 1 的 App0050，被 5 個同群的 User 分別使用 100 次，他的分數就是 500；App Cluster 是 1 的 App0100，被 2 個同群的 User 分別使用 200 次，他的分數就是 400，所以我們會優先推薦 App0050。另外，我們將每個使用者在各應用程式的使用次數轉換為個人使用頻率，計算每個 App Cluster 中，應用程式被同群 User 使用頻率最高的優先推薦。例如：App Cluster 是 1 的 App0050，被 5 個同群的 User 使用的頻率分別是 0.39、0.36、0.09、0.12、0.06，那麼他的分數就是 1.02；App Cluster 是 1 的 App0100，被 2 個同群的 User 使用的頻率分別是 0.2、0.16，那麼他的分數就是 0.36，所以我們會優先推薦 App0050。

我們將欲推薦的 User 從未使用過的應用程式列出來，並找出各應用程式的分數，再依序找出此 App Cluster 裡分數最高的前五個應用程式，推薦給使用者。請注意，在我們的實驗中，實際上沒有真正對使用者進行推薦，而是以模擬推薦的方式，驗證使用者是否用

[17] 參考網址：https://www.python.org/

到我們所推薦的應用程式；然而，使用者是否使用某應用程式，也不能完全代表他或她是否喜歡該應用程式。圖 18 為實驗一流程說明。

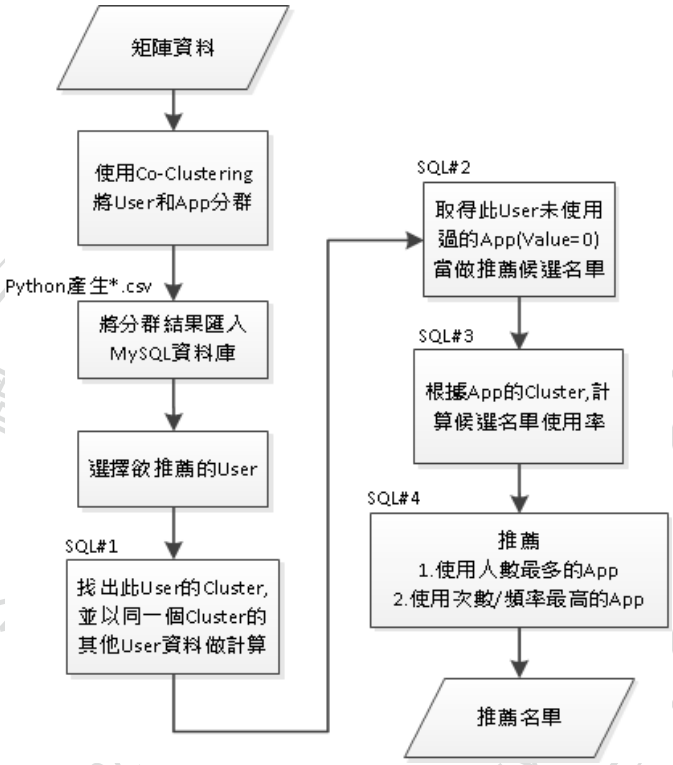

圖 14 實驗流程圖

以下介紹實驗一所用的 SQL。

SQL#1 找出與欲推薦 User 同一個 Cluster 其他 User，並以這些同群使用者的資料為基準，其中 user_cluster_result 儲存了使用者分群的結果，$user_id$表示欲推薦的使用者。

```
SELECT user FROM user_cluster_result

WHERE user_cluster = (

                        SELECT user_cluster FROM user_cluster_result

                        WHERE User='$user_id$'

                    )
```

SQL#2 取得 User 未使用過的 App 做為推薦候選名單，user_app_value 這個 Table 儲存了使用者對應用程式的操作狀況，app_cluster_result 儲存了應用程式分群的結果。

```
SELECT a.app_id, a.value, b.app_cluster

FROM user_app_value a, app_cluster_result b

WHERE a.User='$user_ud$' and a.value='0' and a.app_id=b.app_id
```

國立政治大學 National Chengchi University

SQL#3.1 計算每個應用程式的分群$app_cluster$的使用人數，使用@row 語法依照人數多少從高到低排序。

```
SELECT *,@row := @row + 1 as row  FROM (
        SELECT app_id, count(user) AS user_app_value,app_cluster
        FROM recommender_list a, (
                SELECT @row := 0
                                                ) r
        WHERE app_cluster=$app_cluster$ AND user_app_value <> 0
        GROUP BY app_id ) b
ORDER BY user_app_value DESC
```

SQL#3.2 計算每個應用程式的分群$app_cluster$的使用次數/頻率，使用@row 語法依照使用次數/頻率多少從高到低排序。

```
SELECT *,@row := @row + 1 AS row  FROM (
SELECT app_id, SUM(user_app_value) AS user_app_value, app_cluster
FROM recommender_list a, (SELECT @row := 0) r
WHERE app_cluster=$app_cluster$
GROUP BY app_id ) b
ORDER BY  user_app_value DESC
```

SQL#4 推薦排名前 5 的應用程式。

```
SELECT * FROM recommender_result

WHERE app_sequence<=5
```

當我們在整理實驗所需的資料集時，發現類似的應用程式在 Google Play 的分類有些特別，例如像是部分 Bus 相關的應用程式分類為 Transportation，而部分 MRT 相關的應用程式分類為 Travel & Local，我們設計了額外的實驗，從 User 使用應用程式的相似性，探討應用程式在 Co-Clustering 做分類的結果，想了解 Google Play 的分類情形是否可做調整。

Co-Clustering Software 所提供的參數有許多，必要輸入的參數有演算法(–A)、行分群數量(–C)、 輸入之資料(–I)、列分群數量(–R)四種。本研究所用的參數為-A、-C、-I、-R、-O 五種，在演算法中，我們使用了三種演算法分別做實驗；以應用程式為列(Column)，將應用程式分成 30 個 Cluster；以使用者為行(Row)，將使用者分為 2 個 Cluster。本實驗的參數設定說明如表 2。

國立政治大學 National Chengchi University

表 2 本實驗使用之參數說明

參數	參數名稱	參數說明	範例
-A	使用之演算法	· Algorithm Type ➢ 包含三種演算法 · i -- Information Theoretic Co-Clustering algorithm(DEFAULT) · e -- Minimum Squared Residue I Co-Clustering Algorithm · r -- Minimum Squared Residue II Co-Clustering Algorithm	-A e
-C	行分群數量	· COL Cluster Num ➢ 輸入行欄位欲分群數量，預設值為 1	-C 30
-I	輸入之資料	· inputMatrixType ➢ d：稠密矩陣 ➢ s：稀疏矩陣 · inputFormatType ➢ 若輸入稠密矩陣參數 d，必須指定檔案格式 · s：矩陣與其維度資訊分別儲存於兩個檔案 · t：矩陣與其維度資訊儲存於同一檔案 · inputFilename	-I s f user_app_count
-R	列分群數量	· ROW Cluster Num ➢ 輸入列欄位欲分群數量，預設值為 1	-R 2
-O	輸出結果	· outputFileType ➢ c：輸出 Co-Cluster 結果 · b：以每三行數值描述一個 Cluster，第一行為此 Cluster 的行列數，第二、三行分別代表為第幾列、行屬於此 Cluster · s：輸出列、行的 Cluster 於兩列數值 · 0：Cluster 標籤從 0 開始 · 1：Cluster 標籤從 1 開始 ➢ o：輸出目標函數數值 ➢ s：輸出統計資訊 · outputAccessMode：輸出模式 ➢ a：將結果附加在檔案後 ➢ o：將結果覆蓋檔案 · outputFilename：輸出的檔案名稱	-O c s 1 o result

國立政治大學 National Chengchi University

4.2 資料集

本實驗所使用的資料，由政治大學資訊科學學系 XMIND Team[18]所提供的[57]，該研究開發了一個實驗平台，可以記錄和分析使用者在智慧型手機上的所有操作。資料以側錄的方式，將使用者實際操作手機應用程式的狀況記錄下來。表 3 為本研究所使用的部分資料紀錄，包含 ID、使用者、使用時間及應用程式。本資料集蒐集了從 2010 年 9 月至 2011 年 3 月的使用紀錄，共 26 萬多筆資料，共有 30 個使用者、1125 個手機應用程式，圖 19 為各使用者在紀錄中使用應用程式的數量，X 軸代表使用者，Y 軸呈現該使用者有使用的應用程式數量，其中有 5 位使用者 User_001、User_008、User_012、User_014、User_026 在 1125 個應用程式中，沒有任何紀錄。紀錄包含使用者 ID、應用程式(App)、使用時間等資訊。在 1125 個手機應用程式中，根據 Google Play 分類，將他們分為 30 大類，在表 6 中欄位 Android Market Category 說明，圖 19 的 X 軸為 Google Play 分類，Y 軸表示各分類的數量，另外有 429 個未被分類的 App 沒有在圖 19 中呈現。

表 3 使用者操作行為紀錄(範例)

ID	使用者	使用時間	應用程式
1	User_002	2010-11-18 20:30:15	com.whatsApp
2	User_002	2011-02-08 21:55:20	com.android.browser
3	User_003	2011-02-09 22:42:05	com.facebook.katana
4	User_003	2010-10-01 09:50:29	com.google.android.location

[18] 參考網址：http://x-mind.km.nccu.edu.tw/

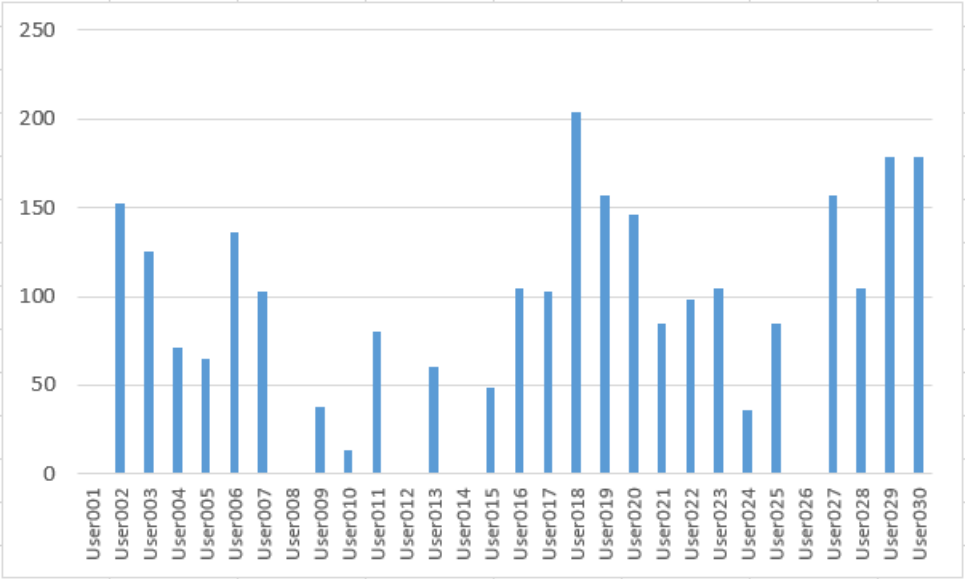

圖 15 使用者使用應用程式個數

表 4 手機應用程式分類說明(範例)

ID	Name	Android Market Category	Package Name
App0033	Angry Birds	Arcade & Action	com.rovio.angrybirds
App0160	Kobo eBooks	Books & Reference	com.kobobooks.android
App0061	Unblock Me FREE	Brain & Puzzle	com.kiragames.unblockmefree
App0014	Adao File Manager	Business	com.adao.android.afm
App0389	WeWeWeb Bridge (Free)	Cards & Casino	net.weweweb.android.free.bridge
App0081	Penguin I - Classic	Casual	com.hz.game.penguin
App0252	Droid Comic Viewer	Comics	net.androidcomics.acv
App0024	Gmail	Communication	com.google.android.gm
App0554	Kids Connect the Dots Lite	Education	zok.android.dots
App0082	Talking Roby Celik Free	Entertainment	com.outfit7.talkingrobyfree
App1056	Daily Money	Finance	com.bottleworks.dailymoney
App0117	Self-Hypnosis for Sound Sleep	Health & Fitness	imoblife.soundSleep
App0084	SpeechSynthesis Data Installer	Libraries & Demo	com.svox.langpack.installer
App0045	Koe Fusen	Lifestyle	com.google.android.voicesearch
App0037	Music Player Lock Screen	Media & Video	com.android.music
App0059	Shazam	Music & Audio	com.shazam.android
App0076	gReader (Google Reader | RSS)	News & Magazines	com.noinnion.android.greader.reader
App0079	Beautiful Widgets	Personalization	com.levelup.beautifulwidgets
App0013	Root Explorer (File Manager)	Productivity	com.speedsoftware.rootexplorer
App0093	FxCamera	Photography	ymst.android.fxcamera
App0457	Finger Race	Racing	com.hz.game.finger
App0468	Key Ring Reward Cards	Shopping	com.froogloid.kring.google.zxing.client.android
App0921	LED Scroller 3 - FREE	Social	no.dkit.android.ledscrollerfull
App0995	MySwimmingTimes	Sports	com.my
App0092	SmartMonitor Lite	Tools	com.smart.SmartMonitorLite
App0098	HOMERUN BATTLE 3D FREE	Sports Games	com.com2us.HB3DLITE
App0288	Taipei Bus Situation Alert	Transportation	lbs.ebus
App0341	NZRoadInfo	Travel & Local	com.android.email
App0842	The Weather Channel	Weather	com.weather.Weather
App0007		Uncategorized	com.htc.album

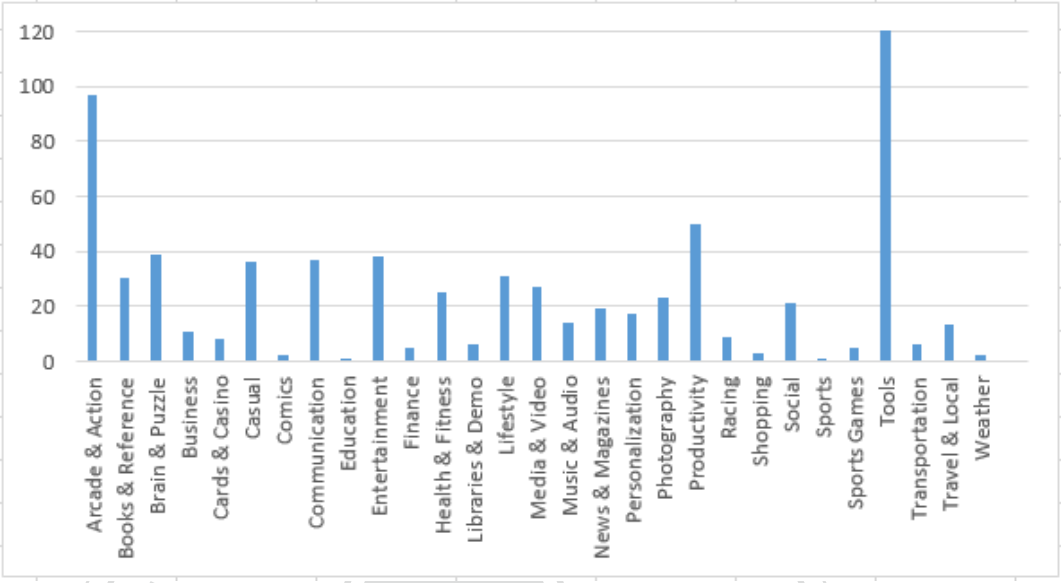

圖 16 應用程式在各分類的分布

我們將資料整理如表 5，分別進行實驗。資料型態(Data Type)分為使用者是否使用應用程式、使用的次數以及使用頻率三種，另外也依照使用的時段分為前段和後段，前段時間為 2010 年 9 月 27 日至 2011 年 1 月 31 日，後段時間為 2011 年 2 月 1 日至 2011 年 3 月 22 日；以及全部時間，2010 年 9 月 27 日至 2011 年 3 月 22 日。

在 4.3 的實驗中，我們針對每種資料型態(Data Type)和前/後時段的組合，使用 Co-Clustering 跑三個演算法，做三種不同的推薦方式。並以前段時間的資料集作為基礎來進行推薦，再以後段時間的資料集做比對，驗證推薦的應用程式是否真的被使用。4.3.1 小節的推薦結果所使用的資料集中，記錄了每位使用者在前段時間中，是否使用過應用程式，若使用者有使用該應用程式，其值以 1 表示，若使用者未使用該應用程式，其值以 0 表示，此資料集中，每位使用者平均使用了 117 個應用程式，我們以這個資料集，來對使用者做推薦；4.3.2 小節的推薦結果使用的資料集，是使用者在前段時間內使用應用程式的次數。4.3.3 小節則是將資料集轉換為使用頻率來做推薦，舉例說明，使用者 User_002 使用所有

應用程式的次數為 1537 次，對於應用程式 App0017 則會被記錄為 0.23(使用次數 346 佔總使用次數 1537 的比例)，而使用者 User_005 對於應用程式 App0017 則會被記錄為 0.41。

在 4.4 實驗中，我們不將資料分為前後段時間，以整個資料集的紀錄做實驗，分成 4.4.1 是否使用應用程式的資料集，4.4.2 使用者使用應用程式次數的資料集，以及 4.4.3 使用者使用應用程式頻率的資料集，探討使用三種演算法分群的結果。

實驗以三種方式整理資料集，比較每個資料集的實驗結果。一開始，我們只對每位使用者有或沒有使用應用程式的角度做整理，資料集一、四，“是否使用應用程式”中，無法看出使用者對應用程式的依賴程度。我們的一個想法是，以使用者的使用每個應用程式的次數來觀察，或許更能推薦流行及熱門的應用程式給使用者，所以我們用這個方法整理資料集二、四，進而做推薦。接著我們思考，每位使用者操作手機的次數不同，每個人每天使用手機的次數可能很極端，因此我們的另一個想法是，或許將使用者使用應用程式的次數轉換成使用頻率，更能代表使用者操作手機的狀況，資料集三、六就是以此方式做整理，做貼近個人化推薦。

表 5 實驗使用之資料集及其結果說明

Data Type	**時段**	**資料集**	**實驗說明與結果**
是否使用應用程式	前段	資料集一	4.3.1
使用次數	前段	資料集二	4.3.2
使用頻率	前段	資料集三	4.3.3
是否使用應用程式	全部	資料集四	4.4.1
使用次數	全部	資料集五	4.4.2
使用頻率	全部	資料集六	4.4.3

國立政治大學 National Chengchi University

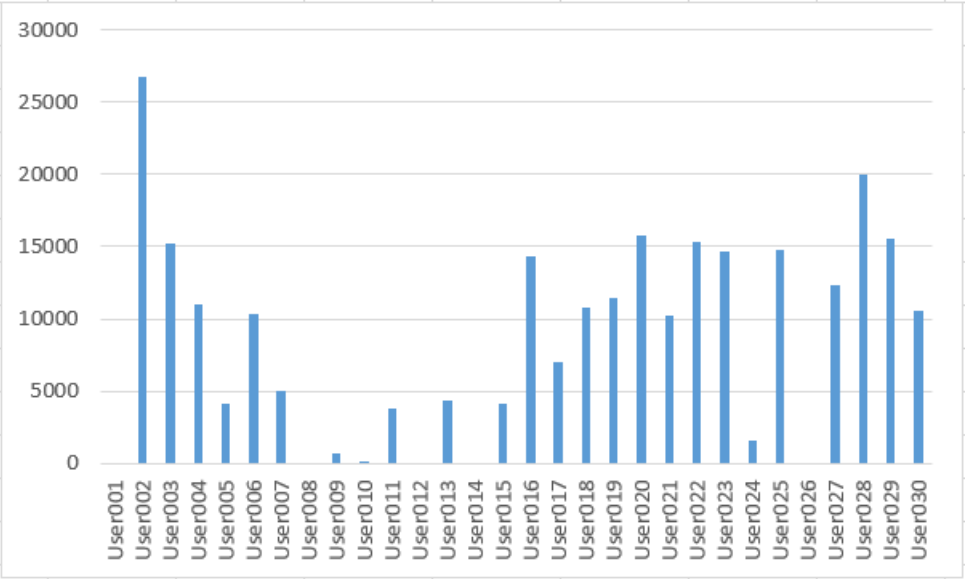

圖 17 各使用者使用應用程式的總次數

我們將資料以矩陣的方式儲存，因為使用的資料集中，多數的元素並沒有資料，為稀疏矩陣(Sparse Matrix)，由於矩陣在程式中常使用二維陣列表示，二維陣列的大小與使用的記憶體空間成正比，如果多數的元素沒有資料，則會造成記憶體空間的浪費，為此，必須設計稀疏矩陣的陣列儲存方式，利用較少的記憶體空間儲存完整的矩陣資訊。

在 Co-Clustering Software 的使用上，我們將原始資料轉換為 CCS 格式。CCS 格式的檔案包含四個部分，*_dim、*_col_ccs、*_row_ccs、*_tfn_nz。*_dim 為一個一維矩陣，包含了列的數量、行的數量以及陣列中非 0 值的數量。*_col_ccs 這個檔案將矩陣每列(Column)非 0 值開始的指標(Point)位置，並在最後再加上一個指標位置(Length+1)。*_row_ccs 記錄了矩陣中，每列(Column)非 0 數值在行(Row)的位置，其值從 0 開始。*_tfn_nz 此檔案將矩陣中，所有非 0 的值記錄下來。

假設實際的資料為下列矩陣，則整理後的 CCS 檔案為表 6：

$$\begin{bmatrix} 10 & 20 & 0 & 0 & 0 & 0 \\ 0 & 30 & 0 & 40 & 0 & 0 \\ 0 & 0 & 50 & 60 & 70 & 0 \\ 0 & 0 & 0 & 0 & 0 & 80 \end{bmatrix}$$

表 6 稀疏矩陣檔案說明

檔案名稱	**檔案格式**	**範例**
_dim	（行數, 列數, 非 0 值的數量）	[a 4 6 8]
_col_ccs	非 0 值的 column point	[a 0 1 3 4 6 7 8]
_row_ccs	非 0 值的 row indices	[a 0 0 1 2 1 2 2 3]
_tfn_nz	非 0 值	[a 10 20 30 40 50 60 70 80]

4.3 實驗結果(一)

在此，我們將前段時間為基礎對實用者做推薦，以後段時間資料驗證結果。在每個 App Cluster 皆推薦前 5 個 App 給使用者，計算準確率(Precision)與召回率(Recall)來評定三個演算法對於每一個 Cluster 的推薦結果。準確率為推薦正確的應用程式數量(TP)，除以推薦正確(TP)加上推薦錯誤的應用程式數量(FP)；召回率為推薦正確的應用程式數量(TP)，除以推薦正確(TP)加上應該被推薦卻未被推薦的應用程式數量(FN)。其中，FN 是以應用程式在推薦名單的數量，扣除推薦成功的數量 TP。一般來說準確率與召回率不會同時都很高，所以在不同的系統需求下所要求的就相對不同。在我們的實驗中主要是希望推薦使用者真正感興趣的應用程式，得到少數的推薦結果，會比得到過多的結果好。因此我們認為，準確率較能評估我們的推薦結果。

表 7 Confusion Matrix 說明

TP：推薦正確的應用程式數量	FP：推薦錯誤的應用程式數量
FN：應該被推薦卻未被推薦的應用程式數量	TN：未被推薦且為使用的應用程式

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{F} - \text{measure} = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

以下以 ITCC 代表 Information Theoretic Co-Clustering Algorithm，以 MSRICC 代表 Minimum Squared Residue I Co-Clustering Algorithm，以 MSRIICC 代表 Minimum Squared Residue II Co-Clustering Algorithm。

此實驗主要是推薦給使用者從未使用過的應用程式，所以當使用者在前、後段時間都有使用的應用程式，不會出現在推薦清單裡。我們統計了每個使用者在後段時間新增使用的應用程式數量如下表，User_030 使用 34 個應用程式為最多，而 User_009 使用的應用程式則最少，另外 User_005、User_13、User_019 和 User_027 在後段時間所使用的應用程式也不多。

表 8 使用者只在後段時間所使用的應用程式個數

使用者	應用程式數量	使用者	應用程式數量
User_002	31	User_019	4
User_003	24	User_020	12
User_004	11	User_021	10
User_005	3	User_022	15
User_006	23	User_023	10
User_007	28	User_025	5
User_009	1	User_027	4
User_011	21	User_028	6
User_013	2	User_029	27
User_016	7	User_030	34
User_018	25		

4.3.1 以使用者是否使用應用程式做推薦(YN)

我們以同群使用者，使用人數最多的 App 為基準，推薦前 5 名的應用程式給使用者，得到了以下結果(表 9)。以 User_002 來說，準確率 ITCC 方法最高，約 0.019，MSRICC 的 0.012 最低，召回率也是 ITCC 方法最高，約 0.024，MSRICC 的 0.017 最低。以 User_011 來說，準確率以 MSRIICC 方法最高，約 0.018，MSRICC 的 0.012 最低，召回率也是 MSRIICC 方法最高，約 0.037，MSRICC 的 0.026 最低。而 User_022，準確率以 ITCC 方法最高，約 0.013，MSRICC 和 MSRIICC 的方法較低約 0.01，召回率也是 ITCC 方法較高，約 0.033，以 MSRICC 和 MSRIICC 的方法較低約 0.03。另外，我們也計算三種演算法的平均準確率與召回率，如表 10。

表 9 實驗 4.3.1，以使用者是否使用應用程式資料做推薦，準確率、召回率及 F-measure 比較表

Algorithm	**ITCC**			**MSRICC**			**MSRIICC**		
User	**Precision**	**Recall**	**F-measure**	**Precision**	**Recall**	**F-measure**	**Precision**	**Recall**	**F-measure**
User_002	0.15	0.1935484	0.1690141	0.1111111	0.1612903	0.1315789	0.1333333	0.1935484	0.1578947
User_003	0.225	0.375	0.28125	0.0888889	0.1666667	0.115942	0.0666667	0.125	0.0869565
User_004	0.1	0.3636364	0.1568627	0.0444444	0.1818182	0.0714286	0.0444444	0.1818182	0.0714286
User_005	0.05	0.6666667	0.0930233	0.0444444	0.6666667	0.0833333	0.0444444	0.6666667	0.0833333
User_006	0.1	0.173913	0.1269841	0.0888889	0.173913	0.1176471	0.0888889	0.173913	0.1176471
User_007	0.15	0.2142857	0.1764706	0.0888889	0.1428571	0.109589	0.0888889	0.1428571	0.109589
User_009	0	0	0	0	0	0	0	0	0
User_011	0.125	0.2380952	0.1639344	0.1111111	0.2380952	0.1515152	0.1555556	0.3333333	0.2121212
User_013	0	0	0	0.0222222	0.5	0.0425532	0	0	0
User_016	0.025	0.1428571	0.0425532	0.0222222	0.1428571	0.0384615	0.0444444	0.2857143	0.0769231
User_018	0.025	0.04	0.0307692	0.0444444	0.08	0.0571429	0.0666667	0.12	0.0857143
User_019	0.025	0.25	0.0454545	0.0222222	0.25	0.0408163	0.0222222	0.25	0.0408163
User_020	0.075	0.25	0.1153846	0.0444444	0.1666667	0.0701754	0.0222222	0.0833333	0.0350877
User_021	0	0	0	0.0222222	0.1	0.0363636	0.0222222	0.1	0.0363636
User_022	0.1	0.2666667	0.1454545	0.0888889	0.2666667	0.1333333	0.0888889	0.2666667	0.1333333
User_023	0	0	0	0	0	0	0	0	0
User_025	0.025	0.2	0.0444444	0.0444444	0.4	0.08	0.0444444	0.4	0.08
User_027	0	0	0	0.0222222	0.25	0.0408163	0	0	0
User_028	0.025	0.1666667	0.0434783	0.0222222	0.1666667	0.0392157	0	0	0
User_029	0	0	0	0.0222222	0.037037	0.0277778	0.0444444	0.0740741	0.0555556
User_030	0	0	0	0.0444444	0.0588235	0.0506329	0.0444444	0.0588235	0.0506329

表 10 實驗 4.3.1，平均準確率及召回率

Algorithm	Precision	Recall
ITCC	0.057142857	0.168635042
MSRICC	0.047619047	0.197620237
MSRIICC	0.048677249	0.164559459

4.3.2 以使用者使用應用程式的次數做推薦(Count)

我們以同群使用者，使用人數最多的 App 為基準，推薦前 5 名的應用程式給使用者，得到了以下結果(表 11)。以 User_002 來說，準確率 MSRICC 方法最高，約 0.002，MSRIICC 最低，召回率也是 MSRICC 方法最高，約 0.008，MSRIICC 最低。以 User_011 來說，準確率以 MSRICC 方法最高，約 0.002，MSRIICC 的 0.0017 最低，召回率也是 MSRICC 方法最高，約 0.012，MSRIICC 的 0.011 最低。而 User_022，準確率以 ITCC 方法最高，約 0.002，MSRIICC 的方法較低約 0.001，召回率也是 ITCC 方法較高，約 0.014，以 MSRIICC 的方法較低約 0.011。另外，我們也計算三種演算法的平均準確率與召回率，如表 12。

國立政治大學 National Chengchi University

表 11 實驗 4.3.2，以使用者使用應用程式的次數資料，推薦使用人數最多的 App，準確率、召回率及 F-measure 比較表

Algorithm	ITCC			MSRICC			MSRIICC		
User	Precision	Recall	F-measure	Precision	Recall	F-measure	Precision	Recall	F-measure
User_002	0.05	0.2258065	0.0818713	0.0518519	0.2258065	0.0843373	0.0482759	0.2258065	0.0795455
User_003	0.0785714	0.4583333	0.1341463	0.0814815	0.4583333	0.1383648	0.0758621	0.4583333	0.1301775
User_004	0.0285714	0.3636364	0.0529801	0.0296296	0.3636364	0.0547945	0.0344828	0.4545455	0.0641026
User_005	0.0214286	1	0.041958	0.0222222	1	0.0434783	0.0206897	1	0.0405405
User_006	0.0285714	0.173913	0.0490798	0.0296296	0.173913	0.0506329	0.0275862	0.173913	0.047619
User_007	0.0714286	0.3571429	0.1190476	0.0666667	0.3214286	0.1104294	0.0482759	0.25	0.0809249
User_009	0.0071429	1	0.0141844	0	0	0	0.0068966	1	0.0136986
User_011	0.05	0.3333333	0.0869565	0.0518519	0.3333333	0.0897436	0.0482759	0.3333333	0.0843373
User_013	0	0	0	0	0	0	0	0	0
User_016	0.0142857	0.2857143	0.0272109	0.0148148	0.2857143	0.028169	0.0137931	0.2857143	0.0263158
User_018	0.0142857	0.08	0.0242424	0.0296296	0.16	0.05	0.0137931	0.08	0.0235294
User_019	0.0071429	0.25	0.0138889	0.0074074	0.25	0.0143885	0.0068966	0.25	0.0134228
User_020	0.0285714	0.3333333	0.0526316	0.0148148	0.1666667	0.0272109	0.0137931	0.1666667	0.0254777
User_021	0.0142857	0.2	0.0266667	0.0222222	0.3	0.0413793	0.0137931	0.2	0.0258065
User_022	0.0428571	0.4	0.0774194	0.037037	0.3333333	0.0666667	0.0344828	0.3333333	0.0625
User_023	0	0	0	0	0	0	0	0	0
User_025	0.0214286	0.6	0.0413793	0.0222222	0.6	0.0428571	0.0206897	0.6	0.04
User_027	0	0	0	0.0074074	0.25	0.0143885	0.0068966	0.25	0.0134228
User_028	0.0142857	0.3333333	0.0273973	0.0148148	0.3333333	0.0283688	0.0137931	0.3333333	0.0264901
User_029	0.0142857	0.0740741	0.0239521	0.0222222	0.1111111	0.037037	0.0206897	0.1111111	0.0348837
User_030	0.0214286	0.0882353	0.0344828	0.0222222	0.0882353	0.035503	0.0275862	0.1176471	0.0446927

表 12 實驗 4.3.2，推薦使用人數最多，App 的平均準確率及召回率

Algorithm	Precision	Recall
ITCC	0.025170068	0.312231224
MSRICC	0.026102293	0.274040244
MSRIICC	0.02364532	0.315416067

我們以同群使用者，使用次數最高的 App 為基準，推薦前 5 名的應用程式給使用者，得到了以下結果(表 13)。以 User_002 來說，準確率 ITCC 方法最高，約 0.0008，MSRICC 的 0.0005 最低，召回率也是 ITCC 方法最高，約 0.0035，MSRICC 的 0.0024 最低。以 User_011 來說，準確率以 ITCC 方法最高，約 0.0013，MSRICC 的 0.0011 最低，召回率也是 ITCC 方法最高，約 0.0085，MSRICC 的 0.0071 最低。而 User_022，準確率以 MSRICC 方法最高，約 0.0014，MSRIICC 的方法較低約 0.0010，召回率也是 MSRICC 方法較高，約 0.0123，以 MSRIICC 的方法較低約 0.0092。另外，我們也計算三種演算法的平均準確率與召回率，如表 14。

表 13 實驗 4.3.2，以使用者使用應用程式的次數資料，推薦使用次數最高的 App，準確率、召回率及 F-measure 比較表

Algorithm	**ITCC**			**MSRICC**			**MSRIICC**		
User	**Precision**	**Recall**	**F-measure**	**Precision**	**Recall**	**F-measure**	**Precision**	**Recall**	**F-measure**
User_002	0.0214286	0.0967742	0.0350877	0.0148148	0.0645161	0.0240964	0.0206897	0.0967742	0.0340909
User_003	0.0642857	0.375	0.1097561	0.0518519	0.2916667	0.0880503	0.0344828	0.2083333	0.0591716
User_004	0.0285714	0.3636364	0.0529801	0.0296296	0.3636364	0.0547945	0.0275862	0.3636364	0.0512821
User_005	0.0142857	0.6666667	0.027972	0.0222222	1	0.0434783	0.0206897	1	0.0405405
User_006	0.0142857	0.0869565	0.0245399	0.0222222	0.1304348	0.0379747	0.0206897	0.1304348	0.0357143
User_007	0.0428571	0.2142857	0.0714286	0.0444444	0.2142857	0.0736196	0.0344828	0.1785714	0.0578035
User_009	0	0	0	0	0	0	0	0	0
User_011	0.0357143	0.2380952	0.0621118	0.0296296	0.1904762	0.0512821	0.0344828	0.2380952	0.060241
User_013	0	0	0	0	0	0	0	0	0
User_016	0.0071429	0.1428571	0.0136054	0.0074074	0.1428571	0.0140845	0.0068966	0.1428571	0.0131579
User_018	0.0142857	0.08	0.0242424	0.0222222	0.12	0.0375	0.0137931	0.08	0.0235294
User_019	0.0071429	0.25	0.0138889	0.0074074	0.25	0.0143885	0.0068966	0.25	0.0134228
User_020	0.0071429	0.0833333	0.0131579	0.0074074	0.0833333	0.0136054	0.0068966	0.0833333	0.0127389
User_021	0.0142857	0.2	0.0266667	0.0074074	0.1	0.0137931	0.0068966	0.1	0.0129032
User_022	0.0357143	0.3333333	0.0645161	0.037037	0.3333333	0.0666667	0.0275862	0.2666667	0.05
User_023	0	0	0	0	0	0	0	0	0
User_025	0.0071429	0.2	0.0137931	0.0148148	0.4	0.0285714	0.0068966	0.2	0.0133333
User_027	0	0	0	0	0	0	0	0	0
User_028	0.0142857	0.3333333	0.0273973	0.0074074	0.1666667	0.0141844	0.0137931	0.3333333	0.0264901
User_029	0.0071429	0.037037	0.011976	0.0148148	0.0740741	0.0246914	0.0137931	0.0740741	0.0232558
User_030	0.0142857	0.0588235	0.0229885	0.0148148	0.0588235	0.0236686	0.0206897	0.0882353	0.0335196

表 14 實驗 4.3.2，推薦使用次數最多，App 的平均準確率及召回率

Algorithm	Precision	Recall
ITCC	0.016666667	0.179053924
MSRICC	0.016931217	0.189719235
MSRIICC	0.015106732	0.182587866

4.3.3 以使用者使用應用程式的頻率做推薦(Frequency)

我們以同群使用者，使用人數最多的 App 為基準，推薦前 5 名的應用程式給使用者，得到了以下結果(表 15)。以 User_002 來說，準確率 MSRIICC 方法最高，約 0.0036，ITCC 的 0.0014 最低，召回率也是 MSRIICC 方法最高，約 0.0074，ITCC 的 0.0027 最低。以 User_011 來說，準確率以 MSRICC 方法最高，約 0.0036，MSRIICC 的 0.0024 最低，召回率也是 MSRICC 方法最高，約 0.011，MSRIICC 的 0.0073 最低。而 User_022，準確率以 MSRICC 和 MSRIICC 方法最高，約 0.0047，ITCC 的方法較低約 0.0042，召回率也是 MSRICC 和 MSRIICC 方法較高，約 0.0205，以 ITCC 的方法較低約 0.0167。另外，我們也計算三種演算法的平均準確率與召回率，如表 16。

國立政治大學 National Chengchi University

表 15 實驗 4.3.3，以使用者使用應用程式的頻率資料，推薦使用人數最多的 App，準確率、召回率及 F-measure 比較表

Algorithm	ITCC			MSRICC			MSRIICC		
User	Precision	Recall	F-measure	Precision	Recall	F-measure	Precision	Recall	F-measure
User_002	0.0166667	0.0322581	0.021978	0.0307692	0.0645161	0.0416667	0.0461538	0.0967742	0.0625
User_003	0.0666667	0.1666667	0.0952381	0.0615385	0.1666667	0.0898876	0.0615385	0.1666667	0.0898876
User_004	0.0166667	0.0909091	0.028169	0.0307692	0.1818182	0.0526316	0.0307692	0.1818182	0.0526316
User_005	0.0333333	0.6666667	0.0634921	0.0307692	0.6666667	0.0588235	0.0153846	0.3333333	0.0294118
User_006	0.0166667	0.0434783	0.0240964	0.0307692	0.0869565	0.0454545	0.0307692	0.0869565	0.0454545
User_007	0.05	0.1071429	0.0681818	0.0461538	0.1071429	0.0645161	0.0307692	0.0714286	0.0430108
User_009	0	0	0	0	0	0	0	0	0
User_011	0.0333333	0.0952381	0.0493827	0.0461538	0.1428571	0.0697674	0.0307692	0.0952381	0.0465116
User_013	0	0	0	0	0	0	0	0	0
User_016	0	0	0	0	0	0	0.0153846	0.1428571	0.0277778
User_018	0.0166667	0.04	0.0235294	0.0307692	0.08	0.0444444	0.0307692	0.08	0.0444444
User_019	0.0166667	0.25	0.03125	0.0153846	0.25	0.0289855	0.0153846	0.25	0.0289855
User_020	0.0166667	0.0833333	0.0277778	0.0307692	0.1666667	0.0519481	0.0307692	0.1666667	0.0519481
User_021	0.0166667	0.1	0.0285714	0.0153846	0.1	0.0266667	0	0	0
User_022	0.05	0.2	0.08	0.0615385	0.2666667	0.1	0.0615385	0.2666667	0.1
User_023	0	0	0	0	0	0	0	0	0
User_025	0	0	0	0	0	0	0.0153846	0.2	0.0285714
User_027	0	0	0	0	0	0	0.0153846	0.25	0.0289855
User_028	0	0	0	0	0	0	0	0	0
User_029	0	0	0	0.0153846	0.037037	0.0217391	0.0153846	0.037037	0.0217391
User_030	0	0	0	0.0307692	0.0588235	0.040404	0.0153846	0.0294118	0.020202

表 16 實驗 4.3.3，推薦使用人數最多的 App，平均準確率及召回率

Algorithm	Precision	Recall
ITCC	0.016666667	0.089318716
MSRICC	0.022710623	0.113134194
MSRIICC	0.021978022	0.11689785

我們以同群使用者，使用頻率最高的 App 為基準，推薦前 5 名的應用程式給使用者，得到了以下結果(表 17)。以 User_002 來說，準確率 MSRIICC 方法最高，約 0.0036，ITCC 的 0.0014 最低，召回率也是 MSRIICC 方法最高，約 0.0074，ITCC 的 0.0027 最低。以 User_011 來說，準確率和召回率在 MSRICC 和 MSRIICC 方法一樣，分別約 0.0036 和 0.011，ITCC 為 0 最低。而 User_022，準確率以 MSRICC 和 MSRIICC 方法最高，約 0.0047，ITCC 的方法較低約 0.0042，召回率也是 MSRICC 和 MSRIICC 方法較高，約 0.0205，以 ITCC 的方法較低約 0.0167。另外，我們也計算三種演算法的平均準確率與召回率，如表 18。

表 17 實驗 4.3.3，以使用者使用應用程式的頻率資料，推薦使用頻率最高的 App，準確率、召回率及 F-measure 比較表

Algorithm	**ITCC**			**MSRICC**			**MSRIICC**		
User	**Precision**	**Recall**	**F-measure**	**Precision**	**Recall**	**F-measure**	**Precision**	**Recall**	**F-measure**
User_002	0.0166667	0.0322581	0.021978	0.0307692	0.0645161	0.0416667	0.0461538	0.0967742	0.0625
User_003	0.05	0.125	0.0714286	0.0461538	0.125	0.0674157	0.0307692	0.0833333	0.0449438
User_004	0.0166667	0.0909091	0.028169	0.0153846	0.0909091	0.0263158	0.0307692	0.1818182	0.0526316
User_005	0.0333333	0.6666667	0.0634921	0.0307692	0.6666667	0.0588235	0.0153846	0.3333333	0.0294118
User_006	0.0166667	0.0434783	0.0240964	0.0307692	0.0869565	0.0454545	0.0307692	0.0869565	0.0454545
User_007	0.05	0.1071429	0.0681818	0.0461538	0.1071429	0.0645161	0.0307692	0.0714286	0.0430108
User_009	0	0	0	0	0	0	0	0	0
User_011	0.0333333	0.0909091	0.0487805	0.0461538	0.1428571	0.0697674	0.0461538	0.1428571	0.0697674
User_013	0	0	0	0	0	0	0	0	0
User_016	0	0	0	0	0	0	0.0153846	0.1428571	0.0277778
User_018	0.0166667	0.04	0.0235294	0.0307692	0.08	0.0444444	0.0307692	0.08	0.0444444
User_019	0.0166667	0.5	0.0322581	0.0153846	0.25	0.0289855	0.0153846	0.25	0.0289855
User_020	0.0166667	0.0769231	0.0273973	0.0307692	0.1666667	0.0519481	0.0307692	0.1666667	0.0519481
User_021	0.0166667	0.0909091	0.028169	0.0153846	0.1	0.0266667	0	0	0
User_022	0.05	0.1666667	0.0769231	0.0615385	0.2666667	0.1	0.0615385	0.2666667	0.1
User_023	0	0	0	0	0	0	0	0	0
User_025	0	0	0	0	0	0	0.0153846	0.2	0.0285714
User_027	0	0	0	0	0	0	0.0153846	0.25	0.0289855
User_028	0	0	0	0	0	0	0	0	0
User_029	0	0	0	0.0153846	0.037037	0.0217391	0.0153846	0.037037	0.0217391
User_030	0	0	0	0.0307692	0.0588235	0.040404	0.0153846	0.0294118	0.020202

表 18 實驗 4.3.3，推薦使用頻率最高的 App，平均準確率及召回率

Algorithm	Precision	Recall
ITCC	0.015873016	0.096707756
MSRICC	0.021245421	0.106821062
MSRIICC	0.021245421	0.115197169

4.3.4 小結

以同群使用者，使用人數最多的 App 為基準做推薦，我們發現 ITCC、MSRICC、MSRIICC 三種演算法的表現，都以資料集一“是否使用應用程式”的準確率最高，其次是資料集三“使用應用程式頻率”次高，而資料集二“使用應用程式次數”的準確率最低。在召回率的部分，由實驗發現以資料集一“是否使用應用程式”的召回率最高，其次是資料集二“使用應用程式次數”次高，而資料集三“使用應用程式頻率”的召回率最低。

以同群使用者，使用次數/頻率最高的 App 為基準做推薦，我們發現 ITCC、MSRICC、MSRIICC 三種演算法的表現，以資料集一“是否使用應用程式”的準確率最高，其次是資料集三“使用應用程式頻率”次高，而資料集二“使用應用程式次數”的準確率最低。在召回率的部分，ITCC 以資料集一“是否使用應用程式”的召回率最高，其次是資料集二“使用應用程式次數”次高，而資料集三“使用應用程式頻率”的召回率最低；MSRICC 與以資料集一“是否使用應用程式”的召回率最高，其次是資料集三“使用應用程式頻率”次高，而資料集二“使用應用程式次數”的召回率最低。

從 4.3 中，我們發現兩種推薦方法，在 4.3.1(YN)資料集一的實驗中，三種演算法都有較好的準確率及召回率。

在表 10、12、14、16、18 中，我們分別列出各實驗的平均值，從這些實際資料來看，或許準確率和召回率不高，但我們認為，若考慮目前 Google Play 上有超過 100 萬個應用程式，在資料集中 1125 個應用程式中，能夠有 1 個以上的應用程式正確被推薦，，應該算是不錯的表現。

在 User_009 的部分，根據表 8 的統計，此使用者在後段時間只有多使用一個應用程式，實驗結果顯示，在我們實驗的推薦名單中，三種演算法都沒有將這一個應用程式推薦給他，造成準確率及召回率是 0 的狀況。另一位使用者 User_023，在三種演算法的準確率及召回率也都是 0，可能在此資料集沒有使用者跟他的操作紀錄類似，造成分群和推薦結果不好。

從此實驗中，我們可以看到，若在資料集中沒有相似的使用者或應用程式，本研究所設計的實驗並不能解決推薦系統常有的冷啟動(Cold Start)的問題，造成準確率或召回率是 0 的狀況發生。

在各個資料集中，若未使用 Co-Clustering 作分類，直接以應用程式的排名來做推薦，計算的準確率、召回率及 F 度量如下表：

表 19 以最多人使用的應用程式排名，使用是否使用應用程式資料集中做推薦

User	Precision	Recall	F-measure
User_002	0.044444444	0.064516129	0.052631579
User_003	0.044444444	0.083333333	0.057971014
User_004	0.022222222	0.090909091	0.035714286
User_005	0.044444444	0.666666667	0.083333333
User_006	0.022222222	0.043478261	0.029411765
User_007	0.088888889	0.142857143	0.109589041
User_009	0.022222222	1	0.043478261
User_011	0.044444444	0.095238095	0.060606061
User_013	0	0	0
User_016	0	0	0
User_018	0.022222222	0.04	0.028571429
User_019	0.022222222	0.25	0.040816327
User_020	0	0	0
User_021	0	0	0
User_022	0.044444444	0.133333333	0.066666667
User_023	0	0	0
User_025	0	0	0
User_027	0	0	0
User_028	0	0	0
User_029	0	0	0
User_030	0	0	0

表 20 以最多使用次數的應用程式排名，以使用應用程式次數資料集做推薦

User	Precision	Recall	F-measure
User_002	0.014285714	0.064516129	0.023391813
User_003	0.042857143	0.25	0.073170732
User_004	0.021428571	0.272727273	0.039735099
User_005	0.021428571	1	0.041958042
User_006	0.007142857	0.043478261	0.012269939
User_007	0.035714286	0.178571429	0.05952381
User_009	0.007142857	1	0.014184397
User_011	0.035714286	0.238095238	0.062111801
User_013	0	0	0
User_016	0	0	0
User_018	0.014285714	0.08	0.024242424
User_019	0.007142857	0.25	0.013888889
User_020	0.007142857	0.083333333	0.013157895
User_021	0.014285714	0.2	0.026666667
User_022	0.035714286	0.333333333	0.064516129
User_023	0	0	0
User_025	0.007142857	0.2	0.013793103
User_027	0	0	0
User_028	0	0	0
User_029	0.007142857	0.037037037	0.011976048
User_030	0.014285714	0.058823529	0.022988506

國立政治大學 National Chengchi University

表 21 以使用頻率最高的應用程式排名，以使用應用程式頻率資料集做推薦

User	Precision	Recall	F-measure
User_002	0	0	0
User_003	0.046153846	0.001923077	0.003692308
User_004	0.030769231	0.002797203	0.005128205
User_005	0.015384615	0.005128205	0.007692308
User_006	0	0	0
User_007	0.030769231	0.001098901	0.002122016
User_009	0	0	0
User_011	0.046153846	0.002197802	0.004195804
User_013	0	0	0
User_016	0	0	0
User_018	0	0	0
User_019	0.015384615	0.003846154	0.006153846
User_020	0.015384615	0.001282051	0.002366864
User_021	0.015384615	0.001538462	0.002797203
User_022	0.046153846	0.003076923	0.005769231
User_023	0	0	0
User_025	0	0	0
User_027	0	0	0
User_028	0	0	0
User_029	0.015384615	0.000569801	0.001098901
User_030	0	0	0

另外，我們將使用者人數做調整，減少為 10 人，經實驗證明結果，若是使用者人數減少，會影響推薦的結果，使準確率降低。準確率、召回率及 F 度量如下表：

國立政治大學 National Chengchi University

表 22 以最多人使用的應用程式排名，使用是否使用應用程式資料集中做推薦

Algorithm	ITCC			MSRICC			MSRIICC		
User	**Precision**	**Recall**	**F-measure**	**Precision**	**Recall**	**F-measure**	**Precision**	**Recall**	**F-measure**
User_002	0.03076923	0.06451613	0.04166667	0.03076923	0.06451613	0.04166667	0.13846154	0.29032258	0.1875
User _003	0.04615385	0.125	0.06741573	0.04615385	0.125	0.06741573	0.06153846	0.16666667	0.08988764
User _004	0.04615385	0.27272727	0.07894737	0.04615385	0.27272727	0.07894737	0.01538462	0.09090909	0.02631579
User _005	0	0	0	0.03030303	0.5	0.05714286	0.03076923	0.66666667	0.05882353
User _006	0.01538462	0.04347826	0.02272727	0	0	0	0.03076923	0.08695652	0.04545455
User _007	0.03076923	0.07142857	0.04301075	0.04615385	0.10714286	0.06451613	0.04615385	0.10714286	0.06451613
User _009	0	0	0	0	0	0	0	0	0
User _011	0.04615385	0.14285714	0.06976744	0	0	0	0.06153846	0.19047619	0.09302326
User _013	0	0	0	0	0	0	0	0	0

表 23 以最多人使用的應用程式排名，以使用應用程式次數資料集做推薦

Algorithm	ITCC			MSRICC			MSRIICC		
User	**Precision**	**Recall**	**F-measure**	**Precision**	**Recall**	**F-measure**	**Precision**	**Recall**	**F-measure**
User _002	0.03076923	0.06451613	0.04166667	0.04615385	0.09677419	0.0625	0.01538462	0.03225807	0.02083333
User _003	0.04615385	0.125	0.06741573	0.03076923	0.08333333	0.04494382	0.01538462	0.04166667	0.02247191
User _004	0.01538462	0.09090909	0.02631579	0.01538462	0.09090909	0.02631579	0.01538462	0.09090909	0.02631579
User _005	0	0	0	0	0	0	0.01538462	0.33333333	0.02941177
User _006	0.03076923	0.08695652	0.04545455	0.03076923	0.08695652	0.04545455	0.01538462	0.04347826	0.02272727
User _007	0.03076923	0.07142857	0.04301075	0.07692308	0.17857143	0.10752688	0.04615385	0.10714286	0.06451613
User _009	0	0	0	0.01538462	1	0.03030303	0.01538462	1	0.03030303
User _011	0.03076923	0.0952381	0.04651163	0.03076923	0.0952381	0.04651163	0.06153846	0.19047619	0.09302326
User _013	0.01538462	0.5	0.02985075	0	0	0	0	0	0

表 24 以最多使用次數的應用程式排名，以使用應用程式次數資料集做推薦

Algorithm	ITCC			MSRICC			MSRIICC		
User	Precision	Recall	F-measure	Precision	Recall	F-measure	Precision	Recall	F-measure
User _002	0.03076923	0.06451613	0.04166667	0.01538462	0.03225807	0.02083333	0.01538462	0.03225807	0.02083333
User _003	0.01538462	0.04166667	0.02247191	0.03076923	0.08333333	0.04494382	0.01538462	0.04166667	0.02247191
User _004	0	0	0	0	0	0	0.01538462	0.09090909	0.02631579
User _005	0	0	0	0	0	0	0	0	0
User _006	0.03076923	0.08695652	0.04545455	0.03076923	0.08695652	0.04545455	0.01538462	0.04347826	0.02272727
User _007	0.03076923	0.07142857	0.04301075	0.01538462	0.03571429	0.02150538	0.03076923	0.07142857	0.04301075
User _009	0	0	0	0.01538462	1	0.03030303	0	0	0
User _011	0.03076923	0.0952381	0.04651163	0.03076923	0.0952381	0.04651163	0.04615385	0.14285714	0.06976744
User _013	0	0	0	0	0	0	0	0	0

表 25 以最多人使用的應用程式排名，以使用應用程式頻率資料集做推薦

Algorithm	ITCC			MSRICC			MSRIICC		
User	Precision	Recall	F-measure	Precision	Recall	F-measure	Precision	Recall	F-measure
User _002	0.03076923	0.06451613	0.04166667	0.03076923	0.06451613	0.04166667	0.03076923	0.06451613	0.04166667
User _003	0.01538462	0.04166667	0.02247191	0.03076923	0.08333333	0.04494382	0.01538462	0.04166667	0.02247191
User _004	0	0	0	0.01538462	0.09090909	0.02631579	0.01538462	0.09090909	0.02631579
User _005	0	0	0	0	0	0	0	0	0
User _006	0.03076923	0.08695652	0.04545455	0.03076923	0.08695652	0.04545455	0.03076923	0.08695652	0.04545455
User _007	0.03076923	0.07142857	0.04301075	0.03076923	0.07142857	0.04301075	0.04615385	0.10714286	0.06451613
User _009	0	0	0	0	0	0	0	0	0
User _011	0.03076923	0.0952381	0.04651163	0.03076923	0.0952381	0.04651163	0.03076923	0.0952381	0.04651163
User _013	0	0	0	0.01538462	0.5	0.02985075	0.01538462	0.5	0.02985075

國立政治大學 National Chengchi University

表 26 以使用頻率最高的應用程式排名，以使用應用程式頻率資料集做推薦

Algorithm	**ITCC**			**MSRICC**			**MSRIICC**		
User	**Precision**	**Recall**	**F-measure**	**Precision**	**Recall**	**F-measure**	**Precision**	**Recall**	**F-measure**
User _002	0.03076923	0.06451613	0.04166667	0.03076923	0.06451613	0.04166667	0.03076923	0.06451613	0.04166667
User _003	0.01538462	0.04166667	0.02247191	0.01538462	0.04166667	0.02247191	0.01538462	0.04166667	0.02247191
User _004	0	0	0	0.01538462	0.09090909	0	0.01538462	0.09090909	0
User _005	0	0	0	0	0	0	0	0	0
User _006	0.03076923	0.08695652	0.04545455	0.03076923	0.08695652	0.04545455	0.03076923	0.08695652	0.04545455
User _007	0.01538462	0.03571429	0.02150538	0.03076923	0.07142857	0.04301075	0.03076923	0.07142857	0.04301075
User _009	0	0	0	0	0	0	0	0	0
User _011	0.03076923	0.0952381	0.04651163	0.03076923	0.0952381	0.04651163	0.03076923	0.0952381	0.04651163
User _013	0	0	0	0.01538462	0.5	0	0	0	0

將我們的實驗與應用程式排名推薦的結果，比較兩者的平均準確率。我們發現，在實驗中使用 Co-Clustering 分群結果來做推薦，其準確率的平均值較高，所以此方法比單純使用排行榜推薦的方法，更能正確找出使用者在後段時間有使用到的應用程式。

國立政治大學 National Chengchi University

表 27 準確率平均值比較表

Dataset	Algorithm	Method	Precision
資料集一	ITCC	最多人使用	0.057142857
	MSRICC		0.047619047
	MSRIICC		0.048677249
	排行榜		0.02010582
資料集二	ITCC	最多人使用	0.025170068
	MSRICC		0.026102293
	MSRIICC		0.02364532
	排行榜		0.02010582
	ITCC	使用次數最高	0.016666667
	MSRICC		0.016931217
	MSRIICC		0.015106732
	排行榜		0.013945578
資料集三	ITCC	最多人使用	0.016666667
	MSRICC		0.022710623
	MSRIICC		0.021978022
	排行榜		0.02010582
	ITCC	使用頻率最高	0.015873016
	MSRICC		0.021245421
	MSRIICC		0.021245421
	排行榜		0.013186813

國立政治大學 National Chengchi University

4.4 實驗結果(二)

研究發現 1125 個應用程式使用各演算法做分類的結果，和 Google Play 上的分類結果不盡相同。本實驗從 User 使用應用程式的相似性，探討應用程式在 Co-Clustering 做分類的結果。下表 28 為各演算法在三個資料集的應用程式分群數量。

表 28 應用程式群集數量

Algorithm	是否使用	使用次數	使用頻率
ITCC	8	26	12
MSRICC	7	28	13
MSRIICC	7	25	11

在我們所使用的資料中，列出四個與 Sudoku 相關的 App，以及六個與旅遊和交通相關的 App。我們分別觀察這些 App 在各實驗的分類狀況。

表 29 Sudoku 相關應用程式資料

ID	Name	Android Market Category
App0096	Sudoku	Brain & Puzzle
App0170	Sudoku Free	Brain & Puzzle
App0355	Sudoku Shapes	Brain & Puzzle
App0802	Super Sudoku	Cards & Casino

表 30 旅遊及交通相關應用程式

ID	Name	Android Market Category
App0288	Taipei Bus Situation Alert	Transportation
App0388	Street View on Google Maps	Travel & Local
App0536	GPS Status & Toolbox	Travel & Local
App0700	TaipeiMetro	Travel & Local
App0968	RMaps	Travel & Local
App1021	Taiwan Highway CCTV FREE	Travel & Local

4.4.1 以使用者是否使用應用程式做應用程式分群(YN)

表 31 為應用程式使用各演算法分群後，每個 Cluster 中應用程式的數量。三種演算法中皆有一個較大數量的群集，分別是 ITCC 的第 6 個群集，有 494 個應用程式；MSRICC 的第 6 個群集，有 578 個應用程式；以及 MSRIICC 的第 5 個群集，分別都有 566 個應用程式。

表 31 各群集中，應用程式數量

ITCC	**App Cluster**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
	Number	295	43	86	40	31	494	22	114
MSRICC	**App Cluster**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	
	Number	274	61	48	34	96	578	34	
MSRIICC	**App Cluster**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	
	Number	268	60	47	128	566	22	34	

在 Sudoku 相關的應用程式方面，使用 MSRICC 和 MSRIICC 方法的情況下，將這四個 App 分成了兩類，而且沒有和 Google Play 所做的 Android Market Category 分類相同；使用 ITCC 方法則是將四個 App 分成了四類，由此結果可得知，App0096、App0170、App0802 三個應用程式在這個資料集中，被使用者操作的行為是類似的。分類結果詳見表 32。

表 32 各演算法中，四個 Sudoku 相關 App 的分類情形

Algorithm	**ID**	**Name**	**Android Market Category**	**App Cluster**
ITCC	App0096	Sudoku	Brain & Puzzle	6
	App0170	Sudoku Free	Brain & Puzzle	1
	App0355	Sudoku Shapes	Brain & Puzzle	3
	App0802	Super Sudoku	Cards & Casino	2
MSRICC	App0096	Sudoku	Brain & Puzzle	6
	App0170	Sudoku Free	Brain & Puzzle	6
	App0355	Sudoku Shapes	Brain & Puzzle	2
	App0802	Super Sudoku	Cards & Casino	6
MSRIICC	App0096	Sudoku	Brain & Puzzle	5
	App0170	Sudoku Free	Brain & Puzzle	5
	App0355	Sudoku Shapes	Brain & Puzzle	2
	App0802	Super Sudoku	Cards & Casino	5

在旅遊及交通相關的應用程式方面，使用 ITCC 方法的情況下，將這六個 App 分成了四類；使用 MSRIICC 和 MSRIICC 方法則是將六個 App 分成了三類，而且沒有和 Google

Play 所做的 Android Market Category 分類相同，由此結果可得知， App0388、App0700、App0968、App1021 四個應用程式在這個資料集中，被使用者操作的行為是類似的。分類結果詳見表 33。

表 33 各演算法中，五個旅遊與交通相關 App 的分類情形

Algorithm	ID	Name	Android Market Category	App Cluster
ITCC	App0288	Taipei Bus Situation Alert	Transportation	6
	App0388	Street View on Google Maps	Travel & Local	8
	App0536	GPS Status & Toolbox	Travel & Local	3
	App0700	TaipeiMetro	Travel & Local	1
	App0968	RMaps	Travel & Local	6
	App1021	Taiwan Highway CCTV FREE	Travel & Local	1
MSRICC	App0288	Taipei Bus Situation Alert	Transportation	1
	App0388	Street View on Google Maps	Travel & Local	6
	App0536	GPS Status & Toolbox	Travel & Local	5
	App0700	TaipeiMetro	Travel & Local	6
	App0968	RMaps	Travel & Local	6
	App1021	Taiwan Highway CCTV FREE	Travel & Local	6
MSRIICC	App0288	Taipei Bus Situation Alert	Transportation	1
	App0388	Street View on Google Maps	Travel & Local	5
	App0536	GPS Status & Toolbox	Travel & Local	4
	App0700	TaipeiMetro	Travel & Local	5
	App0968	RMaps	Travel & Local	5
	App1021	Taiwan Highway CCTV FREE	Travel & Local	5

國立政治大學 National Chengchi University

4.4.2 以使用者使用應用程式的次數做應用程式分群(Count)

表 34 為應用程式使用各演算法分群後，每個 Cluster 中應用程式的數量。三種演算法中皆有一個較大數量的群集，分別是 ITCC 的第 2 個群集，有 607 個應用程式；MSRICC 的第 7 個群集，有 587 個應用程式；以及 MSRIICC 的第 22 個群集，分別都有 564 個應用程式。

表 34 各群集中，應用程式數量

ITCC	**App Cluster**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**		
	Number	29	607	56	17	10	4	10	11	13	163	59	13	10	6	7	9	10	32	5	7	5	10	11	2	12	7		
MSRICC	**App Cluster**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**
	Number	6	11	5	14	110	4	587	5	8	12	20	20	7	7	3	113	4	16	15	41	7	14	17	6	7	48	11	7
MSRIICC	**App Cluster**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**			
	Number	14	14	27	12	8	11	170	13	18	7	9	12	2	53	7	10	6	5	45	5	7	564	32	64	10			

在 Sudoku 相關的應用程式方面，使用 MSRICC 方法的情況下，將這四個 App 分成了兩類；使用 ITCC 和 MSRIICC 方法則是將四個 App 分成了三類，而且沒有和 Google Play 所做的 Android Market Category 分類相同，由此結果可得知， App0170、App0355 兩個應用程式在這個資料集中，被使用者操作的行為是類似的。分類結果詳見表 35。

表 35 各演算法中，四個 Sudoku 相關 App 的分類情形

Algorithm	**ID**	**Name**	**Android Market Category**	**App Cluster**
ITCC	App0096	Sudoku	Brain & Puzzle	11
	App0170	Sudoku Free	Brain & Puzzle	2
	App0355	Sudoku Shapes	Brain & Puzzle	2
	App0802	Super Sudoku	Cards & Casino	10
MSRICC	App0096	Sudoku	Brain & Puzzle	20
	App0170	Sudoku Free	Brain & Puzzle	7
	App0355	Sudoku Shapes	Brain & Puzzle	7
	App0802	Super Sudoku	Cards & Casino	7
MSRIICC	App0096	Sudoku	Brain & Puzzle	14
	App0170	Sudoku Free	Brain & Puzzle	22
	App0355	Sudoku Shapes	Brain & Puzzle	22
	App0802	Super Sudoku	Cards & Casino	7

在旅遊及交通相關的應用程式方面，使用 MSRICC 方法的情況下，將這六個 App 分成了三類；使用 ITCC 和 MSRIICC 方法則是將六個 App 分成了兩類，而且沒有和 Google

Play 所做的 Android Market Category 分類相同，由此結果可得知， App0288、App0700、App0968 三個應用程式在這個資料集中，被使用者操作的行為是類似的。分類結果詳見表 36。

表 36 各演算法中，五個旅遊與交通相關 App 的分類情形

Algorithm	ID	Name	Android Market Category	App Cluster
ITCC	App0288	Taipei Bus Situation Alert	Transportation	2
	App0388	Street View on Google Maps	Travel & Local	10
	App0536	GPS Status & Toolbox	Travel & Local	10
	App0700	TaipeiMetro	Travel & Local	2
	App0968	RMaps	Travel & Local	2
	App1021	Taiwan Highway CCTV FREE	Travel & Local	10
MSRICC	App0288	Taipei Bus Situation Alert	Transportation	7
	App0388	Street View on Google Maps	Travel & Local	16
	App0536	GPS Status & Toolbox	Travel & Local	5
	App0700	TaipeiMetro	Travel & Local	7
	App0968	RMaps	Travel & Local	7
	App1021	Taiwan Highway CCTV FREE	Travel & Local	26
MSRIICC	App0288	Taipei Bus Situation Alert	Transportation	22
	App0388	Street View on Google Maps	Travel & Local	7
	App0536	GPS Status & Toolbox	Travel & Local	7
	App0700	TaipeiMetro	Travel & Local	22
	App0968	RMaps	Travel & Local	22
	App1021	Taiwan Highway CCTV FREE	Travel & Local	7

4.4.3 以使用者使用應用程式的頻率做應用程式分群(Frequency)

表 37 為應用程式使用各演算法分群後，每個 Cluster 中應用程式的數量。其中 ITCC 的第 7 個群集，MSRICC 的第 1 個群集，以及 MSRIICC 的第 2 個群集，分別都有 937 個應用程式。進一步確認後，我們發現相同的 937 個應用程式被分在同一類。

表 37 各群集中，應用程式數量

ITCC	**App Cluster**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	
	Number	2	86	21	5	4	14	937	4	11	5	29	7	
MSRICC	**App Cluster**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**
	Number	937	12	12	3	5	16	8	15	13	35	58	7	4
MSRIICC	**App Cluster**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**		
	Number	3	937	6	69	11	18	3	18	6	5	49		

在 Sudoku 相關的應用程式方面，使用這三種演算法，將這四個 App 分成了三類，其中 App0355 及 App0802 都被分在同一群集，由此結果可得知，App0355、App0802 兩個應用程式在這個資料集中，被使用者操作的行為是類似的。分類結果詳見表 38。

表 38 各演算法中，四個 Sudoku 相關 App 的分類情形

Algorithm	**ID**	**Name**	**Android Market Category**	**App Cluster**
ITCC	App0096	Sudoku	Brain & Puzzle	11
	App0170	Sudoku Free	Brain & Puzzle	2
	App0355	Sudoku Shapes	Brain & Puzzle	7
	App0802	Super Sudoku	Cards & Casino	7
MSRICC	App0096	Sudoku	Brain & Puzzle	6
	App0170	Sudoku Free	Brain & Puzzle	11
	App0355	Sudoku Shapes	Brain & Puzzle	1
	App0802	Super Sudoku	Cards & Casino	1
MSRIICC	App0096	Sudoku	Brain & Puzzle	8
	App0170	Sudoku Free	Brain & Puzzle	4
	App0355	Sudoku Shapes	Brain & Puzzle	2
	App0802	Super Sudoku	Cards & Casino	2

在旅遊及交通相關的應用程式方面，使用 ITCC、MSRICC 和 MSRIICC 三種方法，將六個 App 分成在同一群，由此結果可得知， 這六個應用程式在這個資料集中，被使用者操作的行為是類似的。分類結果詳見表 39。

表 39 各演算法中，五個旅遊與交通相關 App 的分類情形

Algorithm	ID	Name	Android Market Category	App Cluster
ITCC	App0288	Taipei Bus Situation Alert	Transportation	7
	App0388	Street View on Google Maps	Travel & Local	7
	App0536	GPS Status & Toolbox	Travel & Local	7
	App0700	TaipeiMetro	Travel & Local	7
	App0968	RMaps	Travel & Local	7
	App1021	Taiwan Highway CCTV FREE	Travel & Local	7
MSRICC	App0288	Taipei Bus Situation Alert	Transportation	1
	App0388	Street View on Google Maps	Travel & Local	1
	App0536	GPS Status & Toolbox	Travel & Local	1
	App0700	TaipeiMetro	Travel & Local	1
	App0968	RMaps	Travel & Local	1
	App1021	Taiwan Highway CCTV FREE	Travel & Local	1
MSRIICC	App0288	Taipei Bus Situation Alert	Transportation	2
	App0388	Street View on Google Maps	Travel & Local	2
	App0536	GPS Status & Toolbox	Travel & Local	2
	App0700	TaipeiMetro	Travel & Local	2
	App0968	RMaps	Travel & Local	2
	App1021	Taiwan Highway CCTV FREE	Travel & Local	2

4.4.4 小結

應用程式開發者將應用程式上架提供下載時，自行對該應用程式定義其分類，分類的方法是以個人主觀想法下，從現有的 Google Play 分類中選擇一個，或許與使用者的想法不同。從實驗中，我們以使用者在操作的相似性，利用 Co-Clustering 方法將應用程式分類，發現同樣是 Sudoku 的遊戲，在不同的資料集中，使用者操作應用程式的相似性不同，有不同的分類結果，與 Google Play 的分類有差異，或許在這部分，開發者在選擇分類時可做些調整。

第五章 結論與未來可能研究方向

5.1 結論

本研究提出的是一個推薦系統的框架，不需要還原資料矩陣，即可得到欲推薦的項目。我們先將資料分為前段時間及後段時間，以前段時間的資料為基準，後段時間的資料為驗證，依照是否使用應用程式、使用應用程式的次數，和使用應用程式的頻率等三種方式將使用者操作手機的紀錄資料整理成資料矩陣，資料矩陣內的值代表每個使用者與各應用程式的使用情形，並使用 Information Theoretic Co-Clustering Algorithm 和 Minimum Squared Residue Co-Clustering Algorithm 等 Co-Clustering 演算法，將使用者和應用程式做分群，利用分群結果，計算同群使用者中，同群的應用程式使用人數最多、使用次數最高，和使用頻率最高的應用程式，推薦給各使用者，並計算準確率及召回率做比較。

實驗結果顯示，考慮使用者是否使用應用程式的資料集，準確率以 Information Theoretic Co-Clustering Algorithm 表現最好，召回率則是 Minimum Squared Residue I Co-Clustering Algorithm 最高；考慮使用者操作使用應用程式次數的資料集，準確率及召回率在 Minimum Squared Residue I Co-Clustering Algorithm 的表現都是最好的；考慮使用者操作使用應用程式頻率的資料集，準確率以 Minimum Squared Residue II Co-Clustering Algorithm 表現最好，召回率則是 Minimum Squared Residue I Co-Clustering Algorithm 最高。

另外現有的 Google Play 分類是由應用程式開發者將應用程式上架提供下載時定義的，這樣的分類方法是以個人主觀想法做選擇，或許與使用者的看法不同。在實驗中，我們以使用者在操作手機的方式，以 Co-Clustering 方法將應用程式分類，發現同樣的遊戲，在 Google Play 的分類即有差異，或許在這部分，開發者在選擇分類時可做些調整。

5.2 未來可能研究方向

本研究提出的推薦系統的框架目前提供的演算法，未來可以加入不同的 Co-Clustering 演算法，比較各演算法的準確率及召回率。目前我們所使用的資料集為使用者操作手機應用程式的紀錄，根據此資料集，我們推薦應用程式給使用者，在這樣的架構下，我們可以套用不同的資料集，將資料集轉換成矩陣資料，即可對使用者做不同項目的推薦，保有彈性。目前我們使用了兩種推薦方式，分別是推薦同群使用者最多和使用次數或頻率最高的應用程式，未來也能加入其他的推薦方法，比較各種推薦方法的結果。未來也能將實驗使用演算法分群之後的結果，使用工具圖形化，可以觀察分群的分布。由本研究實驗所推薦的應用程式名單，驗證使用者確實有使用的部分，未來我們可以實際找使用者做評比，確認是否真的喜歡我們所推薦的應用程式。另外，未來我們也可以對使用者進行興趣指標訪談，針對每位不同使用者的興趣做推薦。在應用程式的部分，未來可以先分成一般用途和專門用途兩種，之後再分別進行推薦。

國立政治大學
National Chengchi University

參考文獻

[1] 陳昭宇，根基於自我組織特徵映射圖為基礎之最佳化演算法之推薦系統，國立中央大學碩士論文，2005。

[2] 李惠雯、林怡伶、陳怡如、陳郁琳，推薦系統之研究，吳鳳技術學院專題研究，2009。

[3] 蔡淑慧，模糊協同過濾於網路教材推薦之研究，中國文化大學碩士論文，2005。

[4] J. A. HARTIGAN, “Direct Clustering of a Data Matrix,” Journal of the American Statistical Association Volume 67, Issue 337, 1972.

[5] I. S. Dhillon. “Coclustering documents and words using Bipartite Spectral Graph Partitioning”KDD '01 Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining Pages 269-274 ACM New York, NY, USA, 2001.

[6] Wei Peng, Tao Li. “Temporal relation co-clustering on directional social network and author-topic evolution, ” Journal of Knowledge and Information Systems, March 2011, Volume 26, Issue 3, pp 467-486.

[7] I. S. Dhillon, S. Mallela, D. S. Modha. “Information Theoretic Coclustering, “KDD '03 Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining Pages 89-98 ACM New York, NY, USA, 2003.

[8] Hyuk Cho y, Inderjit S. Dhillon y, Yuqiang Guan y, Suvrit Sra. “Minimum Sum-Squared Residue Co-clustering of Gene Expression Data, ” Proceedings of the 2004 SIAM International Conference on Data Mining.

[9] T. George and S. Merugu. “A scalable collaborative filtering framework based on co-clustering.”Data Mining, Fifth IEEE International Conference, 2005.

[10] KWT Leung, DL Lee, WC Lee. “A Collaborative Location Recommendation Framework based on Co-Clustering,” Proceeding SIGIR '11 Proceedings of the 34th international ACM

國立政治大學 National Chengchi University

SIGIR conference on Research and development in Information Retrieval Pages 305-314 ACM New York, NY, USA, 2011.

[11] 張雅芳，利用 MapReduce 實作分散式協同推薦系統，國立清華大學碩士論文，2001。

[12] 林凡鈞，動態相似度協同過濾法的推薦系統，國立交通大學碩士論文，2002。

[13] 莊清男，協同過濾式群體推薦，國立中央大學碩士論文，2004。

[14] 曾靖茹，群集式協同過濾推薦方法之研究，國立中山大學碩士論文，2002。

[15] 林邑信，植基於 OSGi 與特徵選取之個人化推薦系統，國立高雄應用科技大學碩士論文，2000。

[16] 林順德，應用多維度關聯規則探勘在回饋式推薦系統，雲林科技大學碩士論文，1999。

[17] B. Sarwar, G. Karypis, J. Konstan. “Item-based collaborative filtering recommendation algorithms,” Published by ACM 2001 Article WWW '01 Proceedings of the 10th international conference on World Wide Web Pages 285-295 ACM New York, NY, USA, 2001.

[18] N. Good, J. B. Schafer, J. A. Konstan, A. Borchers, B. Sarwar, J. Herlocker, J. Riedl. “Combining Collaborative Filtering with Personal Agents for Better Recommendations,” AAAI '99/IAAI '99 Proceedings of the sixteenth national conference on Artificial intelligence and the eleventh Innovative Applications of artificial intelligence conference innovative Applications of artificial intelligence Pages 439-446 American Association for Artificial Intelligence Menlo Park, CA, USA, 1999.

[19] 戴英修，考慮商品異質性改善協同過濾推薦系統，國立清華大學碩士論文，1999。

[20] 王振遠，一個採用混合式推薦系統解決多樣性問題的方法，國立東華大學碩士論文，2002。

國立政治大學 National Chengchi University

[21] 林秀芬，基於協同過濾和雲模型的混合式推薦系統之研究，中國文化大學碩士論文，2001。

[23] M. V. Setten, M. Veenstra, A. Nijholt, B. V. Dijk. “Case-Based Reasoning as a Prediction Strategy for Hybrid Recommender Systems,” Journal of Advances in Web Intelligence, Lecture Notes in Computer Science Volume 3034, 2004.

[24] P. Melville, R. J. Mooney, R. Nagarajan. “Content-Boosted Collaborative Filtering for Improved Recommendations,” Eighteenth national conference on Artificial intelligence Pages 187-192 American Association for Artificial Intelligence Menlo Park, CA, USA, 2002.

[25] C. Basu, H. Hirsh, W. Cohen. “Recommendation as classification: Using social and content-based information in recommendation,” Venue: In Proceedings of the Fifteenth National Conference on Artificial Intelligence, 1998.

[26] 金柏均，花蓮旅遊景點查詢及推薦系統，國立東華大學碩士論文，2002。

[27] 范姜雅藍，建構於 Facebook 上之餐飲商店推薦系統，國立新竹教育大學碩士論文，2001。

[28] 鄧永聖，一個利用標籤為基礎之混合式推薦系統－以文化資產與歷史古蹟為例，逢甲大學碩士論文， 2000。

[29] R. Burke. “Hybrid Recommender Systems: Survey and Experiments,” Journal of User Modeling and User-Adapted Interaction November 2002, Volume 12, Issue 4, pp 331-370.

[30] 蔡子敬，以模糊語意法協助音樂 CD 資訊推薦系統之設計，國立屏東商業技術學院碩士論文，2000。

[31] U. Shardanand, P. Maes. “Social information filtering: algorithms for automating “word of mouth”,” CHI '95 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems Pages 210-217 ACM Press/Addison-Wesley Publishing Co. New York, NY, USA, 1995.

國立政治大學 National Chengchi University

[32] 羅健銘，協同過濾於網站推薦之研究，國立台北科技大學碩士論文，2000。

[33] 林文新，應用協同過濾法設計信用卡點數兌換之推薦系統，大同大學碩士論文，2008。

[34] Greg Linden, Brent Smith, Jeremy York. “Amazon.com Recommendations: Item-to-Item Collaborative Filtering,” IEEE Internet Computing, vol. 7, no. 1, pp. 76-80, Jan./Feb. 2003.

[35] 蔡松霖，電子商務推薦系統模型之初探，國立東華大學博士論文，2002。

[36] R. J. Mooney, L. Roy. “Content-based book recommending using learning for text categorization,” DL '00 Proceedings of the fifth ACM conference on Digital libraries Pages 195-204 ACM New York, NY, USA, 2000.

[37] T. Tran, R. Cohen. “Hybrid Recommender Systems for Electronic Commerce,” AAAI Technical Report WS-00-04, 2000.

[38] R. Forsati, H. M. Doustdar, M. Shamsfard, A. Keikha, M. R. Meybodi. “A fuzzy co-clustering Approach for hybrid recommender systems,” International Journal of Hybrid Intelligent Systems, Vol. 10, No. 2., 2013.

[39] 陳正德，以項目為基礎的協同過濾應用於網路教材瀏覽推薦之研究，銘傳大學碩士論文，2003。

[40] 黃信傑，以協同過濾輔助內容分析之文件推薦系統，國立中山大學碩士論文，2005。

[41] 鍾岳豪，以學習者個人偏好與試題反應理論為基礎之個人化英文文章推薦系統，逢甲大學碩士論文，2000。

[42] M. Balabanović, Y. Shoham. “content-based, collaborative recommendation,” Journal of Communications of the ACM CACM Homepage archive Volume 40 Issue 3, March 1997 Pages 66-72 ACM New York, NY, USA.

[43] 林恩羽，一個輔助營養師的諮詢與推薦系統，逢甲大學碩士論文，2001。

國立政治大學 National Chengchi University

[44] 邱俊耀，結合本體以多準則熵權重運算之推薦系統-以糖尿病藥物為例，朝陽科技大學碩士論文，2011。

[45] 林斐清，用使用者之 Facebook 社會網絡關係建立協同過濾推薦系統，屏東科技大學碩士論文，2001。

[46] 劉睿哲，使用擴充資料進行共分群的協同式推薦系統，國立中央大學碩士論文，2001。

[47] 曾億才，社群網路上運用群組關係於照片標註推薦系統，國立交通大學碩士論文，2001。

[48] M. Böhmer, L. Ganev, A. Krüger. “AppFunnel A Framework for Usage-centric Evaluation of Recommender Systems that Suggest Mobile Applications,” IUI '13 Proceedings of the 2013 international conference on Intelligent User interfaces Pages 267-276 ACM New York, NY, USA, 2013.

[49] B. Yan, G. Chen. “AppJoy: Personalized Mobile Application Discovery,” MobiSys '11 Proceedings of the 9th international conference on Mobile systems, Applications, and services Pages 113-126 ACM New York, NY, USA, 2011.

[50] X. Xia, X. Wang, J. Li, X. Zhoua. “Multi-objective mobile App recommendation A system-level collaboration Approach,” Computers & Electrical Engineering Volume 40, Issue 1, January 2014, Pages 203–215, 2013.

[51] K. Shi, K. Ali. “GetJar Mobile Application Recommendations with Very Sparse Datasets,” KDD '12 Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining Pages 204-212 ACM New York, NY, USA, 2012.

[52] S. Rendle, L. Schmidt-Thieme. “Online-Updating Regularized Kernel Matrix Factorization Models for Large-Scale Recommender Systems,” Proceeding RecSys '08 Proceedings of the 2008 ACM conference on Recommender systems Pages 251-258 ACM New York, NY, USA, 2008.

國立政治大學 National Chengchi University

[53] V. Sindhwani, S.S. Bucak, J. Hu, A. Mojsilovic. “A Family of Non-negative Matrix Factorizations for One-Class Collaborative Filtering Problems,”2009

[54] Wu H, Wang YJ, Wang Z, Wang XL, Du SZ. “Two-Phase collaborative filtering algorithm based on co-clustering.” Journal of Software, 2010

[55] A. Banerjee, I. Dhillon, J. Ghosh, S. Merugu, D. S. Modha. “A Generalized Maximum Entropy Approach to Bregman Coclustering and Matrix Approximation,” Proceeding KDD '04 Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining Pages 509-514 ACM New York, NY, USA, 2004.

[56] B. Li1, Q. Yang, X. Xue1. “Can Movies and Books Collaborate? Cross-Domain Collaborative Filtering for Sparsity Reduction,” Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence (IJCAI-09), 2009.

[57] P. Chen, H. Wu, C. Hsu, W. Liao, and T. Li. “Logging and Analyzing Mobile User Behaviors,” International Symposium on Cyber Behavior, Taipei, Taiwan, February 2012.

[58] Y. Cheng and G. Church. Biclustering of expression data. In Proceedings ISMB, pages 93103. AAAI Press, 2000.

國立政治大學 National Chengchi University