A Recommender System Based On Multi-relational Social Networks

何志軒
國立中正大學
s1046206@hotmail.com

吳邦一
國立中正大學
bangye@cs.ccu.edu.tw

摘要

全球的網站數量在近幾年來呈現指數性的成長，有許多不同種類的網站相繼出現，例如：社群網站、資料站、部落格以及維基百科[1]。其中一些資料站會與社群網站結合(像是 Facebook[2] 或是 Twitter[3])或是本身就提供內部的社群網路，這麼做的目的是為了讓資料站的使用者間可以互相討論，進而提高網站的能見度。由於網路上的資訊量成長的過於快速，以至於讓人們越來越難分辨出正確且有用的資訊，這讓推薦系統所扮演的角色越來越重要。在這篇論文中，我們提供了一個在多重關係的社會網路中以單一目標為主的探索式的推薦方法。我們利用人工的方式模擬實際資料，並利用這些資料做實驗來驗證可行性。在未來會利用一個提供內部社會網路電影的資料網站"Rotten Tomatoes[4]"來做實驗，並判斷我們方法的好壞。

關鍵詞：群集探查、多重關係、推薦系統

Abstract

In recent years, the number of global websites has increased exponentially. Many different types of website emerged such as social networks, information sites, blogs and wikis. Some information sites combined social networks (like Facebook or Twitter) or provided an inside social network. The purpose is letting visitors interact with others and further extracting more attention. The growth of information on web is so fast that people are hard to get true and useful information. It makes recommender system play a more important role than before. In this paper, we provide a heuristic ego centered recommending method based on a multi relational social network. We produced artificial data and experimented on it to proof the method is workable. We will use the data in "Rotten Tomatoes", a movie information site with an inside social network, to check the goodness of our method in the future.

Keywords: Community detect, Multi-relation, Recommender system.

1. 前言

隨著網路的發達，有越來越多不同種類的網站出現，其中又以社群網站最受一般大眾的注意，例如 Google+[5]、Facebook、Twitter 及 Flickr[6] 皆非常受大家歡迎。人們可以在社群網站上擁有自己的帳號，且隨意的分享自己所遇到的事、照片甚至是影片。在越來越多人使用社群網站分享訊息的同時也代表網路上的訊息量越來越多，且包含的內容五花八門更可能會有錯誤的訊息；但訊息量變多也代表可以利用的資源越多，要如何篩選出正確且有用的訊息就變成了最大的課題。因此推薦系統越來越受到許多應用軟體及網站的看重，藉由推薦使用者想看的資訊可以讓應用軟體或網站更受歡迎進而提高他們的收益。現在有許多的資料站會結合社群網站(例如 Facebook 或 Twitter)或是提供內部的社群網路讓使用者在他們網站註冊自己的帳號。無論是哪一種，目的都是要讓使用者可以互相討論或分享資訊，進而提高網站的能見度。像是"Rotten Tomatoes"就是屬於第二種的電影資料網站，在這個網站使用者可以建立自己的帳戶並且新增別的帳戶為好友，還會顯示已經評分過的電影清單以及有興趣的電影清單，這篇論文會利用這個網站的資料進行實驗來判斷我們提出的方法的好壞。

在社會網路分析(Social Network Analysis, SNA)領域中通常會將社會網路以 graph 表示，其中以點代表人，點之間的邊代表人之間存在某種關係，利用這個方法可以將一群人之間的關係簡化為一張圖，而人之間一定會存在不同的小群體，分群會將關係較為緊密的一群人分在同一個群體，且讓群之間的關係相對較為鬆散。在之前已經有許多知名的學者對如何分群提出許多不同的看法，例如最著名的 Newman 和 Girvan 提出的 betweeness [8]，利用點之間的最短路徑計算每條邊被走過次數，再利用他們對 community 的定義，藉由每次重新計算得到最大 betweeness 的邊並刪除，最後再計算 modularity [8]，就可以得到分群的結果。其後也有許多不同的學者提出不一樣的方法，例如：Lancichinetti et al method [10]、Louvain method [1] 與 Palla et al method [9]等，都是非常知名的分群方法，甚至還有一些是針對多重關係(multi-relation)的圖所用的分群方法。分群最常見的應用方式就是社群網站的好友推薦，而推薦就是利用使用者之間或物件(Item)之間的相似度(Similarity)，並針對相似度高的使用者或物件做推薦，相似度的計算方法也已經有許多學者提出計算的方法，例如：Cosine-based similarity [11、12]、Pearson correlation coefficient [6]與 Jaccard coefficient [5]，這些方法都被廣泛的研究以及使用，而隨著不同的資料推薦的

1. https://www.wikipedia.org/
2. https://www.facebook.com/
3. https://twitter.com/
4. www.rottentomatoes.com/
5. https://plus.google.com/
6. https://www.flickr.com/

物件也會不同。

這篇論文將使用到三種資料，第一種是使用者之間的好友關係，第二種是使用者所喜歡的物件(在這篇論文指的是電影)以及物件的屬性(在這篇論文指的是電影的種類以及演員和導演)。所使提出的方法是先利用分群找出推薦目標(使用者)所屬的群，再利用群內的使用者所喜歡的物件(電影)間的關聯性找出要推薦的物件。接下來在第二節會介紹本篇論文所提出的方法以及想法，第三節介紹我們人工產生資料的方法及利用人工資料所得到的結果，最後一節則是本篇論文的結論及未來研究方向。

2. 群集以及推薦方法

在目前的推薦系統中，可以將推薦的方法分為三種，第一種為 Memory-based Collaborative Filtering Techniques[15、17]，這種方法是利用使用者與物件(user-item)之間的關係來計算相似度或權重；第二種為 Model-based Collaborative Filtering Techniques[15、16]，這種方法會利用 Machine learning 或 Data mining 的方法建立模型，再利用許多資料讓系統學習如何辨識複雜的單詞，就可以找出可以推薦的物件；第三種為 Hybrid Collaborative Filtering Techniques[2、3、4、7、14]，這種方法會結合不同的推薦方法來預測或是推薦 。

我們提出的方法是屬於 Memory-based CF Technique；這種方法又可以細分為兩種，第一種為以使用者為基礎(User-based)的方法，這種方法會計算使用者之間的相似度，再找出相似度較高的使用者並推薦他們所喜歡的物件給推薦目標；第二種為以物件為基礎(Item-based)的方法，這種方法與以使用者為基礎的方法的差別在於這種方法是計算物件之間的相似度，並不考慮使用者之間的關聯性。單獨使用這兩種方法所能得到的效果有限，所以我們想要同時運用到兩種方法的特性，先利用多重關係的分群找出推薦目標所屬的群，這相當於找出與推薦目標相似的使用者，接著再利用這些使用者所喜歡的物件之間的關聯性找出推薦目標可能會喜歡的物件。

2.1. 方法簡介

我們提出的方法所需要使用到的資料為一群使用者，這些使用者之間可能會存在好友的關係，使用者也會喜歡一些物件，而這些物件會有一至多種屬性；我們會先利用使用者間的好友關係以及物件的屬性建立使用者間的多重關係圖(詳細內容會在第三節做說明)，接著再以推薦目標為出發點向外擴展找出同群的使用者，找到這些使用者後，再利用這些使用者所喜歡的物件以及這些物件的屬性計算出與推薦目標所喜歡的物件相似度較高的物件做推薦。

2.2. 群集偵測及多重關係

群的概念被定義成群內部的使用者之間的關係較為緊密，群跟群之間則較為鬆散；換句話說就是群內的邊密度較群之間高，在一些應用上會希望得知使用者之間是如何分群的。傳統的分群是在單一關係的圖中進行分群，但目前漸漸的也有學者在研究多重關係的分群，而分群演算法的時間複雜度(Time complexity)較高，在點數過多時並不適合頻繁的使用，所以我們提出了一個以單一目標向外擴展的群集探查(Community detect)方法，這樣可以針對單一目標做推薦且降低單次推薦的時間複雜度。

本篇論文所提出的方法的第一部分是在多重關係圖中尋找推薦目標所屬的群。在探查推薦目標所屬的群前必須先設定一個 α 值，每次群往外擴展後都必須重新計算群內各個關係的出現頻率，若某種關係頻率低於 α，則下次擴展時不將該關係列入考慮。方法可以分為以下的步驟：

1. 一開始先往與推薦目標相連關係數目最多的相鄰點擴展，將這些相鄰點納入群的集合 C_m 內。
2. 計算 C_m 內各關係出現的頻率，並將頻率高於 α 的關係的集合設為 R_h。
3. 若 C_m 內的點與他們相鄰點以 R_h 內的子集合相連則將這些相鄰點納入 C_m 內。
4. 重複步驟 2 與 3 直到無法擴展。

這麼做是為了要找出與推薦目標的喜好較為相近的使用者並減少之後推薦時所需的計算量，找到後就可以利用群內其餘使用者所喜歡的物件更進一步的計算物件之間的相似度。

現在提出一個例子說明(如圖 1. 多重關係圖例)，假設點 A 是推薦目標且 $\alpha = 0.3$。

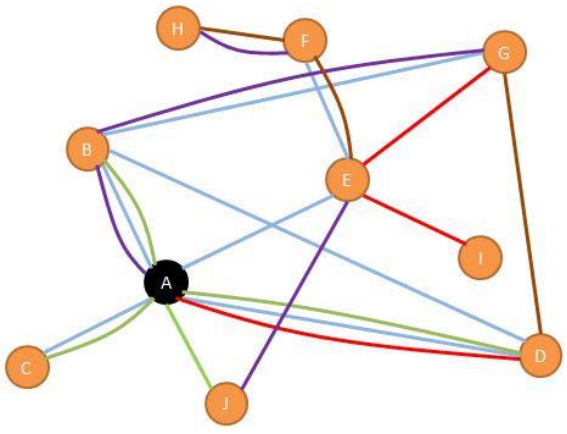

圖 1 多重關係圖例

(1) $C_m = \{A\}$, $R_h = \emptyset$
(2) A 與 B、D 是以三種關係相連，所以將 B、D 加入 C_m 中，$C_m = \{A, B, D\}$
(3) 計算各關係的出現頻率，
rate(BLUE) = 0.429,
rate(GREEN) = 0.286,
rate(RED) = 0.143,

rate(PURPLE) = 0.143
所以R_h = {BLUE}

(4) 繼續以關係 BLUE 向外擴展，C_m = {A, B, C, D, E, G}

(5) 各關係出現頻率為
rate(BLUE) = 0.429,
rate(GREEN) = 0.214,
rate(RED) = 0.143,
rate(PURPLE) = 0.143
rate(BROWN) = 0.071
所以R_h = {BLUE}

(6) 繼續以關係 BLUE 向外擴展，C_m = {A, B, C, D, E, F, G}

(7) 各關係出現頻率為
rate(BLUE) = 0.438,
rate(GREEN) = 0.188,
rate(RED) = 0.125,
rate(PURPLE) = 0.125
rate(BROWN) = 0.125
所以R_h = {BLUE}

(8) 無法再繼續擴展，結束並得到推薦目標 A 所屬的群為C_m = {A, B, C, D, E, F, G}

這個方法最重要的步驟就是每次計算關係的出現頻率與α值的設定，若設定的太高則關係數目太少會讓向外擴展的點數不夠，這樣在之後推薦時物件的個數會過少；相反的，若設定的太低會向外擴展太多使得物件個數太多，讓推薦準確度降低。演算法如下所示：

Algorithm 1: Community_detect

Input: a multi-relational graph G and a recommended target T.
Output: a set of nodes belonging to the same community with the recommended target.

```
1:  C_m ← {T};
2:  R_h ← ∅;
3:  N ← neighbor of C_m;
4:  N_max ← neighbors_with_most_relation(N);
5:  C_m ← C_m ∪ N_max;
6:  for all kind of relation r_i in C_m do
7:    if appearing_ratio(r_i) ≥ α do
8:      R_h ← R_h ∪ r_i;
9:    end
10: end
11: while R_h ≠ ∅ do
12:   N ← ∅;
13:   for all neighbor n_i of C_m do
14:     if relation(n_i, C_m) ⊆ R_h do
15:       N ← N ∪ n_i;
16:     end
17:   end
18:   C_m ← C_m ∪ N;
19:   R_h ← ∅;
20:   for all kind of relation r_i in C_m do
21:     if appearing_ratio(r_i) ≥ α do
22:       R_h ← R_h ∪ r_i;
23:     end
24:   end
25: end
26: return C_m;
```

2.3. 推薦方法

在上述的群集探查方法中已經得到推薦目標所屬的群集了，接著就利用群集內的使用者所喜歡的物件的聯集計算出與推薦目標所喜歡的物件相似度高的物件做推薦。在之前已經有許多學者研究出計算相似度的方法，我們會利用一個以物件為基礎的計算方法，名為"Cosine-based similarity"，該方法是先利用使用者與物件的關係建好矩陣，若要計算物件 i 與物件 j 之間的相似度就計算第 i 行向量與第 j 行向量的 cosine 值，越接近 1 代表越相像，越接近-1 則代表越不相像。

我們提出的方法是先算出推薦目標所喜歡的物件的屬性(attribute)出現頻率。接著找出同個群內其他使用者喜歡的物件及這些物件所包含的屬性，並用這些資料建立一個物件與屬性的矩陣，並計算出屬性之間的相似度。接下來再給每一種屬性權重並且利用 Weighted Sum[13]這個方法給同群內的其他使用者所喜歡的物件評分，分數越高代表推薦目標越有可能喜歡。而我們提出的方法可以分為以下的步驟：

1. 利用推薦目標所喜歡的物件找出這些物件的屬性的聯集$S_{att} = \{s_i | 1 \le i \le g\}$，並計算出這些屬性的出現頻率$S_{fre} = \{f_i | 1 \le i \le g\}$。
2. 找出同個群內其他使用者喜歡的物件聯集$T_{item} = \{d_i | 1 \le i \le m\}$與$T_{item}$中物件的屬性聯集$T_{att} = \{a_i | 1 \le i \le n\}$，建立一個$m \times n$的矩陣$T = [t_{ij}]_{m \times n}$，$t_{ij} = 1$代表第 i 項物件有第 j 種屬性，若$t_{ij} = 0$則相反。
3. 接著再利用 T 來計算屬性之間的 Cosine-based similarity；假設$P = [p_{ij}]_{n \times n} = \text{cosineSim}(T)$，$\vec{t_i}$為 T 中第 i 行的向量，則
$$p_{ij} = \frac{\vec{t_i} \cdot \vec{t_j}}{\|\vec{t_i}\| \|\vec{t_j}\|}$$
4. 建立一個集合$V_{att} = \{v_k | 1 \le k \le n\}$，若$T_{att}$中的第 k 種屬性與$S_{att}$中第 i 種屬性相同($a_k = s_i$)，則$v_k = nf_i$，反之則為 0。
5. 利用 P 與V_{att}計算$M = [m_{ij}]_{m \times n}$
$$m_{ij} = \frac{\sum_{\text{all } N} \left(p_{jN} * (v_j + v_N) \right)}{\sum_{\text{all } N} (|p_{jN}|)},$$
$$N = \text{attribute}(d_i)$$
接著計算每一列的總和為w_i
$$w_i = \frac{\sum_{k=1}^{n} m_{ik}}{\#\text{attribute}(d_i)}, 1 \le i \le m$$
w_i為第 i 種物件的權重。
6. 設定一個門檻值(threshold)μ，若$w_i \ge \mu$就將物件 i 推薦給推薦目標。

這個方法的想法是先找出與推薦目標有相同喜好的群，找到後再計算推薦目標所喜歡的屬性的出現頻率；接著再找出同群的其他使用者所喜歡的物件以及這些物件的屬性，利用 Cosine-based similarity 計算出這些屬性之間的相似度的矩陣後再利用 Weighted Sum 來計算物件中的每個屬性的權重，再將這些權重加總取平均，取平均是為了避免因為每種物件的屬性個數不同而影響到結果，接下來再設定門檻值 μ，只要權重大於 μ 的物件就對目標做推薦。

3. 實驗及分析

我們為了要驗證我們提出的方法的可行性，我們會先建立人工資料，接著再利用我們的方法來得到我們所提出的方法的好壞。而目前有許多判斷一個方法好壞的方法，本篇論文使用的方法為查準率 (Precision)、查全率 (Recall) 以及 F 度量 (F-measure)；這三種方法會先將得到的結果分為四項正確正例(True Positive, TP)、錯誤正例(False Positive, FP)、錯誤負例(False Negative, FN)以及正確負例(True Negative, TN)；TP 在我們的實驗結果代表正確的推薦個數、FP 為錯誤的推薦個數、FN 為目標喜歡但沒推薦的個數，而 TN 則代表目標不喜歡也沒有推薦的個數。

Precision 所代表的意義為推薦命中的比率，可以表示為

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

而 Recall 所代表的意義為目標所喜歡的所有物件的命中比率，可以表示為

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

而 F-measure 為 Precision 與 Recall 的調和平均數

$$\text{F} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

我們將用這三種方法做為判斷好壞的依據。

3.1. 人工資料產生方式

我們的人工資料可以分為三個部分：使用者、電影以及電影的屬性(種類和演員)；使用者之間可能存在好友關係，使用者會喜歡某些電影而這些電影又有著各自的種類及演員。

我們將使用者 1000 人分為 5 群、電影 4800 部分為 4 群、電影種類 16 種分為 4 群及不分群的 3 種而演員共 12000 人分為 4 群。每群人皆有各自喜歡的同一群電影及隨機不同群電影(雜訊)，數量為 50-750 部；每部電影也會有各自的同一群種類及隨機的不分群種類(雜訊)和同一群的演員，種類的數量為 1-4，演員的數量為 5 人；其中電影有一種種類的機率為 0.2，兩種的機率為 0.35，三種的機率為 0.36，四種的機率為 0.09；而各個雜訊的產生機率皆為 0.05。詳細的產生及分配情況請見下列各表。

表 1 使用者分群

	A	B	C	D	E
人數	400	200	200	100	100
編號	1-400	401-600	601-800	801-900	901-1000
群內邊產生機率	0.7	0.8	0.6	0.7	0.8
群間邊產生機率	0.2				

表 2 電影分群

	1	2	3	4
數量	1500	1450	1100	750
編號	1-1500	1501-2950	2951-4050	4051-4800

表 3 電影屬性分群

		(a)	(b)	(c)	(d)	(e)
種類	數量	4	4	3	2	3
	編號	1-4	5-8	9-11	12-13	14-16
		(1)	(2)	(3)	(4)	
演員	數量	3000	3000	3000	3000	
	編號	1-3000	3001-6000	6001-9000	9001-12000	

表 4 使用者與電影分配

使用者	A	B	C	D	E
電影	1	2	3	1	4
雜訊	2、3、4	1、3、4	1、2、4	2、3、4	1、2、3

表 5 電影與屬性分配

電影	1	2	3	4
種類	(a)	(b)	(c)	(d)
雜訊	(e)			
演員	(1)	(2)	(3)	(4)

3.2. 建立多重關係圖

在產生完人工資料後，第一步要先把這些資料轉換成多重關係圖，才能利用我們提出的群集偵測的方法找出與推薦目標有共同喜好的群。而將資料轉換為多重關係圖時，任兩個使用者之間要存在關係必須滿足兩項條件，第一項為兩個使用者之間必須為好友關係，第二項為兩個使用者必須有共同喜歡的電影，而兩個使用者之間的關係就由共同喜歡的這些電影的屬性來決定。我們將電影的屬性分為兩種，第一種為電影的種類(Type1)，第二種為電影的演員(Type2)，會這樣分的原因是因為本質以及數量上的不同。

建圖時我們會對每一對存在好友關係的使用者進行判斷，先找出他們喜歡的電影交集再統計這些電影的 Type1 屬性的出現頻率(δ_i)以及 Type2 屬性的出現次數(t_i)。以下說明如何判斷 Type1 屬性是否能成為兩使用者之間的一種關係，

- a_i代表 Type1 屬性 i 的出現次數，s_{att}代表全部 Type1 屬性的出現總次數，k_{att}代表喜歡的電影交集中的屬性聯集種類數，R_{jk}代表使用者 j 跟 k 之間的關係集合。

$$\delta_i = \frac{a_i}{s_{att}}$$

$$\text{if } \delta_i \geq \frac{2}{k_{att}} \text{ then } R_{jk} \leftarrow R_{jk} \cup \{i\}$$

而判斷 Type2 屬性能否成為使用者之間的關係則用下列的方式

- t_i為 Type2 屬性 i 的出現次數，t_m為喜歡的電影交集個數。

$$\text{if } t_i \geq 5 \text{ and } t_i \geq \frac{t_m}{20} \text{ then } R_{jk} \leftarrow R_{jk} \cup \{i\}$$

在 Type1 中的是電影的種類，而人們通常會對特定種類的電影較有興趣，所以這邊的目的是找出出現次數相對較多的電影種類當作兩使用者之間的關係；在 Type2 中則是演員，有些人會對特定的演員感興趣，進而影響到他對電影的喜好，但是並不是每個人都會喜歡特定的演員，所以我們從出現的次數以及在交集電影中所佔的比例來決定某兩位使用者是否共同喜歡特定演員。我們的人工資料中，好友關係圖中的邊數為 128023，轉換為多重關係圖後，不計算重邊的情況下只剩 20077 條邊，圖 2 為建好多重關係圖之後的鄰居數分佈情形，圖 3 為關係種類數的分佈情形。

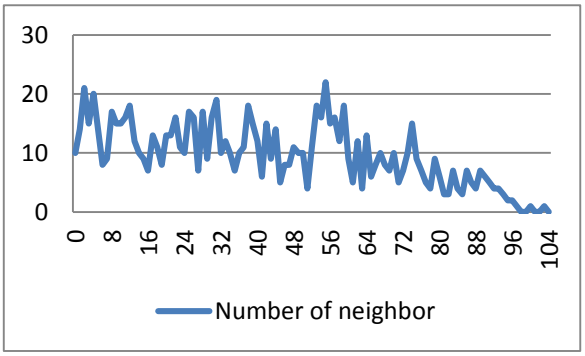

圖 2 多重關係鄰居數分佈圖

判斷是否能成為關係的條件都可以降低或是提高，若把條件降低，之後群集偵測得到的群就會越大，該群喜歡的電影種類就會越多樣；若把條件提高，則偵測出的群會較小，電影的種類也會因為限制而變得較為專一。

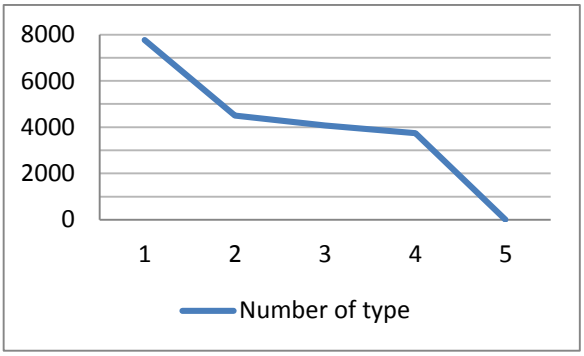

圖 3 多重關係種類數分佈圖

下表為在多重關係圖中各群的使用者的平均鄰居數，由下表以及前面各表可以觀察出，使用者所喜歡的電影的 Type1 屬性多寡會對多重關係圖的鄰居個數造成影響；群集 C 與 E 所喜歡的電影群集為 3 和 4，而電影群集 3 和 4 所被分配到的 Type1 屬性個數為 3 種與 2 種，C 與 E 的平均鄰居個數都比其餘對應電影群集 Type1 屬性為 4 種的 A、B 與 D 高。

表 6 各群平均鄰居數

使用者群集	A	B	C	D	E
平均鄰居數	31.96	29.01	57.52	21.76	78.90

3.3. 群集偵測與推薦結果

得到多重關係圖之後就要利用我們提出的群集偵測方法找出各個推薦目標的群，在群集偵測時需要先決定 α 值，下表為各群使用者的 α 值。

表 7 使用者群集與 α 值

使用者群集	A	B	C	D	E
α 值	0.2	0.25	0.32	0.52	0.52

我們輪流把每一個使用者當作推薦目標，並且計算分群結果中同群的使用者佔了群集多少百分比，以及同群所佔百分比除以不同群所佔百分比，下表為各群的平均。

表 8 各群平均同群百分比

使用者群集	A	B	C	D	E
同群百分比	74.8%	50.2%	55.3%	81.8%	95.0%

同群/不同群	3.028	1.022	1.517	3.508	10.607

由上表可以看出除了 B 和 C 兩群外，各群平均偵測出的同群百分比有七成以上，而 B 和 C 雖然比較低但是也有在五成以上。而雖然 B 與 C 的分群結果沒有其餘三群好，但是在最後的推薦步驟還是能獲得不錯的結果。

接下來就是針對分群得到的結果進行運算及推薦。首先要計算同群所喜歡的電影聯集中出現的屬性之間的相似度，但是因為電影的數量太少，所以同群的使用者所喜歡的電影聯集與電影數量很接近；但是由於我們的方法會利用推薦目標所喜歡的電影的屬性的出現頻率當作推薦目標對該屬性的評分，所以最後的結果不會受到電影聯集的多寡影響，會有影響的是電影聯集中同群電影的多寡。計算完屬性間的相似度後就要針對每個推薦目標所屬群集的電影聯集中的每一部電影計算 Weighted Sum 並且將這些電影依據分數由高到低排序，有了這些電影的分數接下來就要找出如何定出最適合的推薦門檻值，由於電影的分數大小分佈會被推薦目標所喜歡的電影數目所影響，所以我們將每個推薦目標的最佳門檻值以及推薦目標所喜歡的電影數目做成散佈圖並對該圖做線性擬合(圖 4)找出推薦目標喜歡的電影數目與門檻值的轉換式為

$$y = 223.88x^{-0.595}$$

x 為推薦目標喜歡的電影數目，y 為門檻值。最後我們利用該式去計算每個推薦目標的門檻值並做推薦。圖 5 為以編號 1 到 1000 的使用者做為推薦目標所得到的推薦結果的 Precision、Recall 與 F-measure 的分佈圖，由圖中可以看出來每群推薦結果的 Precision 平均約在 0.8 左右，Recall 則在 0.7 左右，而 F-measure 則在 0.75 左右；雖然這只是人工產生的資料，但是這些結果表現出這個方法是可行的。

3.4. 方法分析

我們提出的方法的使用前提是需要有三層資料(使用者、電影、電影屬性)之間的關係以及使用者之間的關係。在有這些資料的情況下我們提出的方法可分為兩部分：群集偵測和推薦計算，在群集偵測中最主要的目的是找出與推薦目標有相同喜好的群，進而減少第二部分的計算量，因為如果要對所有的電影做計算勢必會花費許多的時間，而由於我們產生的資料量不夠大，所以這部分的表現較不顯著。在推薦計算中我們使用了 Cosine-based Similarity 來計算屬性之間的相似度，這麼做是為了在最後計算 Weighted Sum 時提高與推薦目標所喜歡的電影屬性相似的電影的分數；而最後門檻值的計算可能會因為資料不同而使得轉換式也需要重新計算。

4. 結論及未來研究方向

本篇論文提出了一個推薦的方法，該方法可以分為兩個部分，第一個部分是利用使用者之間的好友關係以及共同喜歡的物件的屬性建立一個多重關係圖，再從推薦目標向外擴展找出可能所屬的群；第二部分是利用名為"Cosine-based similarity"的計算方法計算屬性之間的相似度再利用”Weighted Sum”與推薦目標所喜歡的屬性出現頻率就可以對物件進行評分。

我們為了驗證我們提出的方法的可行性而產生人工資料，最後得到的結果顯示出我們提出的方法是可行的。我們之後也會從電影資料網站”Rotten Tomatoes”抓取真實資料來進行實驗，並修改我們所提出的方法使它更接近真實情況。

本篇論文提出的方法最重要的部分是在擴展群時要如何保留對推薦有利的關係，使得找到的群的喜好與推薦目標的喜好是相符合的，因為若能找出擁有相同喜好的群，就能夠更容易的找出推薦目標所喜歡的物件；以及推薦目標所喜歡的電影數量與門檻值的轉換，我們在未來也會從這個方向著手進行研究。

參考文獻

[1] Blondel, V. D., Guillaume, J. L., Lambiotte, R., & Lefebvre, E. (2008). *Fast unfolding of communities in large networks.* Journal of Statistical Mechanics: Theory and Experiment, 2008(10), P10008.

[2] Guha, R., Kumar, R., Raghavan, P., & Tomkins, A. (2004, May). *Propagation of trust and dis-trust.* In Proceedings of the 13th international conference on World Wide Web (pp. 403-412). ACM.

[3] Guo, G., Zhang, J., & Thalmann, D. (2013). *Merging trust in collaborative filtering to alleviate data sparsity and cold start.* Knowledge-Based Systems.

[4] Hannon, J., Bennett, M., & Smyth, B. (2010, September). *Recommending twitter users to follow using content and collaborative filtering approaches.* In Proceedings of the fourth ACM conference on Recommender systems (pp. 199-206). ACM

[5] Jaccard, P. (1901). *Etude comparative de la distribution florale dans une portion des Alpes et du Jura.* Impr. Corbaz.

[6] Lancichinetti, A., Fortunato, S., & Kertesz, J. (2009). *Detecting the overlapping and hierarchical community structure in complex networks.* New Journal of Physics, 11(3), 033015.

[7] Liu, F., & Lee, H. J. (2010). *Use of social net-work information to enhance collaborative fil-tering performance.* Expert Systems with Applications, 37(7), 4772-4778.

[8] Newman, M. E., & Girvan, M. (2004). *Finding and evaluating community structure in*

networks. Physical review E, 69(2), 026113.

[9] Palla, G., Derenyi, I., Farkas, I., & Vicsek, T. (2005). *Uncovering the overlapping community structure of complex networks in nature and society.* Nature, 435(7043), 814-818.

[10] Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., & Riedl, J. (1994, October). *GroupLens: an open architecture for collaborative filtering of netnews.* In Proceedings of the 1994 ACM conference on Computer supported cooperative work (pp. 175-186). ACM.

[11] Salton, G., & McGill, M. J. (1983). *Introduction to modern information retrieval.*

[12] Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2000, October). *Analysis of recommendation algorithms for e-commerce.* In Proceedings of the 2nd ACM conference on Electronic commerce (pp. 158-167). ACM.

[13] Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2001, April). *Item-based collaborative filtering recommendation algorithms.* In Proceedings of the 10th international conference on World Wide Web (pp. 285-295). ACM.

[14] Su, J. H., Chang, W. Y., & Tseng, V. S. (2013). *Personalized Music Recommendation by Mining Social Media Tags.* Procedia Computer Science, 22, 303-312.

[15] Su, X., & Khoshgoftaar, T. M. (2009). *A survey of collaborative filtering techniques.* Advances in artificial intelligence, 2009, 4.

[16] Su, X., & Khoshgoftaar, T. M. (2006, Novem-ber). *Collaborative filtering for multi-class data using belief nets algorithms.* In Tools with Arti-ficial Intelligence, 2006. ICTAI'06. 18th IEEE International Conference on (pp. 497-504). IEEE.

[17] Wang, J., De Vries, A. P., & Reinders, M. J. (2006, August). *Unifying user-based and item-based collaborative filtering approaches by similarity fusion.* In Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval (pp. 501-508). ACM.

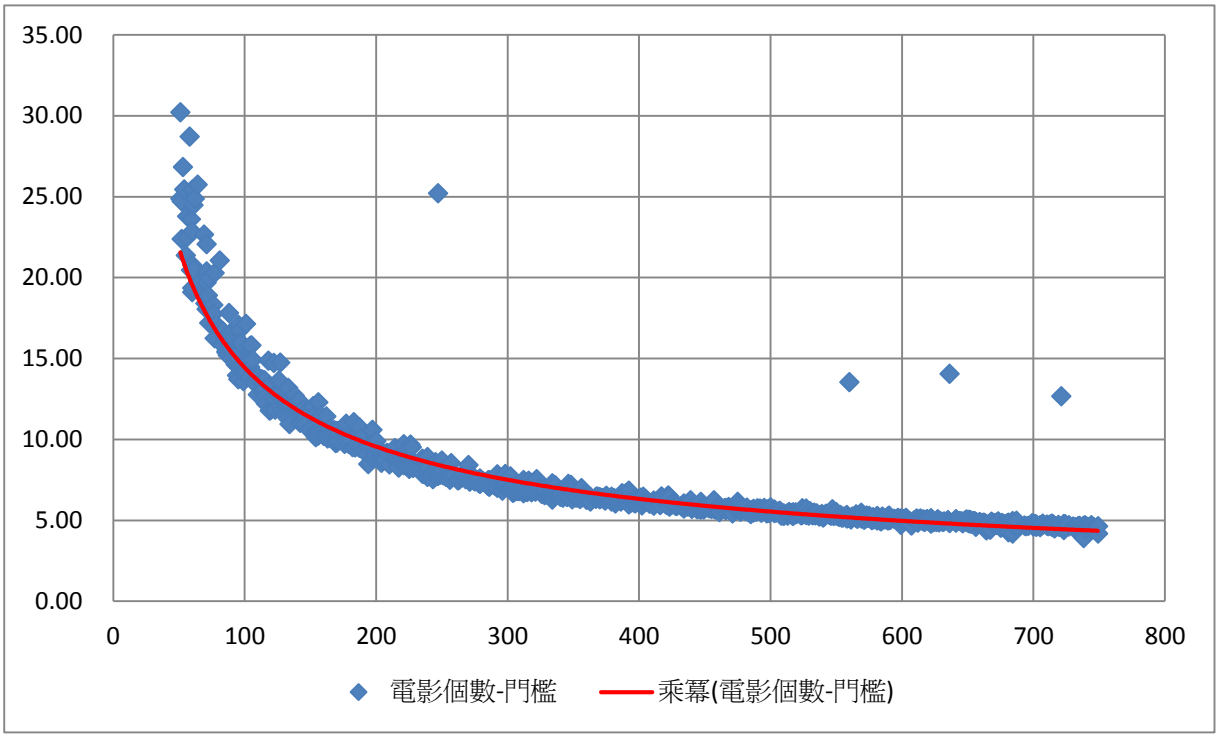

圖 4 電影個數與門檻值散佈圖

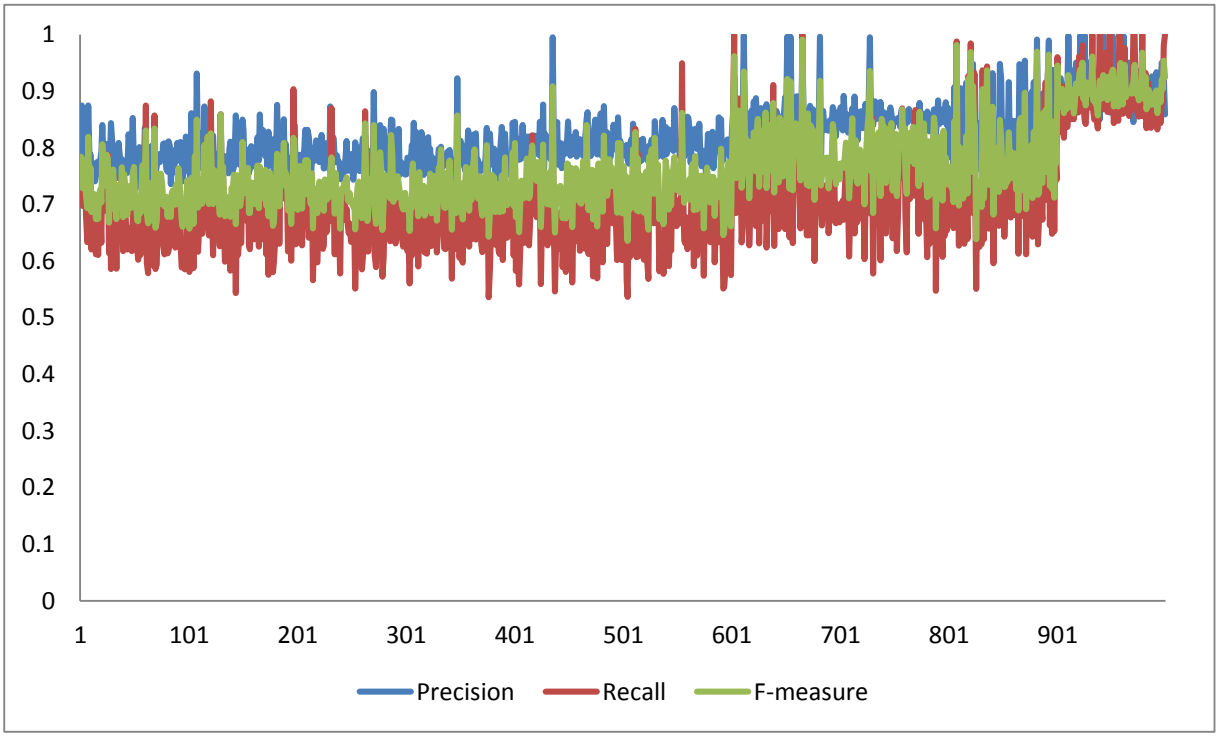

圖 5 Precision, Recall 與 F-measure 分佈圖