Université de Tunis
Institut Supérieur de Gestion

MASTERE RECHERCHE

Spécialité
Science et Technique de l'informatique décisionnelle (STID)

Option
Informatique et Gestion de la Connaissance (IGC)

A NEW FRAMEWORK FOR DYNAMIC COMMUNITY-BASED RECOMMENDATION

Sabrine Ben Abdrabbah

Nahla Ben Amor	Professeur, ISG Tunis	Directeur du mémoire
Raouia Ayachi	Maitre assistante, ESSEC Tunis	Co-directeur

Lieu: LARODEC

2013-2014

Acknowledgment

In the end of this work, I would like to express my sincere thanks to my advisor Pr. Nahla Ben Amor for her guidance and help throughout my studies and for valuable advice. It has been an honor to work with you.

I would thank my co-advisor Dr. Raouia Ayachi for her help, her advice and especially for reading carefully this report allowing me to improve its content.

I would thank Dr. Rémy Cazabet for his precious help and clarifications on everything related to community detection and for providing me the pseudo code of his algorithm (*iLCD*).

I am also grateful to all the professor of *Institut Supérieur de Gestion de Tunis* for providing me the background that i need for study and research. I thank all the members of the *LARODEC* laboratory.

The most special thanks goes to my fiance Eymen Kadri for his unconditional encouragement, patience and love through all this long process.

In closing, i would like to thank my two families (Ben Abdrabbah and Kadri) and my friends who have been of great support through my studies.

Contents

List of Figures

List of Tables

List of Algorithms

Introduction

Recommender systems have emerged in the past several years as an efficient tool to deliver users with more intelligent and proactive information service. They recommend products or services that fit well with the learned users' preferences and needs. These systems generally combine, on one hand, information extracted from users' profiles and social interactions, and on the other hand, machine learning techniques that are used to predict the user ratings or preferences. Each recommender system has its particular specifications which are related to the application domain, the used technique for recommendation and the kind of its users (i.e. a person or a social group). Several types of recommenders have been proposed that can be categorized into three major categories (Song, Lin, Tseng, & Sun, 2006), namely content-based filtering, collaborative filtering, and hybrid approaches.

In this work we focus on collaborative filtering which predicts users interests/preferences from those of remaining users sharing similar tastes. Collaborative filtering is considered as one of the most used techniques due to its efficiency and its high accuracy (Qiang & Yan, 2012). Typically, the recommendation process for this technique starts when users express their preferences by rating items. The system analyzes these ratings to determine the exact preferences of the users, then, matches the active user's preferences and the preferences collection to discover users having similar taste with the active user. Finally, the system recommends a set of items for the active user according to the preferences of their similar users.

From another side, the community detection presents a growing interest for many researchers, especially in the web applications. A panoply of community detection algorithms exist in literature and most of them focus on static community detection but recently the dynamic aspect of networks has sparked a new line of research. Static community detection algorithms have been explored in collaborative filtering based on the idea that using community structure in networks will enhance the performance of the recommendations

(Fatemi & Tokarchuk, 2013; Qiang & Yan, 2012; Qin, Menezes, & Silaghi, 2010; Sahebi & Cohen, 2011; Zhao, Lee, Hsu, Chen, & Hu, 2013). Nevertheless, the accumulation of an important mass of data in the same time and in the same graph can lead to illegible graphs, not able to deal with the dynamic aspect of real world networks.

In this paper we model the dynamic behavior of users interests in recommender systems. Users' interests are learned from users' ratings data and more specially by looking firstly at the items which have been rated by users and then finding a way to link items to each other in order to build the dynamic network of items. Our major contribution consists in presenting a novel approach named *Dynamic Community-based Collaborative Filtering* (denoted D2CF for short) capturing dynamic users communities to offer recommendations more suitable for real world networks. Our D2CF approach will be based on 3 main steps, namely:

- *Pre-processing step* which consists in exploiting the timestamped data collecting during a long period in order to build a dynamic network where the nodes are the objects and the links represent the interactions between them.
- *Dynamic community detection step* which aims to find the groups of nodes that are more connected to each other than to other nodes over time.
- *Recommendation step* which consists in computing the prediction score for a given user-item pair based on the detected communities.

This report is organized as follows: Chapter 1 briefly presents the basic concepts of both recommendation and community detection. Chapter 2 enumerates community based recommendation methods related to our work. Chapter 3 details our proposed architecture for dynamic community-based recommendation. Finally Chapter 4 presents experimental results evaluating the performance of our proposed architecture.

Chapter 1

Basics on recommendation and community detection

1.1 Introduction

Recommender systems have been widely used in various domains and diverse applications, and have drawn increasing intention from different research communities such as machine learning, electronic commerce and information retrieval (Qin et al., 2010). In parallel, community detection have been attracting the interest of researchers during the last decade and it becomes one of the most prominent domain in complex network analysis (Xie, Chen, & Szymanski, 2013). This popular research topic has applications in many fields such as biology, social science, etc.

This chapter will be devoted to basic on both recommendation and community detection. Section 1.2 presents Recommender systems and focus in particular on Collaborative filtering approach and Section 1.3 gives an overview on static and dynamic community detection methods.

1.2 Recommender systems

With the vast growth of information on the Internet, recommender systems have been proposed to address the information overload problem by filtering the relevant data and suggesting items of potential interest to users. They have become fundamental applications in diverse web areas such as:

1. The *e-commerce websites* like Amazon and eBay in which recommendation systems provide an intelligent mechanism to find out items that users will probably like to buy according to their behavior history of prior purchase transactions (Schafer, Konstan, & Riedl, 1999).

2. The *video websites* like Youtube and Netflix where recommendation systems predict items that fulfill users' needs and preferences on movies, music and videos among the tremendous amount of available items (Qin et al., 2010).

3. The *online web journal* like LiveJournal and Web library which make use of recommender systems to identify articles and journals of interest to readers to ease the task of finding preferred items from a huge collection of items (Porcel, Moreno, & Herrer-Viedma, 2009).

In this section, we will first give some notations and definitions relative to recommender systems. Then, we will briefly present existing recommendation methods with a particular focus on collaborative filtering.

1.2.1 Notations and definitions

- $U = \{u_1, u_2, ..., u_n\}$ denotes the set of users in the system where n is the number of users.
- $I = \{i_1, i_2, ..., i_m\}$ denotes the set of items in the system where m is the number of items.
- *Active user*: the user that is currently interacting with the application and for whom recommendations need to be generated.
- *Target item*: the current item for which we would like to predict user's preference.
- *Browsed item*: the current item that a user has bought, visited, heard, viewed or rated positively.
- *Good item*: the recommended item that can satisfy user's needs.
- *The preference*: is a numerical value $P_{u,i}$ expressing the predicted preference or likeliness degree of item i given by the active user u.
- *Top-N recommendations*: the N recommended items that have the top highest preference value and that are not already seen by the active user.
- *User profile*: a set of user's preferences.

1.2.2 Recommendation methods

Recommendation methods can be classified into three major categories: content-based filtering, collaborative filtering and hybrid approaches.

- *Content-based approach* (Belkin & Croft, 1992) selects items based on their content along user's profile. Its principle is to recommend new items similar to the ones that the user has preferred in the past. In this approach, the system must have access to a set of items features to be able to compute similarity of items. For example, if a user has positively rated a pop song, then the system recommends to him other songs similar to pop.
- *Collaborative filtering approach* (Resnick, Iacovou, Suchak, Bergtrom, & Riedl, 1994) infers user's preferences from remaining users having similar tastes. The similarity of two users is computed using their ratings history. The recommendation process for this technique takes as input a matrix of given user-item ratings and as an output either a numerical value indicating the degree of likeliness of a certain item or a list of Top-N recommendation. Google+ circles exploit our content using a collaborative filtering approach.
- *Hybrid approach* (Soboroff & Nicholas, 1999) combines several recommendation techniques to exploit merits of each one using different hybridization strategies (mixed, weighted, cascade, ect.). Commonly, Collaborative filtering is combined with Content-based filtering since Content-based approach cannot recommend items that have no available features, while collaborative filtering predicts new items according to users' ratings.

In this work, we will focus on Collaborative Filtering (denoted CF for short) as it is the most widely implemented technology. It works well with complex objects and it proves an explainable result which is an important aspect in recommender systems.

1.2.3 Collaborative Filtering

Collaborative Filtering methods rely on a matrix of user-item ratings to predict missing user-item preferences. The more users express their preferences on items, the more accurate the recommendations become. The output is a top N recommendation list L containing items with the highest preference value for the active user or a degree of likeliness of a certain item as depicted in Figure 1.1.

The typical collaborative filtering scenarios are based on three steps:

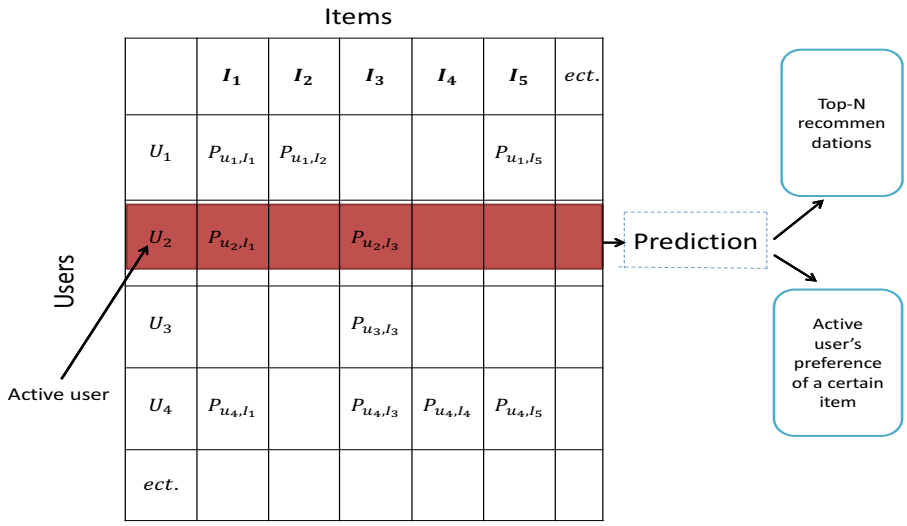

Figure 1.1: Collaborative filtering process

1. Collect preferences of users.
2. Match the active user's preferences and the preferences collection to discover users having similar taste in the past.
3. Recommend items liked by users similar to the active user.

Collaborative filtering systems are often characterized as either being memory-based exploiting users' ratings database to compare users against each other using similarity measures, or model-based in which a model is learned from the historical rating data and used to predict a user's rating for a particular item that he had not seen before. In what follows, we provide details of *memory-based* and *model-based* method categories.

Memory-based Collaborative Filtering

Memory-based methods (also called *user-based CF*) consist in using the entire users' rating data to compute the similarity between users and select the most similar ones for recommendation. User-based CF executes the following tasks to generate recommendations

for an active user:

1. Compute similarity between users based on their rating patterns (if two users rated the same items in the past then they are similar).
2. Select the most similar users to the active user.
3. Compute the preference value $P_{u,i}$ for the target item i as a weighted average of ratings assigned for i by the most similar users to the active user u. Equation (1.1) expresses $P_{u,i}$ where $\bar{r_u}$ is the average ratings of user u, $r_{u',i}$ is the rating given by user u' to item i, S refers to the most similar users to user u and $s(u,u')$ is the similarity degree between users u and u'.

$$P_{u,i} = \bar{r_u} + \frac{\sum_{u' \in S} s(u,u')(r_{u',i} - \bar{r_{u'}})}{\sum_{u' \in S} |s(u,u')|} \tag{1.1}$$

4. Extract the highest predicted preference to select the Top-N recommendations.

There are several similarity measures (B. Sarwar, Karypis, Konstan, & Riedl, 2000), we cite in particular:

- *Pearson correlation-based similarity*: *Pearson correlation* computes the extent to which two users are linearly related to each other. It is only based on items rated by both u and u'. Formally:

$$S(u,u') = \frac{\sum_{i \in I'}(r_{u,i} - \bar{r_u})(r_{u',i} - \bar{r_{u'}})}{\sum_{i \in I'}(\sqrt{r_{u,i} - \bar{r_u})^2}\sqrt{(r_{u',i} - \bar{r_{u'}})^2}} \tag{1.2}$$

  where I' is the set of items that both u and u' have rated, $r_{u,i}$ is the rating of the user u for the item i and $\bar{r_u}$ is the average rating of the co-rated items of the user u.

- *Vector cosine-based similarity*: similarity s(u,u') is computed by considering each user as a vector of users' ratings and measuring the cosine of the angle formed by these vectors. Formally:

$$S(u,u') = cos(u,u') = \frac{\vec{A} \bullet \vec{B}}{\|\vec{A}\| * \|\vec{B}\|} \tag{1.3}$$

  where A, B correspond to vectors of u and u' respectively and $\bullet$ denotes the dot-product.

Table 1.1: Users' ratings matrix

	i_1	i_2	i_3	i_4	i_5
u_1	5	4	?	3	3
u_2	4	?	3	5	?
u_3	3	3	3	?	5

Example 1.1. *Let us consider the user-item matrix of Table 1.1 composed of three users* $\{u_1, u_2, u_3\}$ *and five items* $\{i_1, i_2, i_3, i_4, i_5\}$*. Our aim is to recommend to* u_2 *the most preferred item not yet used* ($i_2 or i_5$)*. To this end, we should compute both of* P_{u_2,i_2} *and* P_{u_2,i_5}*. So, we should first compute* $s(u_2, u_1)$ *and* $s(u_2, u_3)$ *to select the most similar user to* u_2 *using Equation (1.2) as follows:*

$$
\begin{aligned}
s(u_2,u_1) &= \frac{(r_{u_2,i_1}-\bar{r_{u_2}})(r_{u_1,i_1}-\bar{r_{u_1}})+(r_{u_2,i_4}-r_{u_2})(r_{u_1,i_4}-\bar{r_{u_1}})}{\sqrt{(r_{u_2,i_1}-\bar{r_{u_2}})^2}\sqrt{(r_{u_1,i_1}-\bar{r_{u_1}})^2}+\sqrt{(r_{u_2,i_4}-\bar{r_{u_2}})^2}\sqrt{(r_{u_1,i_4}-\bar{r_{u_1}})^2}} \\
&= \frac{(4-4)(5-3.75)+(5-4)(3-3.75)}{\sqrt{(4-4)^2}\sqrt{(5-3.75)^2}+\sqrt{(5-4)^2}\sqrt{(3-3.75)^2}} = -1 \\
s(u_2,u_3) &= \frac{(r_{u_2,i_1}-\bar{r_{u_2}})(r_{u_3,i_1}-\bar{r_{u_3}})+(r_{u_2,i_3}-\bar{r_{u_2}})(r_{u_3,i_3}-\bar{r_{u_3}})}{\sqrt{(r_{u_2,i_1}-\bar{r_{u_2}})^2}\sqrt{(r_{u_3,i_1}-\bar{r_{u_3}})^2}+\sqrt{(r_{u_2,i_3}-\bar{r_{u_2}})^2}\sqrt{(r_{u_3,i_3}-\bar{r_{u_3}})^2}} \\
&= \frac{(4-4)(3-3.5)+(3-4)(3-3.5)}{\sqrt{(4-4)^2}\sqrt{(3-3.5)^2}+\sqrt{(3-4)^2}\sqrt{(3-3.5)^2}} = 1
\end{aligned}
$$

We will only consider the one most similar user to u_2*, namely* u_3*. The predicted preference* P_{B,I_2} *is therefore computed as follows:*

$$P_{u_2,i_2} = \bar{r_{u_2}} + \frac{s(u_2,u_3)(r_{u_3,i_2}-\bar{r_{u_3}})}{|s(u_2,u_3)|} = 4 + \frac{1(3-3.5)}{1} = 3.5$$

With the same manner, we compute P_{u_2,i_5} *and we obtain:*

$$P_{u_2,i_5} = \bar{r_{u_2}} + \frac{s(u_2,u_3)(r_{u_3,i_5}-\bar{r_{u_3}}}{|s(u_2,u_3)|} = 4 + \frac{1(5-3.5)}{1} = 5.5$$

As $P_{u_2,i_5} > P_{u_2,i_2}$*, then item* i_5 *is more preferred than* i_2 *and consequently* i_5 *will be in the top one recommendation list of* u_2*.*

Model-based Collaborative Filtering

Model-based methods (also called *item-based CF*) consist in using the historical rating data of users to learn a model in order to make predictions. To make a prediction for the active user u, on a certain item i, the item-based collaborative filtering uses the similarities between the rating patterns of items as follows:

1. Select items which are already rated by the active user
2. Compute similarity between target item and the selected items (if two items are liked or disliked by the same users, then these latters are similar).
3. Select the most similar items to the target item.
4. Compute the user's preference for item i using the average ratings given by the active user for the similar items as expressed by Equation (1.4).

$$P_{u,i} = \frac{\sum_{j \in S} s(i,j) \;\; r_{u,j}}{\sum_{j \in S} |s(i,j)|} \tag{1.4}$$

   Where S represents the most similar items to i, $s(i,j)$ denotes the similarity degree between items i and j and $r_{u,j}$ corresponds to the ranting of user u on item j.

5. Extract the highest predicted preference to select the Top-N recommendations.

Among the most commonly used traditional metrics, we cite *Cosine* similarity and *Pearson* Correlation (B. Sarwar, Karypis, Konstan, & Riedl, 2001).

- *Pearson correlation-Based Similarity*: similarity $s(i,j)$ between two items i and j is calculated based on users who rated both i and j. Formally:

$$s(i,j) = \frac{\sum_{u \in U}(r_{u,i} - \bar{r_i})(r_{u,j} - \bar{r_j})}{\sqrt{\sum_{u \in U}(r_{u,i} - \bar{r_i})^2}\sqrt{\sum_{u \in U}(r_{u,j} - \bar{r_j})^2}} \tag{1.5}$$

  where U is the set of users who rated both item i and j, $r_{u,i}$ is the rating of user u on item i and $\bar{r_i}$ is the average rating of the i^{th} item by U.

- *Cosine-based Similarity*: similarity is computed by considering items as vectors and measuring the cosine of the angle between these two vectors. The similarity is given by:

$$S(i,j) = \frac{\vec{i} \bullet \vec{j}}{\|\vec{i}\| * \|\vec{j}\|} \tag{1.6}$$

Example 1.2. *Consider the same ratings matrix of Table 1.1. We want to compute user u_2's prediction for item i_2 and i_5 in order to recommend the most close item to user's taste. So, we need to find the two most similar items for both i_2 and i_5 based on Pearson Correlation similarity Equation(1.5) as follows:*

$$s(i_2,i_1) = \frac{(r_{u_1,i_2}-\bar{r_{i_2}})(r_{u_1,i_1}-\bar{r_{i_1}})+(r_{u_3,i_2}-\bar{r_{i_2}})(r_{u_3,i_1}-\bar{r_{i_1}})}{\sqrt{(r_{u_1,i_2}-\bar{ri_2})^2+(r_{u_3,i_2}-\bar{r_{i_2}})^2}\sqrt{(r_{u_1,i_1}-\bar{ri_1})^2+(r_{u_3,i_1}-\bar{r_{i_1}})^2}}$$

$$= \frac{(4-3.5)(5-4)+(3-3.5)(3-4)}{\sqrt{(4-3.5)^2+(3-3.5)^2}\sqrt{(5-4)^2+(3-4)^2}} = 1$$

$$s(i_2,i_3) = \frac{(r_{u_3,i_2}-\bar{r_{i_2}})(r_{u_3,i_3}-\bar{r_{i_3}})}{\sqrt{(r_{u_3,i_2}-\bar{r_{i_2}})^2}\sqrt{(ru_{3},i_3-\bar{r_{i_3}})^2}} = \frac{(3-3.5)(3-3)}{\sqrt{(3-3.5)^2}\sqrt{(3-3)^2}} = 0$$

$$s(i_2,i_4) = \frac{(r_{u_1,i_2}-\bar{r_{i_2}})(r_{u_1,i_4}-\bar{r_{i_4}})}{\sqrt{(r_{u_1,i_2}-\bar{r_{i_2}})^2}\sqrt{(ru_{1},i_4-\bar{r_{i_4}})^2}} = \frac{(4-3.5)(3-4)}{\sqrt{(0.5)^2}\sqrt{(-1)^2}} = -1$$

This means that the most similar two items to i_2 are i_1 and i_3. Based on this, the prediction P_{u_2,i_2} is therefore computed as follows:

$$P_{u_2,i_2} = \frac{s(i_2,i_1)r_{u_2,i_1}+s(u_2,i_3)r_{u_2,i_3}}{|s(i_2,i_1)|+|s(i_2,i_3)|} = \frac{(1*4)+(0*3)}{1+0} = 4$$

With the same manner, we select the two most similar items to i_5.

$$s(i_5,i_1) = \frac{(r_{u_1,i_5}-\bar{r_{i_5}})(r_{u_1,i_1}-\bar{r_{i_1}})+(r_{u_3,i_5}-\bar{r_{i_5}})(r_{u_3,i_1}-\bar{r_{i_1}})}{\sqrt{(r_{u_1,i_5}-\bar{ri_5})^2+(r_{u_3,i_5}-\bar{r_{i_5}})^2}\sqrt{(r_{u_1,i_1}-\bar{ri_1})^2+(r_{u_3,i_1}-\bar{r_{i_1}})^2}}$$

$$= \frac{(3-4)(5-4)+(5-4)(3-4)}{\sqrt{(-1)^2+1^2}\sqrt{1^2+(-1)^2}} = -1$$

$$s(i_5,i_3) = \frac{(r_{u_3,i_5}-\bar{r_{i_5}})(r_{u_3,i_3}-\bar{r_{i_3}})}{\sqrt{(r_{u_3,i_5}-\bar{r_{i_5}})^2}\sqrt{(ru_{3},i_3-\bar{r_{i_3}})^2}} = \frac{(5-4)(3-3)}{\sqrt{(5-4)^2}\sqrt{(3-3)^2}} = 0$$

$$s(i_5,i_4) = \frac{(r_{u_1,i_5}-\bar{r_{i_5}})(r_{u_1,i_4}-\bar{r_{i_4}})}{\sqrt{(r_{u_1,i_5}-\bar{r_{i_5}})^2}\sqrt{(ru_{1},i_4-\bar{r_{i_4}})^2}} = \frac{(3-4)(3-4)}{\sqrt{(-1)^2}\sqrt{(-1)^2}} = 1$$

Items i_4 and i_3 are selected as the most similar items to i_5 and they are used to compute the preference value of u_2 for item i_5 as follows:

$$P_{u_2,i_5} = \frac{s(i_5,i_3)r_{u_2,i_3}+s(i_5,i_4)r_{u_2,i_4}}{|s(i_5,i_3)|+|s(i_5,i_4)|} = 5$$

Item i_5 has the highest predicted preference and consequently it will be in the top one recommendation list for user u_2.

Due to the nature of data used in collaborative filtering, both *user-based* and *item-based* approaches suffer from one or more weakness. In fact, there is a *cold start* problem when a new user starts with an empty profile. The *sparsity* problem occurs when available data are insufficient for identifying similar items/users. When there is an excessive information of users and items *the scalability* problem gives rise.

To outcome these problems, several recommendation methods were implemented using different techniques. We cite in particular, clustering techniques (Ungar & Foster, 1998; B. M. Sarwar, Karypis, Konstan, & Riedl, 2002), Bayesian techniques (Langseth & Nielsen, 2009; Zhang & Koren, 2007), community detection (Kamahara, Asakawa, Shimojo, & Miyahada, 2005; Qiang & Yan, 2012; Qin et al., 2010; Sahebi & Cohen, 2011), ect.

- **Clustering techniques**: based on rating of users, these techniques group users or items into clusters. Once the clusters are created, items' predictions can be made using ratings average of users pertaining to that cluster. A user can belong to more than one cluster, in this case the prediction is an average across the clusters, weighted by degrees of membership. The clustering techniques reduce the sparsity by shrinking the dimensions of user- item rating matrix and improve the scalability of the systems since the similarity is calculated only for users in the same clusters.

  Ungar and Foster (Ungar & Foster, 1998) proposed to repeat K-means clustering technique more than one time in order to group users into clusters with similar rated items and group items into clusters which tend to be liked by the same users. Then, the *Gibbs sampling* method is applied to update the clusters structure.

  Sarwar et al. (B. M. Sarwar et al., 2002) proposed to use the clustering technique to group users into clusters based on their similarities. In this approach, users belonging to the same cluster are considered as neighbors. Collaborative filtering is used in the recommendation step in which the prediction score is computed based on aggregating ratings of active user's neighbors.

- **Bayesian techniques**: consist in learning as a first step a model from the available information in the database and exploit it to generate a recommendation list. This process is practical when the user preference information changes slowly but is not suitable when user preference models are updated quickly (B. Sarwar et al., 2001).

  Langseth and Nielsen (Langseth & Nielsen, 2009) propose a probabilistic collaborative filtering model that explicitly represents all items and users simultaneously in

the same model. Each item is represented by a vector w_i and each dimension of w_i describes a unique feature of the item. Each user is represented by a vector which describes user's liking for each item feature. These vectors are incorporated in the Bayesian network in order to predict user's ratings about unrated items using his previous ratings on other items.

Zhang and koren (Zhang & Koren, 2007) introduce a bayesian hierarchical model for content-based recommendation. They model each user as a k-dimentional vector, sampled randomly from a gaussian distribution, and items as k-dimensional features vectors. Then, they incorporate data from all users in a Bayesian hierarchical model in order to learn a large number of individual user profiles. These latters are finally user to predict a label-rating of an item for a specific user.

- **Community detection techniques**: aim to exploit network interactions to construct dense groups (communities) composed of a set of nodes strongly connected among them and more weakly to the remaining of the network. In the context of recommender systems, a community is generally defined as a group of users or items that may have the same interests and properties. In this case, recommendation is only restricted on communities into which the active user belongs instead of the entire network. Recommendation techniques focusing on community detection will be detailed in Chapter 2.

1.3 Community detection

Community detection in networks becomes recently a well known problem in many fields especially for social network applications (Deitrick, Valyou, Jones, Timian, & Hu, 2013). In fact, people naturally tend to form groups within their work environment, family, friend, ect. In the last few years, the study of groups and communities becomes fundamental in mining and analysis of sociological graphs and real networks for many applications ranging from statics, computer science, biology, ect. (Chen & Yan, 2010). This can help to understand the collective behavior of users.

For a while the detection of communities within networks was related to the problem of graph partitioning. Both terms refer to the division of a network into dense groups. However, in community detection approach, the number and size of groups are unspecified, but determined by the organization of the network contrarily to the graph partitioning where the number and the size of groups should be known in advance, which reduces the efficiency of such a tool.

A community in a network is generally defined as a set of nodes strongly connected among them and more weakly to the remaining of the network. In fact, community must be dense and clearly separated from the rest of network. Based on this, many partitions could represent meaningful community structures for a given network. That's why many works such as (Gfeller, Chappelier, & Rios, 2005) have been related the significance of a community structure to the stability of a partition against random perturbation of the graph structure.

A growing number of community detection methods have been recently proposed. The goal of this section is to present the most common ones including static and dynamic approaches.

1.3.1 Notations and definitions

- $G(V, E)$: an undirected graph with V the set of vertices (nodes) and E the set of its edges (links).
- $C(V_C, E_C)$: C is a community of G where V_C is a set of nodes belonging to V and E_C is a set of all links belonging to E.
- *Community structure*: This property means that the nodes tend to create dense local structures (i.e a set of nodes strongly connected among them and more weakly to the remaining of the network).
- *Modularity*: a metric that evaluates the quality of a given community decomposition.
- *Snapshot*: an overview of the network at a given time.
- *Overlapping community*: this property means that the nodes can belong to more than one community in the same time.
- *Edge-betweenness*: a metric that represents the number of shortest paths between two different nodes of the graph, which run along this edge.
- *k-clique*: a complete sub graph of k nodes connected with k(k-1)/2 possible links.

1.3.2 Static community detection methods

Static community detection algorithms are applied in graphs called static networks or network of interactions constructed by aggregating all observed interactions over a period of time and representing it as a single graph. These graphs are used as a tool for modeling

complex phenomena by modeling the actors of the phenomena by the vertices and the interactions between them, by links or edges between the vertices. Formally, there is a graph $G = (V, E)$ where V is the set of vertices and E is the set of links, the aim is to find a partition of the vertices of the graph $P = (C_1, C_2, ..., C_k)$, where each subset C_i represents a community without knowing in advance neither the number nor the size of different communities. Figure 1.2 represents an example of community structure in graph.

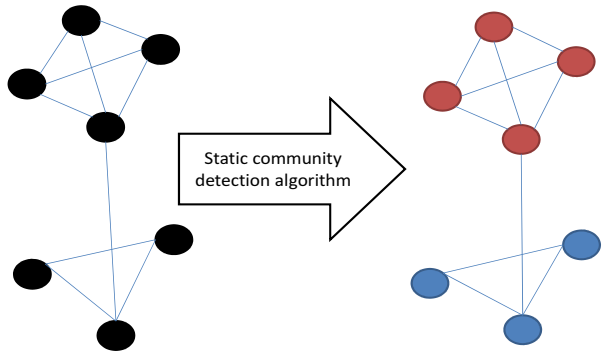

Figure 1.2: An example of static graph with community structure

There exists several works that attempt to detect communities on static networks. We intend to present two analytical categories of static community detection, namely non overlapping and overlapping community detection methods.

Non-overlapping community detection

The first idea of non overlapping communities was to find a simple partitioning for a given network: each node in the network belongs to only one community and there is only one possible community decomposition. The first algorithm has been proposed by Girvan and Newman (Newman & Girvan, 2004) by considering that all nodes belong to the same community, then edges are successively removed according to their *Edge-betweenness* values until obtaining a disconnected graph (i.e each node represents one community). A dendrogram typically created in the end of this process, which represents the different possible cuts for the graph (i.e starting with a community in the root, two communities in the next step, until each node belongs to its own partition). The resulting number

of partitions depends on the dendrogram cut. An example of a dendrogram is shown in Figure 1.3. In this step, Girvan and Newman have introduced the *Modularity* metric to

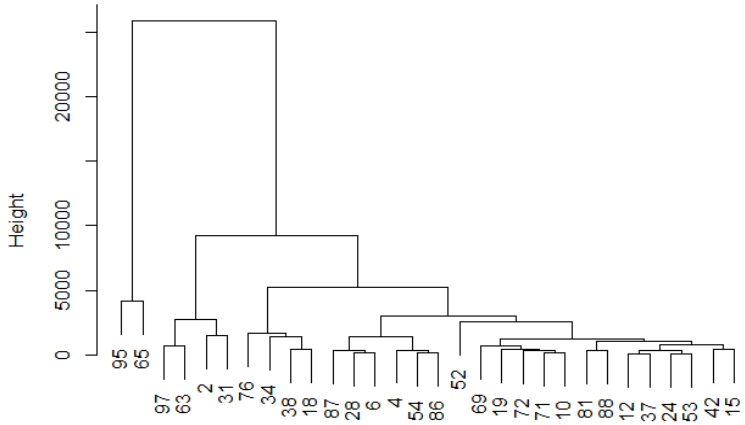

Figure 1.3: An example of dendrogram

determine the optimal cut level in the dendrogram. A high value of modularity indicates a better partitioning of the network into communities.

Motivated by the modularity definition, many works have been proposed. The idea consists in finding the partition corresponding to the maximum value of modularity. The most popular heuristic using agglomerative methods is Louvain method (Blondel, Guillaume, Lambiotte, & Lefebvre, 2008). The algorithm uses a greedy optimization method that attempts to optimize the modularity of a partition of the network in a local way. The idea is to consider each node as a single community. Then, the gain in weighted modularity that can be obtained by integrating each community in its neighbor community is computed in order to aggregate nodes of communities that offer maximal gain.

These steps are repeated iteratively until obtaining a stable modularity value. At the end of each iteration, the algorithm yields the partitioning scheme and considers formed communities as new nodes. This method is used for very extremely large networks.

In (Brandes et al., 2008), it has been proved that modularity optimization is an NP-hard problem. In fact, the resulting values of modularity as well as the community memberships are affected by network topology (i.e the order in which the vertices in the network are processed). Besides, modularity maximization suffers from the problem of *resolution limit*, when communities smaller than a certain size cannot be detected and they should consequently merged with other communities to just obtain the expected size. A large value of the modularity maximization does not necessarily reveal the optimal division.

Overlapping community detection

Overlapping community structures can be observed in many real-world networks such as social networks where an individual usually belongs to different circles at the same time (work colleagues, relatives, sport associations, etc.) as shown in Figure 1.4 and biological networks where a protein may belong to several protein complexes simultaneously. In such networks, communities overlap such that nodes may belong to more than one group, and these groups form the so-called overlapping communities.

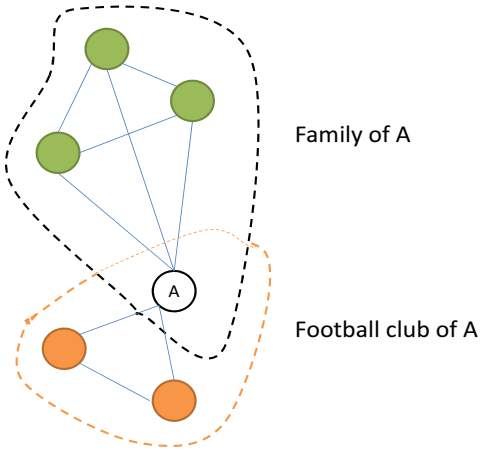

Figure 1.4: Example of overlapping communities

Finding such overlapping communities is a challenging problem and is not supported by community detection algorithms discussed above. The basic technique to detect overlapping communities is *Clique-Percolation Method (CPM) (Palla, Derényi, Farkas, & Vicsek, 2005)*. CPM is based on the assumption that a community is the largest connected subgraph obtained by a k-clique with all k-cliques connected to it. In fact, the algorithm works in two steps, first all k-cliques should be found then an aggregation step is ensured for nodes of cliques sharing at least $k-1$ vertices, whithin the same community. Figure1.5 presents an illustrative example of CPM.

Overlapping communities have been also studied by (Gregory, 2010). In fact, a *Community Overlap propagation* algorithm (COPRA) has been proposed based on the principle of label propagation. The typically idea of the Label Propagation Algorithm (LPA) was initiated by Raghavan et al. (Raghavan, Albert, & S.Kumara, 2007), and it consists in assigning each vertex with a unique label. Then, each vertex, repeatedly, updates its label

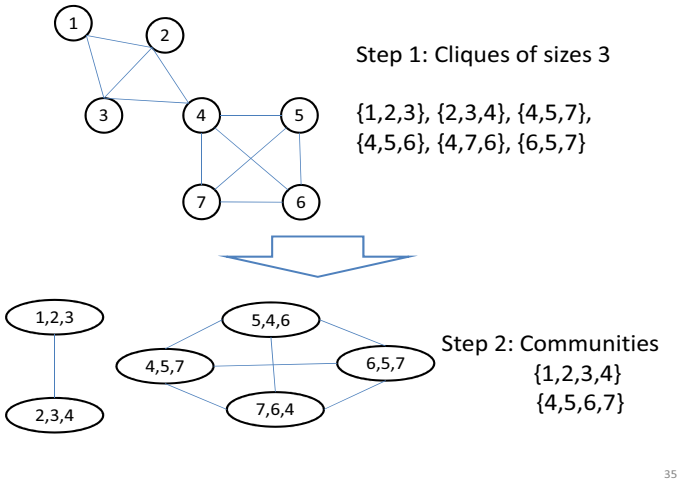

Figure 1.5: An illustrative example of CPM

by replacing it by the label used by the greatest number of its neighbors. After several iterations, the same label tends to become associated with all members of a community. An enhanced version of LPA was proposed by the authors in which each node is labeled with a set of pairs (c, b) where c is a community identifier and b is a belonging coefficient indicating the strength of the membership of the node in the community c.

1.3.3 Dynamic community detection methods

Dynamic networks, also called time varying graphs can be either a set of independent snapshots taken at different time steps (Hopcroft, Khan, Kulis, & Selman, 2004; Palla, Barabasi, & Vicsek, 2007) or a set of linked snapshots (Mucha, Richardson, Macon, Porter, & Onnela, 2010) or temporal network that represents sequences of structural modifications of network over time (Cazabet & Amblard, 2011). In fact, dynamic networks have had to change and evolve over time, links and nodes can also appear and disappear. Figure 1.6 shows the basic events that may occur in the life of a community: a community may born (when it emerges without predecessor) or dead (when it disappears without successor) or merges with another community (when its join together to form a new community) or splits (when it splits into several new communities) or grows (when it gains new members) or contracts (when it loses members).

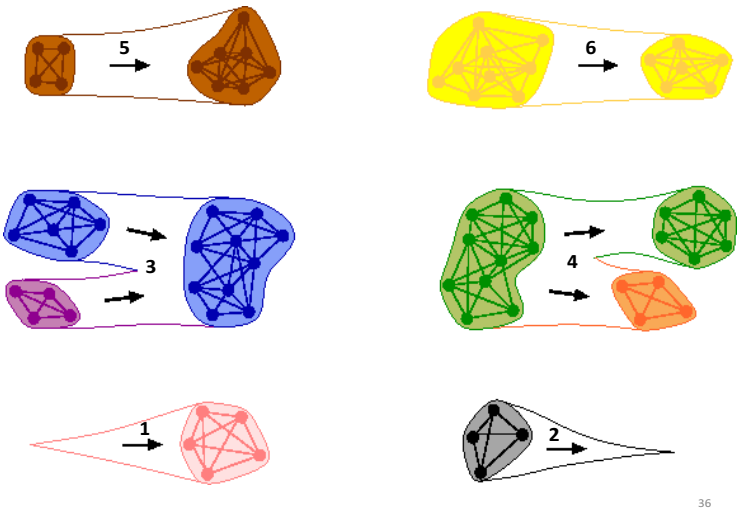

Figure 1.6: The community basic events

Recently, a lot of research has been done on these networks to detect dynamic communities. We present two main categories of dynamic community detection methods, methods based on snapshots and methods based on temporal networks.

Methods based on snapshots

As each snapshot is a static graph, the basic approach to detect community structure in dynamic networks is to use a classical algorithm on each snapshot more or less independently, then, match the resulting communities in order to characterize the evolution of communities over time as depicted by Figure 1.7.

Hopcroft et al. (Hopcroft et al., 2004) are the pioneers of handling community structure in dynamic networks. The idea consists in decomposing the dynamic network into a set of snapshots where each snapshot corresponds to a single point of time. Then, they applied a static community detection algorithm on each snapshot using an agglomerative hierarchical method based on cosine similarity and they matched the communities detected among different snapshots. This method is not able to deal with overlap communities which is considered an important property.

In (Palla et al., 2007), the authors proposed to adapt the static version of *Clique Percolation Method* (CPM) discussed above to detect overlapping communities in Dynamic networks. The detection process can be described as follows:

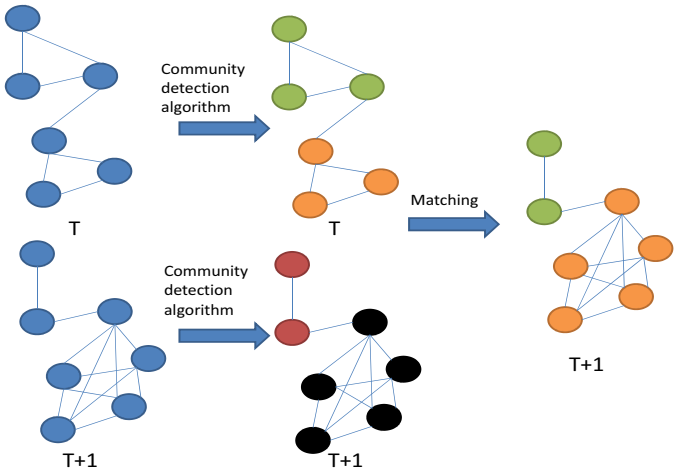

Figure 1.7: Dynamic community detection principle on a set of snapshots

1. Define a succession of snapshots taken at different time steps.
2. Use Clique Percolation Method (CPM) to extract communities at each time step.
3. For each consecutive time steps t and $t+1$ we construct a joint graph U consisting of the union of links from the corresponding two networks (See Figure 1.8).
4. Use Clique Percolation Method (CPM) to extract the community structure of this joint network U.
   (a) If a community in the joint graph U contains a single community from t and a single community from $t+1$, then they are the same.
   (b) If a community in the joint graph U contains a single community from t and two communities from $t+1$, then a splitting process is produced.
   (c) If a community in the joint graph U contains two communities from t and a single community from $t+1$, then a merging process is produced.

Note that CPM communities can only grow, merge or remain unchanged.

Methods applying static algorithms on snapshots cannot cover the real evolution of community structures over time. Analyzing each snapshot separately presents a problem due to the instability of community detection algorithms (i.e they are often not deterministic and few modifications, or even no modifications of the input network may lead to

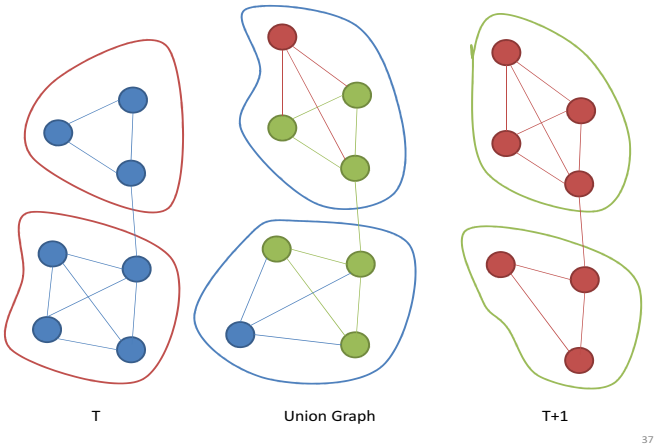

Figure 1.8: A union graph

many changes in the resulting community structure). Another idea has been explored using snapshots consists in taking into account the obtained partitions in the stage t during the community detection process of the stage $t+1$. The general principle of this approach is displayed on Figure 1.9.

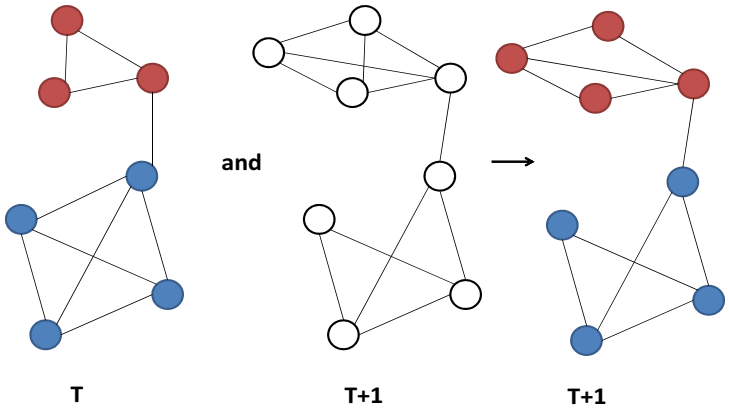

Figure 1.9: Community based on structures of current and previous snapshots

This principle has motivated several researches, we cite in particular (Chakrabarti, Kumar, & Tomkins, 2006) in which the authors presented a new framework for evolutionary

clustering. The clustering produced during a particular time-step should not only rely on the current data but also remain faithful to the clustering of the previous time-step. To this end, they have modified the quality function of the classical community detection to integrate evolution and obtain a sequence of more interesting clustering for the dynamic networks.

Methods based on Temporal networks

Since the temporal information can be also coded in the graph itself, the network evolution is not considered as a succession of snapshots anymore but rather as a succession of modifications on the network. The community detection idea consists then, in taking into consideration, where the latest modification has been carried out on the network to update existing communities. The general principle of this approach is shown on Figure 1.10. New studies have exploited such representation of data that takes into account all temporal changes.

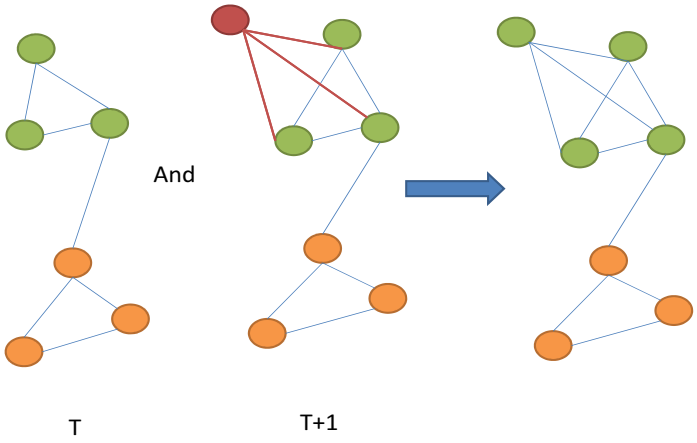

Figure 1.10: The principle methods using temporal network

We cite for instance (Nguyen, Dinh, S.Tokala, & M.T.Thai, 2011) where authors presented a two-phase framework for detecting, updating and tracing the evolution of overlapping communities in dynamic network. The first phase consists in discovering all possible basic communities in network by extracting at first all possible densely connected subgraphs of the network and then combining those who share a significant substructure together. In the second phase all possible changes (add or remove of nodes and edges) have been considered to update the initially basic community structure.

In (Cazabet & Amblard, 2011), the authors proposed an *Intrinsic Longitudinal Community Detection* (iLCD) algorithm which is capable of detecting both static and temporal communities in large networks. The *iLCD* algorithm uses a longitudinal detection of communities in the whole network in the form of a succession of structural changes. Details of this algorithm will be presented in Chapter 3.

1.4 Conclusion

This chapter presented the two main concepts for recommender systems and more especially collaborative filtering approach and community detection. We classified relevant previous studies of community detection into static and dynamic categories. Next chapter is devoted to community based recommendation methods related to our proposed attempt in this work.

Chapter 2

Community-based Recommendation

2.1 Introduction

Recommender systems have become extremely common in recent years. There is a lot research that attempt to enhance and improve the accuracy and the performance of the existing recommendation by incorporating community detection techniques. In fact, the community structure may help the system to personalize recommendations provided to users by focusing on its communities instead of the whole network. The various properties of communities may be relevant because understanding the underlying structure of these groups may have a major impact on users' behaviors and pave the way to mapping emergent semantics in recommender systems. Communities detected in static networks are extremely different from the ones extracted from dynamic networks. Consequently, a considerable impact on recommendation is envisaged depending on the network. This chapter is organized as follows: Section 2.2 presents static community-based recommendation and Section 2.3 is dedicated to the dynamic aspect of community-based recommendation.

2.2 Static community-based recommendation

The general principle of static community-based recommendation consists in detecting communities in static networks and recommending items to the target user depending to which community he pertains. In literature, there are several recommendation approaches dealing with static community detection. We cite in particular (Kamahara et al., 2005) where authors have proposed a community-based recommendation approach in which a

user can find new and unexpected recommendations. Firstly, they proposed an algorithm to cluster users into virtual groups which represent the different aspects of users' interests. Then , they calculated the similarity between target user and communities in order to discover the user's tastes. A user can belong to various communities in the same time. The recommendation of user i for item j is calculated based on the target user's community members using one of the mixed approaches of content-based and collaborative filtering. The list of recommended contents is generated in order of the higher recommendation value.

In the same context, Qin et al. (Qin et al., 2010) have proposed a recommender system for YouTube based on communities extracted from an network of reviewers. This latter is composed of a set of nodes representing video and a set of edges such that an edge is drawn between two nodes if the same reviewer comments on both of them. These edges are weighted by the number of common comments. Once the YouTube recommendation network is build, the authors choose to apply *Clique Percolation Method* (CPM) community detection algorithm (Palla et al., 2005). Then, they proposed a utility value for each node that captures its importance in the network to recommend videos locally and globally. The global approach recommends videos having a higher utility values while the local approach recommends other videos connected to the one watched by the target user and pertained to the same community.

Recommendation-based on community detection can be also useful to provide a solution to the cold start problem (Sahebi & Cohen, 2011). The basic idea of this work is to use communities, extracted from different dimensions of social networks (e.g. Facebook dimension, Twitter dimension, ect.) to capture the similarities between these dimensions and accordingly help recommendation systems to find latent similarities. To this end, they used modularity-based community detection method for multi-dimensional networks (Tang, Wang, & Liu, 2009) to identify hidden structure shared across dimensions. Therefore, they considered only users within a user's community to compute the predicted rating of each item to enhance the traditional collaborative filtering approach in terms of space and time.

Qiang et al. (Qiang & Yan, 2012) model the recommendation system as a bipartite graph composed of nodes representing users and items and edges depicting rating information as shown in Figure 2.1.

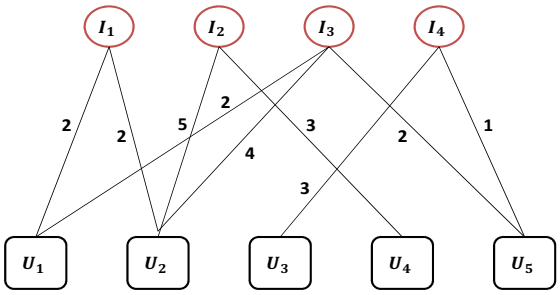

Figure 2.1: Example of a bipartite network

Then, they proposed an extension of Label Propagation Algorithm (LPA) (Raghavan et al., 2007) to detect overlapping community structure. The main idea of recommendation consists in searching the nearest neighbors of the target user in their communities using *Pearson similarity* measure, then recommending items according to ratings of candidate users.

More recently, Zhao et al. (Zhao et al., 2013) proposed a user community-based recommendation for twitter. The main of their work is to recommend a set of users to follow by target user. The main idea is to identify communities of users having similar influence as well as interests in a uni-directional social network and use these communities in recommendation to reduce data sparsity. To this end, they utilized the follower-followee relationships to model the twitter graph (see Figure 2.2). Then, they employed the LDA-based method (Blei, Ng, & Jordan, 2003) to discover communities and finally applied matrix factorization (Koren, Bell, & Volinsky, 2009) to each community to generate a list of candidate followees. The output is therefore the top-k followees to recommend to target user chosen according to their scores. We note that a target user may belong to more than one community.

In (Fatemi & Tokarchuk, 2013), authors proposed a community-based social recommender system to provide personalized recommendations for both individuals and groups. They used the implicit relationships between items derived from the direct interactions of users . In this work, movies are presented as nodes and are linked if at least one reviewer has commented and edges are weighted by the number of common reviewers between a movie pair. After building the network of items, the Louvain method (Blondel et al., 2008) has been exploited to identify groups of movies. These latters are then used for

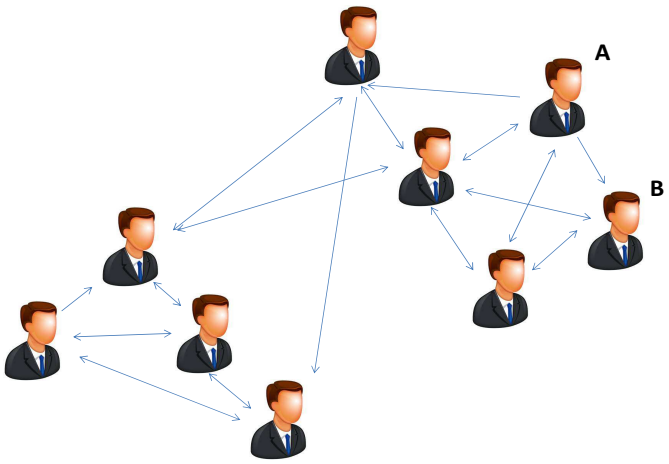

Figure 2.2: Twitter graph representation (A− >B: A follows B, A< − >B: A is a followee of B and vice versa

recommendation according to the movie's communities.

In (Wen, Liu, Zhang, Xiong, & Cao, 2014), authors proposed a community-based recommendation approach using a bipartite graph composed of users and items linked by users preferences. In fact, latent communities are at first identified using an improvement of traditional similarity measures based on ratings. Then, the recommendation list is generated based on ratings given by the other members of the communities to which target user pertains.

These methods only deal with static networks derived from aggregating data over all time, or taken at a particular time. Such aggregation can radically misrepresent the existing community structure. Besides, information can be lost since real-world networks are always evolving over time especially users preferences in social networks. For these reasons, new studies has exploited this temporal information in order to identify efficient communities and track their evolution in dynamic network. This will be the focus of the following section.

2.3 Dynamic community-based recommendation

Modeling temporal dynamics is considered as a key factor for designing both of recommender users preferences model and recommender systems. In fact, the inclusion of temporal patterns is proved invaluable in improving quality of prediction and consequently lead to significant accuracy gains (Yehuda, 2009).

Despite the high impact of temporal effects on user's preferences, the subject attracted a quite negligible attention in community-based recommender systems literature.

A first attempt has been established by Song et al. (Song et al., 2006). In fact, authors have proposed a community-based dynamic recommendation approach (CBDR) that takes into account content semantics of documents, evolutionary patterns and users communities, simultaneously. The aim is to capture dynamic patterns of users' interests and model users' preferences over time. To this end, *Time-Sensitive Adaboost* algorithm has been proposed to adapt users' evolving interests based on the outputs of the following steps:

1. *The content analysis step*: consists in recognizing documents topics from titles and abstracts.
2. *The dynamic patterns analysis step*: consists in tracking the dynamic patterns existing in both documents and user's behaviors.
3. *Community construction step*: consists in building communities based on the fact that people from the same department of the same company tend to have similar interests.
4. *Recommendation step*: provides a recommendation list of documents for all users pertaining to the same community.

This approach is limited since it uses a manual method to identify communities, which is not efficient especially when we deal with strongly evolving and large networks.

More recently, Abrouk et al. (Abrouk, Gross-Amblard, & Cullot, 2010). proposed a community detection method that relies on fuzzy k-means clustering to automatically construct users' communities. In fact, they invoked the clustering method from time to time to dynamically detect the users' interests over time. This allow to update the current user's preferences. Similarity used in this method is computed based on the resources manipulated by users. Then, they exploited these formed communities to determine user's preference for new items with regard to the updated users' ratings.

In (Hönsch, 2011), the author proposed an article recommender system for new portals in which virtual communities are extracted from the keyword relatedness graph, created from keywords deduced from previously accessed articles. The detection of user' interests is repeated continuously in subsequent time intervals in order to keep pace with the dynamics of new portals. Then, they introduced a latent analysis of content to determine the relatedness between communities. Recommendations come from other community members using collaborative filtering to find the most interesting unseen content for the particular user.

In both the previously presented methods, applying clustering techniques from time to time cannot cover the real evolution of community structure over time. Indeed, several structural changes may occur and lost, without being detected. Besides, the temporal complexity of these methods increases in large networks.

To cover up these problems, we propose in the following chapter a global architecture that takes advantage from the temporal information across users along multiple time points to provide an efficient personalized recommendation for users.

2.4 Conclusion

This chapter surveys previous works of community based recommendation which are similar to our attempt to include community detection as part as recommendation. Community detection method provide a powerful tool for recommendation by focusing on the group into which the target user pertains instead of the whole network. However, almost of these works only deal with static networks. The few other works that attempt to take into account the dynamic aspect of networks as well as communities have not used a dynamic community detection algorithm. Based on this we have proposed in the next chapter a novel approach for dynamic community based recommendation.

Chapter 3

Proposed architecture for Dynamic Community-based Recommendation

3.1 Introduction

Delving into the dynamic aspects of network behavior has become a necessity especially with the rapidly growing complex networks, such as social networks. Representing the dynamic aspect of users' interests is crucial in recommendation systems also, but it seems harder to ensure with the current community-based recommendation methods. Indeed, all these methods extract communities from the network at a given time (static network) or from time to time and this is usually hard because it is not easy to recognize the communities of t at $t+1$. To this end, we propose a general architecture to automatically detect dynamic communities using a dynamic community detection algorithm and incorporate these ones in the recommender system based on the fact that if we enhance the quality of the communities passed as input, the recommendation will be consequently enhanced. Main results relative to the proposed method have been accepted for publication in 2^{ed} International Workshop on Dynamic Networks and Knowledge Discovery at ECML PKDD 2014.
Section 3.2 will detail the different steps and parameters of the proposed architecture.

3.2 Proposed architecture

Our target is to exploit the temporal information in order to enhance recommendation based on community structure. We propose architecture, called *Dynamic Community-based Collaborative Filtering* denoted by (D2CF for short) in which the users' interests are tracked over time.

D2CF comprises three main steps. The first step is to build the dynamic network from time-stamped data. In the dynamic network the items and their interactions are represented respectively by nodes and edges. The items interactions are modeled based on the co-rating relationship learned from users profiles. These edges can be either inserted or removed in order to represent the evolution of users' interests over time. The second step applies a dynamic community detection algorithm to extract communities from dynamic network model prepared in the above step. A community is a group of items in which several common users are often interested over time. This pattern is learned from the users' ratings for items. The last step consists in recommending for user, the most likely items that can interest them based on certain categories given by the formed communities. Figure 3.1 illustrates the overall flow of our proposed D2CF architecture. Section 3.2.1 presents the pre-processing step. Section 3.2.2 is devoted to the dynamic community detection step and finally the recommendation step will be detailed in Section 3.2.3.

3.2.1 The pre-processing step

This step consists in building a dynamic network, in which the evolution of the users' interests over time is represented. User's interests (i.e. users profiles) means his preferences which are gathered by observing the set of items that this user looked at or ranked. We assume that:

- If a user does not rate the item, then this latter is not yet watched by him.
- If a user gives a rating superior to 2 for an item, he/she is then interested in this one. So, we will consider only the instances where the ratings values are upper than 2 to model the users' interests in a time varying graph.

It is important to note that in this work, we suppose that the nodes are the items and not the users and hence communities are groups of items. The interactions between nodes are modeled based on the co-rating relationships. Indeed two nodes interact with each other, if at least one user gives the same rating to both of them in the same time. In other words,

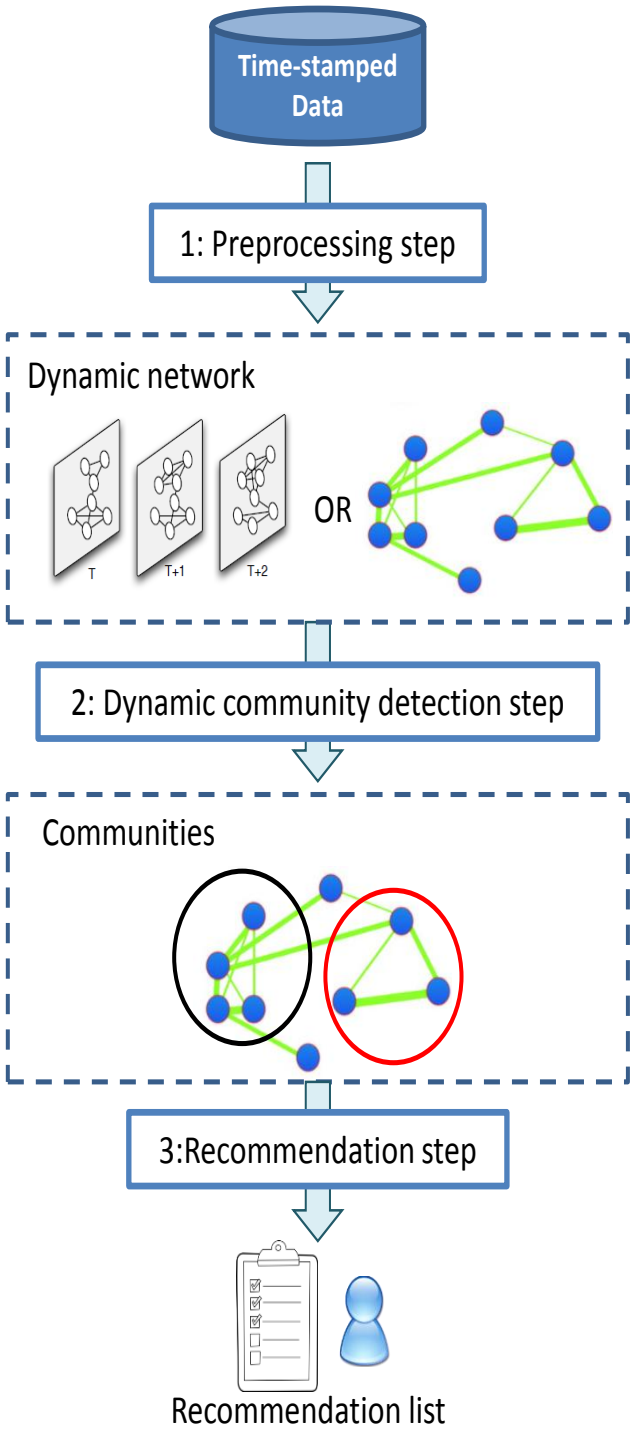

Figure 3.1: Proposed D2CF architecture

Table 3.1: Example of ratings database

UserID	**ItemID**	**Rating**$(1-5)$	**Timestamp**
u_1	i_1	3	12/03/2014
u_1	i_2	3	12/03/2014
u_2	i_2	1	14/03/2014
u_3	i_1	2	17/03/2014
u_4	i_3	5	17/03/2014
u_5	i_1	4	14/03/2014
u_5	i_2	4	14/03/2014

if the user is interested in two different items in the same date, an interaction is created between them. An interaction can be defined as follows:

Date itemID1 itemID2

At this stage, we aim to prepare the list of network changes over time. To this end, we have adapted the method of temporal network building (Cazabet, 2013) to our case of work. An edge is established between two nodes if these ones have interacted with each other at least N times over a period of P days. The estimation of N and P values varies from one network to another and it depends not only on the network topology but also on the information that we want to extract in order to create semantic links between the network nodes. For instance, if we look for an extremely strong links in the mobile communication network, we need to find at least 3 interactions (calls) over 7 days to maintain the link between two nodes (mobile devises) else if we analyze a friendship network we need at least 1 interaction (texto message) over 30 days to maintain the link because the texto interactions are less intense than the calls interactions. If over a period of P days, there are fewer than N interactions between two nodes, the edge will be automatically removed. The removal and the creation of an edge in the dynamic network depends on both of the number of interactions between two items (i.e. the number of common users who are interested in both of them) and the time aspect.

Example 3.1. *Let us consider a time stamped ratings database that contains three items* $\{i_1, i_2, i_3\}$ *and five users* $\{u_1, u_2, u_3, u_4, u_5\}$ *. Each user must rate at least two items. The given ratings are represents in Table 3.1. We suppose in this case that* $N = 2$ *and* $P = 3$*: we should have at least two interactions between a pair of node over a period of three days to create an edge between them. User* u_1 *rates* i_1 *and* i_2 *with the same rating (*3*) on March*

12th, so the first interaction between i_1 and i_2 is created: 12/03/2014 i_1 i_2
User u_5 rates i_1 and i_2 with the same rating (4) on March 14th, so the second interaction between i_1 and i_2 is created: 14/03/2014 i_1 i_2
According to this, an edge is established between i_1 and i_2 on March 14th. If no other one interaction will be created between i_1 and i_2 in the coming 3 days, the edge will be automatically removed on March 17th.

Once this list is prepared, these changes (edges removal and edges creation) can be either represented in the same graph as a temporal network or in different graphs as a sequence of snapshots. In the first case, the network evolution is considered as a succession of network modifications. The idea is to modify the existing communities according to the latest modifications performed on the network. In the second case, the network evolution is considered as a succession of snapshots where each one represents the network state at a particular time step. Indeed, if we suppose that S_0 is the initial snapshot given at t_0 and S_1 is the snapshot given at t_1, S_1 must contain all the changes that have been occurred from t_0 to t_1.

Example 3.2. *Consider a database that contains four items $\{i_1, i_2, i_3, i_4\}$ and a list of network changes which is described as follows:*

14/03/2014	+	i_1	i_2
14/03/2014	+	i_1	i_3
15/03/2014	+	i_3	i_2
16/03/2014	+	i_4	i_1
18/03/2014	−	i_1	i_2

We consider a snapshot for each day, and Figure 3.2 shows the sequence of snapshots corresponding to this list. However, if we want to build the temporal network, we need just to consider S_0 and the corresponding list of changes.

The resulting dynamic network of items is considered as a generic model that represents the evolution of users interests over time.

3.2.2 Dynamic community detection step

Once the dynamic network (temporal network or a sequence of snapshots) is constructed in the pre-processing step, we are now able to use it as input to a dynamic community

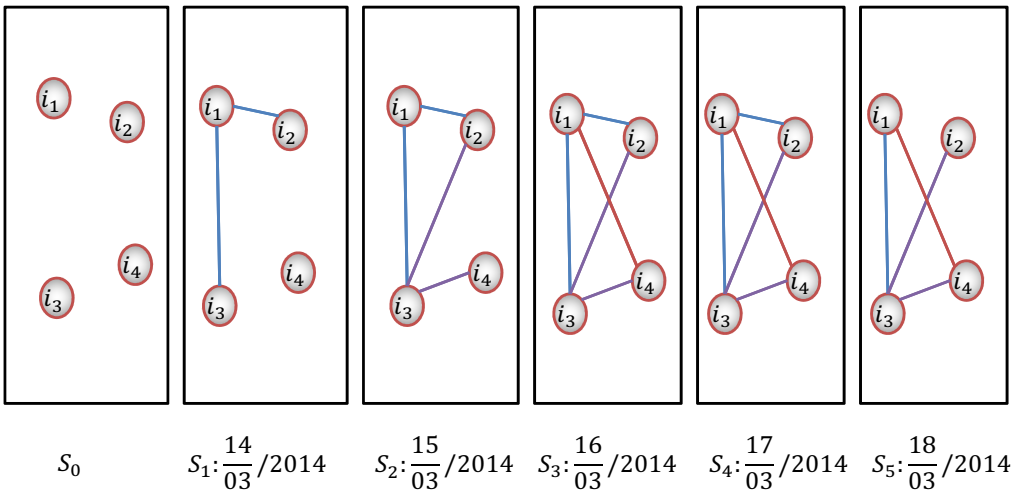

Figure 3.2: A sequence of snapshots

detection method. In this network, the nodes are the items and the edges represent the links between them. If we consider two nodes i_1 and i_2, which are strongly connected, the edge established between i_1 and i_2 means that these two items have at least N users who are interested in both of them in the same time, over a period of P days. Based on this, we can define a community as a set of items that are extremely related with each other physically (strongly connected) and semantically (a learning pattern of items that tend to have the same interests of several users over a period of time). The advantage here is that a community is not restricted to item-related topics but it contains various topics (e.g. Horror, Comedy, Romance, ect.). If the edge between i_1 and i_2 is removed, then the items are no longer related.

The novelty in this work is to apply a dynamic community detection algorithm that takes into account the evolution of the network over time. In fact, by studying the evolution of the network behavior, we can obtain a more appropriate community structure. We consider for instance the evolution on two steps (t_1 and t_2) of the small graph composed of five nodes represented in Figure 3.3. If we only have a simple view on the last step (t_2), we will not be able to identify the appropriate community structure in the network. This is mainly due to the fact that this network is strongly connected, since it merges the whole observed links during the studied period. However, if we have an overview of the entire evolution process, the predicted community structure will be closer to the real situation.

For instance in our example, if we consider steps (t_1) and (t_2), we will be able to say that we probably have two communities (i.e. (abc) and (de)).

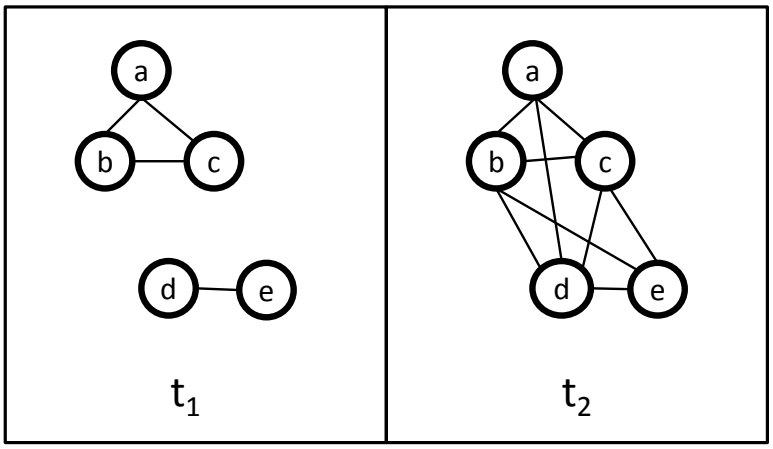

Figure 3.3: Example of a network evolution

As detailed in Chapter 1, there are many algorithms for finding communities in dynamic network. To insure this step, we choose to use the *iLCD* algorithm (Cazabet & Amblard, 2011).
The basic idea of *iLCD* algorithm was inspired from the multi-agent systems. The general principle of the algorithm is essentially based on local computations. Indeed, the communities are able to perceive only the nodes which they contain and then can interact only with other communities with which they have at least a node in common. In other words, the vision of the communities is then limited to its local environment and it has no global knowledge of the network. The authors consider the community as an autonomous entity that takes or not the decision to integrate a node based only on their local view of the network. Every modification (edge adding or deleting) in the network can lead to the creation of new communities, or the disappearance of the existing communities or the fusion with neighboring communities. The principle of the algorithm can be summarized in the following way:

- For any link addition (i,j) in the network, if i is in a community C and j does not belong to C, iLCD check if j must be integrated in C. We find then the set of these newly-formed communities by the appearance of this new link. Only the new communities which were not included in an existing communities are conserved.

- For any deletion of a link, if this edge is found within a community, iLCD check if

the concerned community C loses one or many nodes which may lead to self division.

- After every modification of the network, if one or many communities have been modified, iLCD check if these ones may be fused with other communities. The candidate communities are those which share some nodes with the modified communities.

Algorithm 1 outlines the iLCD pseudo-code where T_e is a sequence of modification in the network defined by (i, j, a, t). i and j are the nodes affected by the modification, a is the modification (edge creation or edge deletion) and t represents the modification instant.

Algorithm 1 iLCD pseudo-code

Require: : T_e (a set of networks modifications)
Ensure: : MC (a set of communities)

for $(i, j, a, t) \in T_e$ **do**
  $MC \leftarrow \emptyset$
  if $a = creation$ **then**
    for $C \in cs_i$ **do**
      if CROISSANCE(j, c) **then**
        $V_c \leftarrow V_c \cup \{j\}$
        $E_c \leftarrow E_c \cup \{\{j, k\} \in V : k \in V_c\}$
        $MC \leftarrow MC \cup \{C\}$
      end if
    end for
    for $C \in cs_j$ **do**
      if CROISSANCE(i, c) **then**
        $V_c \leftarrow V_c \cup \{i\}$
        $E_c \leftarrow E_c \cup \{\{i, k\} \in V : k \in V_c\}$
        $MC \leftarrow MC \cup \{C\}$
      end if
    end for
    if NAISSANCE$(i, j) \neq \emptyset$ **then**
      for $C \in$ NAISSANCE(i, j) **do**
        if $C \notin C_x : C_x \in (cs_i \cup cs_j)$ **then**
          $P \leftarrow P \cup \{C\}$
        end if
        $MC \leftarrow MC \cup \{C\}$
      end for
    end if
  else
    for $C \in (cs_i \cap cs_j)$ **do**
      $CRIME \leftarrow$ CONTRACTION_DIVISION(C, i)
      for $C_2 \in CRIME$ **do**
        if $j \in C_2$ **then**
          $CRIME \leftarrow$ CONTRACTION_DIVISION(C, j)
        end if
      end for
      $P \leftarrow P \setminus \{C\} \cup CRIME$
      $MC \leftarrow MC \cup CRIME$
      if MORT(C) **then**
        $P \leftarrow P \setminus \{C\}$
      end if
    end for
  end if
  for $C \in MC$ **do**
    for $C_2 : V_{C_2} \cap V_C \neq \emptyset$ **do**
      $P \leftarrow P \setminus \{C, C_2\} \cup$ FUSION(C, C_2)
    end for
  end for
end for

CROISSANCE, CONTRACTION_DIVISION, FUSION, and MORT are the key functions of *iLCD*.

- CROISSANCE(i, C): This function indicates if a node i must be added or not to a community C.
- CONTRACTION_DIVISION(C, i): This function indicates if a node i should be in a community C, else it returns the resulting communities by deleting i.
- MORT(C): This function indicates if a community C should disappear from the network.
- FUSION(C_{old}, C_{new}): This function returns the community resulting by fusion C_{old} and C_{new}.

The implementation of these functions is ensured based on the three following metrics:

- *Representativeness*$Rp(i, C)$: a metric that indicates to what extent a node i is representative of a community C.
- *Seclusion*$CI(C)$: a metric that represents the degree of separation of a community C from the rest of network.
- *Potential belonging*$FA(i, C)$: a metric that quantifies the belonging force of a node i to a community C.

This choice is justified by the fact that *iLCD* detects communities in both static and dynamic networks depending on disposal data. Moreover, this algorithm takes into account the evolution of the network which enables to identify the dynamical communities more accurately. Such community detection can be more powerful too, because this analysis matches with the reality of networks. This algorithm deals with both large evolving networks and the overlap of communities. The communities extracted are "atomic" in the sense that there are not other relevant self communities inside of itself. Finally, all operations in *iLCD* algorithm are made at a local level, which can allow to ensure that the complexity will not grow exponentially with the size of the network but more linearly with the number of edges.

This step reveals interesting properties:

- The dynamic of the communities which is very important to deal with the real-world network.
- The diversity of the communities with regards to the corresponding items.

3.2.3 Recommendation step

In this step, the learned patterns (communities) will be exploited to help the recommender system to predict the users' future interests based on certain categories given by these communities. In fact, the items pertaining to the same community are related somehow to each other by sharing common users' preferences. For instance, if a user is interested in i_1, then he has a high chance to be interested in i_2 since i_1 and i_2 pertain to the same community and i_2 is not yet rated by him.

Firstly, a target item should be identified for each user. The target item in this case is the item in which the active user is more interested (The item with the highest rating value). The items that belong to the communities of target item and that are not yet rated by the active user are selected as candidate items. The list of top k recommended items for the active user contains the k candidate items that have the highest predicted preferences (e.g. the top one recommended item for the active user is the item that has the highest prediction value between all candidate items).

Our objective is to compute the preference of the user u on the candidate item i based on the items that belong to the communities of i. By doing this the recommendation is restricted to the communities to which the candidate item pertains. In fact, instead of computing user's preference, taking into consideration all items present on the whole network to discover the most similar to the candidate item, we only rely on the items that pertain to candidate item's communities extracted from the dynamic network as presented in Figure 3.4.

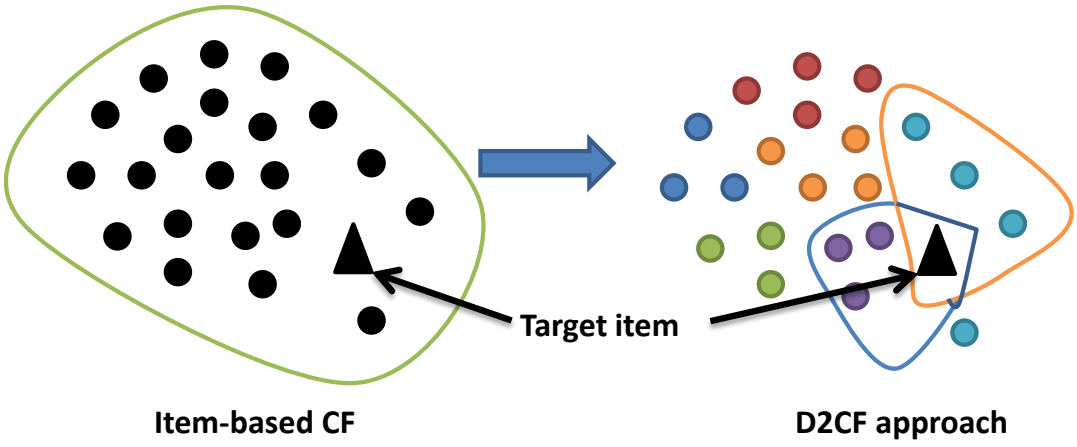

Figure 3.4: The principle of user's preference computation taking into account the community structure in the network

To compute the predicted preference of the active user on each candidate item, we propose an adaptation of the traditional item-based Collaborative filtering method (See Equation 1.4). Based on the communities discovered using a dynamic community detection algorithm, we can intuitively extend the item-based collaborative filtering approach to dynamic community-based collaborative filtering approach. The preference prediction of the user u on the item i is typically computed based on the most similar items to the candidate item i. Intuitively, we only need to replace the set of the most similar items by the set of items which belong to the communities of candidate item (i.e. one community in the non-overlap case and several communities in the overlap case). Thus, we can formally define the preference prediction as follows:

$$P_{u,i} = \frac{\sum_{j \in C} s(i,j)\ r_{u,j}}{\sum_{j \in C} |s(i,j)|} \tag{3.1}$$

where C is the set of items pertaining to the community of i, $r_{u,j}$ is the rating given by the active user u to the item j and $s(i,j)$ is the similarity degree between items i and j.

In the case where the user is new (no ratings history), the target item of this user is learned by browsing item in the recommender system. We select then the candidate items that belong to the communities of the target item. The recommendation list contains the candidate items which are ranked according to their similarities relative to target item. We propose, in all this work, to compute the similarity between two items using the *Pearson correlation* similarity (see Equation 1.5) because it is proved that this one works better as a similarity measure for collaborative filtering (Sahebi & Cohen, 2011).

Example 3.3. *Let us consider the user-item ratings matrix of Table 1.1 composed of three users* $\{u_1, u_2, u_3\}$ *and five items* $\{i_1, i_2, i_3, i_4, i_5\}$ *and three communities of items* C_1, C_2, *and* C_3 *which are defined as follows:* $C_1 = \{i_1, i_3\}$, $C_2 = \{i_4, i_2\}$, $C_3 = \{i_5, i_4\}$. *Our aim is to propose a recommendation list that contains one item to user* u_2.
Firstly, we need to select the target item for user u_2. *We look for the item that* u_2 *has assigned the highest rating value. So,* i_4 *is selected as the target item for* u_2. *Based on the target user, we look for the candidate items.* i_4 *belongs to the communities* C_2 *and* C_3, *so the item belongs to* C_2 *or* C_3 *and is not yet rated by user* u_2 *is selected as candidate item. The list of candidate items is* $\{i_2, i_5\}$. *The next step consists in computing the predicted preference value of user* u_2 *on* i_2 *and* i_5 *as follows:*

$$P_{u_2,i_2} = \frac{s(i_2,i_4) r_{u_2,i_4}}{|s(i_2,i_4)|} = \frac{(-1)*5}{1} = -5,$$

$$P_{u_2,i_5} = \frac{s(i_5,i_4) r_{u_2,i_4}}{|s(i_5,i_4)|} = \frac{(1)*5}{1} = 5.$$

Note that the similarity value are computed in the previous Example 1.2.

Finally, the recommendation list for u_2 will contain the item i_5 having the highest predicted preference.

D2CF architecture deals with dynamic networks and takes advantage of the dynamic community detection process to enhance the recommendation task. Moreover, it provides a diverse recommended list because it looks at the generic pattern of users' interests taking into account their evolution over time.

3.3 Conclusion

In this chapter, we have presented the three main steps of our proposed architecture. The first step consists in building a dynamic network of items where the evolution of users' interests are modeled over time. In the next step, a dynamic community detection algorithm is applied on the dynamic network in order to identify communities of items which will be used as part as recommendation process in the final step.
Next Chapter provides an experimental study in order to evaluate our architecture comparing with two different recommendation approaches, using a real-world data set.

Chapter 4

Experimental study

4.1 Introduction

In this chapter we present the experimentation results relative to our Dynamic community based collaborative filtering approach comparing with two methods of collaborative filtering. The implementation of our architecture is based on Mahout Library tools, which provide an open source java package for recommendation task. This chapter is composed of two Sections. Section 4.2 details the experimental protocol used in the implementation process of our architecture, and Section 4.3 presents the experimental results evaluating the effectiveness of the recommendation based on our proposed architecture.

4.2 Experimental protocol

This section describes our experimental data, then it presents the evaluation metrics that will be used to evaluate this experiment. Finally it presents the procedure that we have followed to implement the D2CF architecture.

4.2.1 Dataset

To evaluate the effectiveness of D2CF approach, we propose to use one of the most popular recommendation datasets which are available through movieLens [1] website. MovieLens is

[1] http://movieLens.umn.edu

a web-based research recommender system that debuted in Fall 1997. Each week hundreds of users visit MovieLens to rate and receive recommendations for movies. The used dataset contains in total 100.000 ratings collected by 943 users on 1682 movies, from 19-09-1997 to 22-04-1998. The score of rating is ranged from 1 to 5. Each user has rated at least 20 movies. The ratings information are timestamped which is crucial in our study.

The MovieLens data are represented as a sequence of temporal events in the following way:

user U_1 rates movie I_1 with 5 at T_1,
user U_2 rates movie I_1 with 3 at T_1,
user U_2 rates movie I_5 with 5 at T_2, etc.

To experimentally determine the impact of the training set size on the performance of the whole recommender system in term of time and quality, we propose to test three scenarios:

- Set_1 : For each user, we randomly select 90% of his ratings as instances in the training set and the remaining ones (i.e. 10%) will be used in the testing set.
- Set_2 : For each user, we randomly select 40% of his ratings as instances in the training set and 10% will be kept as for the testing set.
- Set_3 : For each user, we randomly select 20% of his ratings as instances in the training set and 10% will be used in the testing set.

The data selection should take into account the temporal order of instances, so that, instances of the testing set should be chosen after those of the training set. The predicted preference computations are obtained from the training set and the evaluation of the efficiency of recommended items is performed by the testing set.

4.2.2 Evaluation metrics

Recommender systems research has used several types of measures for evaluating the quality of the provided recommendation. The Evaluation metrics can be categorized into into three classes (Herlocker, Konstan, Terveen, & Riedl, 2004): *predictive accuracy metrics*, *classification accuracy metrics*, and *rank accuracy metrics* and we present the most important representatives of each class.

1. ***Predictive accuracy metrics***: they quantify how much the recommender system's predicted ratings are close to the true user ratings. Most commonly used metrics are

Mean Absolute Error (Goldberg, Roeder, Gupta, & Perkins, 2001), *Normalized Mean Absolute Error* (Goldberg et al., 2001) and *Root mean squared error* (Bennett & Lanning, 2007).

2. ***Classification accuracy metrics***: in order to evaluate the quality of the recommendation list, these metrics measure how many times a recommender system makes correct or incorrect decisions about whether an item is good. Table 4.1 shows the possible categorization of items where N is the number of items in the database.

Table 4.1: Items' categorization

	Suggested	**Non-Suggested**	**Total**
Relevant	$N_{r,s}$	$N_{r,n}$	N_r
Irrelevant	$N_{i,s}$	$N_{i,n}$	N_i
Total	N_s	N_n	N

We can conclude that recommended items can be either successful recommendations (relevant) or unsuccessful recommendation (non relevant) and the relevant items can be either suggested in the recommendation list or not. Precision and recall (Basu, Hirsh, & Cohen, 1998) are the most popular metrics for evaluating information retrieval systems.

Precision: this measure is used to evaluate the validity of a given recommendation list and it is defined as the ratio of relevant items selected by the active user relative to the number of items recommended to him.

$$P = \frac{N_{r,s}}{N_s} \tag{4.1}$$

Recall: it computes the portion of favored items that were suggested for active user relative to the total number of the objects actually collected by him.

$$R = \frac{N_{r,s}}{N_r} \tag{4.2}$$

3. ***Rank accuracy metrics***: Rank accuracy metrics measure the ability of a recommendation algorithm to produce a recommended ordering of items that matches how the user would have ordered the same items. A rank accuracy or ranking prediction metric measures the ability of a recommender to estimate the correct order of items concerning the user's preference. The common used metric is **Kendall's** τ (Goldberg et al., 2001).

We will use *Precision* and *Recall* as our choice of evaluation metrics to report prediction experiments because they are most commonly used in information retrieval and easiest to interpret directly. In fact, if an algorithm has a measured precision of 70%, then the user can expect that, on average, 7 out of every 10 documents returned to the user will be relevant.

4.2.3 Implementation

Our goal is to recommend top k movies for target user. To this end, we need to implement the three main steps of D2CF (Pre-processing step, Dynamic community detection step and Recommendation step).

1. *Pre-processing step*: Our idea is to extract movies interactions in such way that we know what a movie has been assessed with another one with the same score from the part of the same producer taking into consideration timing. In fact, if an interaction between two movies occurred more than N times over a period of P days, an edge is established between them. We define the values of P and N such a way that we conserve more links between nodes. After performing several tests on the movieLens data, we set P and N respectively to 200 and 30 for Set_1, 200 and 20 for Set_2 and 200 and 5 for Set_3. This choice seems to be reasonable since the interactions between movies decrease with the number of training instances (ratings). An edge is established between i_1 and i_2 if:

   - i_1 and i_2 have interacted at least 30 times over a period of 200 days in Set_1.
   - i_1 and i_2 have interacted at least 20 times over a period of 200 days in Set_2.
   - i_1 and i_2 have interacted at least 5 times over a period of 200 days in Set_3.

   An edge is removed between i_1 and i_2 if:

   - There have occurred less than 30 interactions between i_1 and i_2 over a period of 200 days after the edge creation date in Set_1.
   - There have occurred less than 20 interactions between i_1 and i_2 over a period of 200 days after the edge creation date in Set_2.
   - There have occurred less than 5 interactions between i_1 and i_2 over a period of 200 days after the edge creation date in Set_3.

   Movies that are not very visible in the users' ratings data are considered as outliers. The outliers are the nodes that are disconnected of the core of the network due to

their low interactions with other movies (i.e. there are less than N users who give the same rating for both of them).

2. *Dynamic community detection step*: In this stage, we are able to apply any state of art dynamic community detection algorithm. In this experiment we choose to use iLCD algorithm to extract communities from the temporal network built above. Since the quality of resulting communities depends on the threshold value (i.e. with a low threshold, communities can be large and have little connection between them, however with a higher threshold will be smaller and more denser), we choose after several tests to set threshold to 0.5 to obtain overlapping, small and dense communities as shown in Figure 3. Besides, we set the minimum size of communities to 3.

3. *Recommendation step*: Using detected communities, we are now able to generate the top k recommendation list of movies to the active user. This step requires both the user's ID to look for his target item and the community structure as input parameters to select the candidate items that may interest the active user. Then, the predicted preference of each candidate item is computed using *Pearson correlation-based* similarity measure. The items which have the top k preference predictions are recommended to the active user.

In order to evaluate both of the effectiveness and efficiency of our proposed D2CF approach we compare the performance of this one with the following methods:

- The Static Community-based Collaborative Filtering (denoted S2CF for short), which is a static version of our proposed architecture. We kept the same parameters used for the dynamic network without taking into account the temporal dimension (i.e. an edge is established if a pair of nodes interact N times). This is possible since, as mentioned before, the *iLCD* algorithm allows both static and dynamic community detection.

- The traditional item-based Collaborative Filtering (detailed in Section 1.2.3) with *Pearson correlation-based* similarity measure.

4.2.4 Experimental platform

All our experiments were implemented using java language and compiled in Eclipse framework. We ran all our experiments on a windows7 based PC with intel Core i3 processor having a speed of 2.40 GHz and 4GB of Ram.

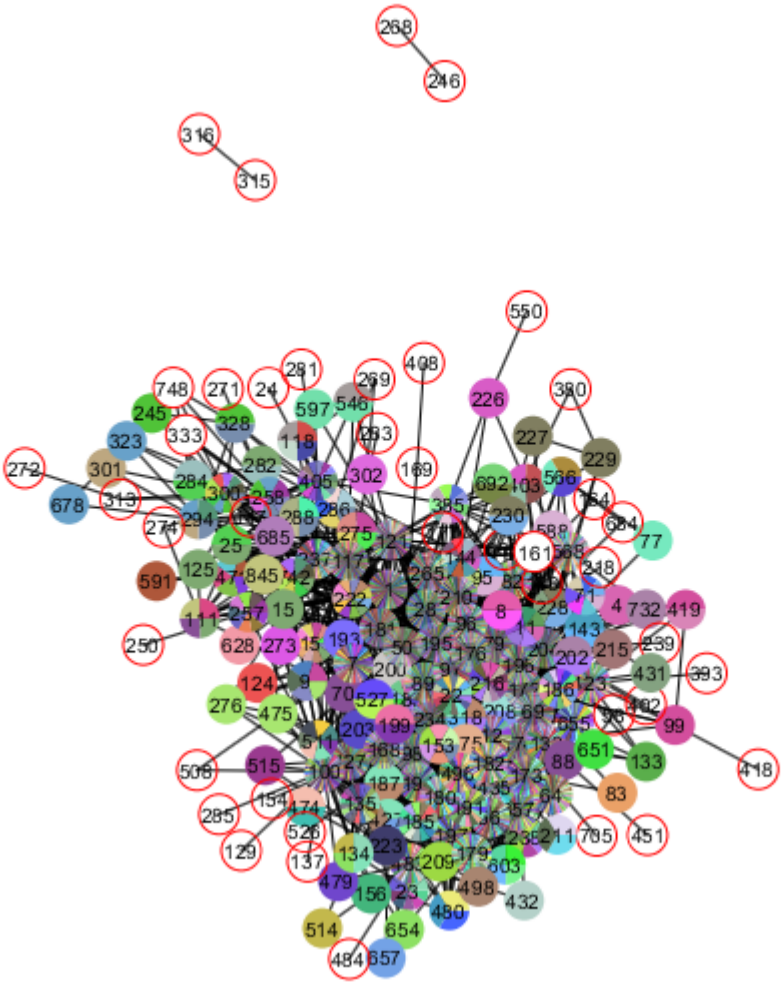

Figure 4.1: A capture screen showing the evolution of communities extracted via iLCD on the MovieLens data

4.3 Experimental results

We perform experiments on each case (*Set*_1, *Set*_2 and *Set*_3) by randomly choosing 10 different users, computing the precision, the recall and the response time of the generated recommendations using D2CF, S2CF and item-based CF, and then taking the average of the results. The obtained results are summarized in Tables 4.2, 4.3 and 4.4.

We can notice that our D2CF method outperforms traditional recommendation methods: item-based collaborative filtering and Collaborative filtering based on static community detection. In fact in *Set*_3, our approach is able even with sparse data to provide users with a rich and varied recommendation list based on the communities of movies.

The *recall* and *precision* values for both item-based and static community- based collaborative filtering decrease as we increase the training set size but our approach combining recommendation and dynamic community detection still provides better recommendation quality as shown in Figure 4.2.

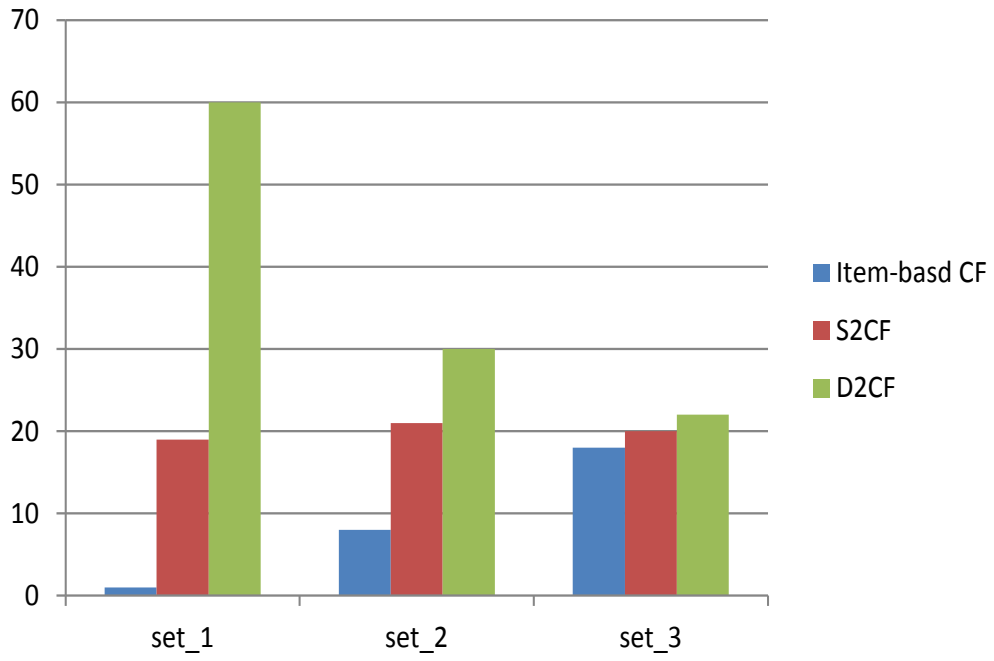

Figure 4.2: Impact of the dataset size on the recommendation quality

We can also see that D2CF gives its best results when the training set size is more important (Set_1) contrarily to item-based CF where the performance goes down as we add more data. In fact, in Set_1, D2CF proposes for one user an average of 6 good items out of every 10 recommended items while S2CD offers an average of two good items out of every 10 recommended items and finally item-based collaborative filtering gives an average of 0.15 good items at best as shown in Figure 4.3. However, in Set_3, D2CF proposes for one user an average of 2.2 good items out of every 10 recommended items. S2CF propose an average of 1.97 good recommendations per user and finally item-based offers and average of 1.8 good recommendation out of 10 recommended items.

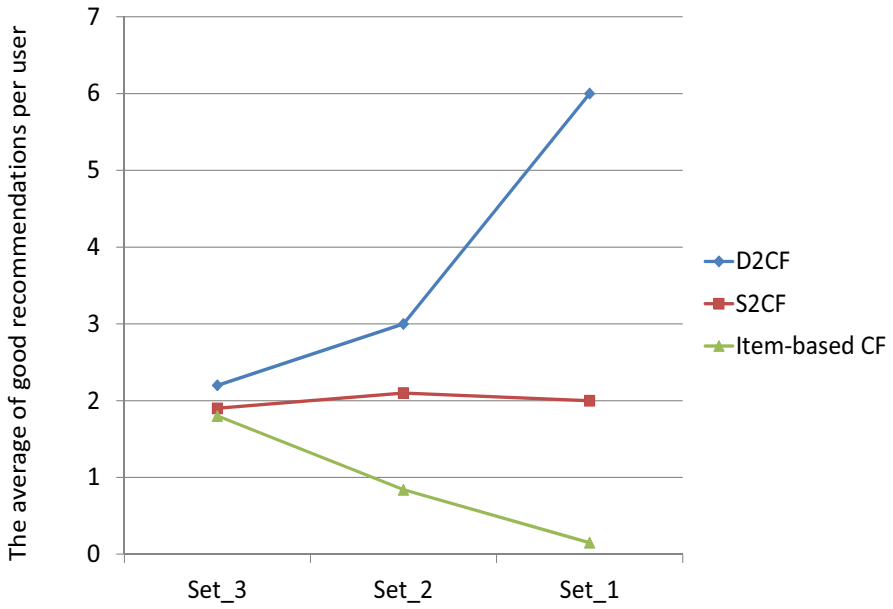

Figure 4.3: Performance of D2CF, S2CF and item-based: how many items (y-axis) have been selected on average as good recommendations out of a total of 10 recommended items for both D2CF, S2CF and item-based CF in Set_1, Set_2 and Set_3 (x-axis).

This observation is explained by the fact that the dynamic network learned by more users' data performs well the prediction of users' preferences for unseen items. However, for item-based CF we believe this happens as the item-based model suffers from data overfitting when data set is more large.

Note that the static version of our approach (i.e. S2CF) is more efficient than the traditional item-based collaborative filtering for both precision and recall criteria. In fact, the collaborative filtering based on the members of communities (i.e. S2CF and D2CF) works slightly better than collaborative filtering based on most similar items (i.e. item-based collaborative filtering) but it presents the worst value of response time in the case of large training set (Set_1) and this is can be justified by the fact that static communities are more large and dense.

D2CF provides the optimal value of response time in large set case (Set_1), this result

shows that the proposed D2CF is capable to scale to large data sets.

D2CF approach presents a significant improvement on recommendation on both small and large sets. We can say that this approach addresses both scalability and sparsity problems and it is able to handle the real-world networks by providing a dynamic recommendation based on dynamic communities.

Table 4.2: Precision and Recall values for Set_1

Approach	**Precision**	**Recall**	**Response time**
D2CF	**0.603**	**0.687**	**25.332 Sec**
S2CF	0.2	0.26	38.75Sec
Item-based CF	0.015	0.02	35.75Sec

Table 4.3: Precision and Recall values for Set_2

Approach	**Precision**	**Recall**	**response time**
D2CF	**0.3**	**0.49**	**19.023 Sec**
S2CF	0.21	0.195	23.413Sec
Item-based CF	0.084	0.091	27.75Sec

Table 4.4: Precision and Recall values for Set_3

Approach	**Precision**	**Recall**	**Response time**
D2CF	**0.223**	**0.34**	**9.81 Sec**
S2CF	0.197	0.221	10.75Sec
Item-based CF	0.18	0.2	12.75Sec

4.4 Conclusion

The experimental study provided in this chapter, shows that our proposed D2CF approach gives motivating results comparing with both item-based collaborative filtering and collaborative filtering based on static communities. Moreover, D2CF approach deals with real-world network where user's preferences and interests keep changing over time.

Conclusion

Since the amount of information in the world is increasing more quickly than our ability to process it, recommender systems have emerged as an efficient tool to provide us with the items that are most valuable to us. Recommender systems have been attacking the interest of researchers during the last decade.

With the aim to enhance the accuracy and the performance of the existing recommendations, many attempts have proposed to incorporate the recommendation with the community detection techniques called community-based recommendation. The community structure brought significant advances to our representing and understanding of real-world systems. In fact, the community structure may help the recommender system to personalize recommendation provided to users by focusing on its communities instead of the whole network. Besides, understanding the underlying structure of these communities may have a major impact on users' behaviors and pave the way to mapping emergent semantics in recommender systems.

Representing the dynamic aspect of users' interests is crucial in recommendation systems but it seems harder to ensure with the current community-based recommendation methods, because all of them rely either on static communities, or on communities extracted from time to time. These latters cannot represent the real aspect of dynamic network behaviors.

In this work, we have proposed a Dynamic Community-based Collaborative Filtering approach that combines recommendation and dynamic community detection to improve recommendation in dynamic networks. We have investigated using co-ratings relationship in order to model the dynamic aspect of users' interests over time into a dynamic graph of items. Then, a dynamic community detection algorithm is used to discover dynamic communities. We have defined communities as groups of items which are learned from the users' ratings behaviors. The items pertaining to the same community share certain

similar properties. The information given by these formed groups is exploited to restrict the recommendation prediction task. We have shown that using communities in the context of recommender systems helps in performing the traditional collaborative filtering. Our proposed architecture is able to deal with any dynamic community detection algorithm and to overcome the problem of dynamic users data which seems even harder to handle with the existing community-based recommendation techniques and which are crucial prerequisite for effective recommendation in real world network.

The experimental results show that our proposed D2CF outperforms both Item-based collaborative filtering and collaborative filtering based on static communities. As a future work, we will explore the similarity computation process of users pertaining to the same community in the recommendation context. Another future work would be to study the sensitivity of D2CF on the data overfitting.

References

Abrouk, L., Gross-Amblard, D., & Cullot, N. (2010). Community detection in the collaborative web. *International Journal of Managing Information Technology (IJMIT), 2010*, 1-9.

Basu, C., Hirsh, H., & Cohen, W. (1998). Recommendation as classification: using social and content-based information in recommendation. In *Proceedings of the 15th national conference on artificial intelligence (aaai'98)* (p. 714-720). Madison, Wis, USA.

Belkin, N., & Croft, W. (1992). Information filtering and information retrieval: two sides of the same coin? *Commun. ACM*, *35*(12), 29-38.

Bennett, J., & Lanning, S. (2007). The netflix prize. In *Proceedings of kdd cup and workshop* (p. 3-6).

Blei, D., Ng, A., & Jordan, M. (2003, March). Latent dirichlet allocation. *J. Mach. Learn. Res.*, *3*, 993-1022.

Blondel, V., Guillaume, J., Lambiotte, R., & Lefebvre, E. (2008). Fast unfolding of communities in large networks- louvain method. *Journal of Statical Mechanics: Theory and Experiment*, P1000.

Brandes, U., Delling, D., Gaertler, M., Gorke, R., Hoefer, M., Nikoloski, Z., & Wagner, D. (2008). On modularity clustering. *IEEE Transactions on Knowledge and Data Enginnering*, *20*(2), 172188.

Cazabet, R. (2013). *Dynamic community detection on temporal networks* (Unpublished doctoral dissertation). Université Toulouse 3 Paul Sabatier.

Cazabet, R., & Amblard, F. (2011, August). Simulate to detect: a multi-agent system for community detection. *The 2011 ACM International Conference on Web Intelligence and Intelligent Agent Technology(WI-IAT)*, *2*, 402-408.

Chakrabarti, D., Kumar, R., & Tomkins, A. (2006). Evolutionary clustering. In *Proceedings of the 12th acm sigkdd international conference on knowledge discovery and data*

mining (p. 554-560). ACM.

Chen, J., & Yan, B. (2010). Detecting functional modules in the yeast protein-protein interaction network. *Bioinformatics*, *22*(18), 2283-2290.

Deitrick, W., Valyou, B., Jones, W., Timian, J., & Hu, W. (2013). Enhancing sentiment analysis on twitter using community detection. *International Journal of Managing Information Technology (IJMIT)*, *5*, 192-197.

Fatemi, M., & Tokarchuk, L. (2013, Sept). A community based social recommender system for individuals groups. *Proceedings of the 2013 International Conference on Social Computing (SocialCom'13)*, 351-356.

Gfeller, D., Chappelier, J., & Rios, P. D. L. (2005). Finding instabilities in the community structure of complex networks. *Physical review*, *72*(5), 056135.

Goldberg, K., Roeder, T., Gupta, D., & Perkins, C. (2001). Eigentaste: a constant time collaborative filtering algorithm. *Information Retrieval*, *4*(2), 133-151.

Gregory, S. (2010). Finding overlapping communities in networks by label propagation. *New Journal of Physics*, *12*(10), 103018.

Hönsch, M. (2011). Detecting user communities based on latent and dynamic interest on a news portals. In *the 7th student research conference in informatics and information technologies iit.src* (Vol. 3, p. 47-50). ACM.

Herlocker, J., Konstan, J., Terveen, L., & Riedl, J. (2004). Evaluating collaborative filtering recommender systems. In *Acm transactions on information systems* (Vol. 22, p. 5-53).

Hopcroft, J., Khan, O., Kulis, B., & Selman, B. (2004). Tracking evolving communities in large linked networks. *Proceedings of the national academy of sciences of the United States of America*, *1*, 5249-5253.

Kamahara, J., Asakawa, T., Shimojo, S., & Miyahada, H. (2005, January). A commynity-based recommendation system to reveal unexpected interests. *Multimedia Modelling Conference*, 433-438.

Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer*, *42*(8), 30-37.

Langseth, H., & Nielsen, T. (2009). A latent model for collaborative filtering. *Technical Report 09-003, Department of Computer Science, Aalborg University, Denmark*.

Mucha, P., Richardson, T., Macon, K., Porter, M., & Onnela, J. (2010). Community structure in time-dependent, mutiscale, and multiplex networks. *Science*, *328*(5980), 876-878.

Newman, M., & Girvan, M. (2004). Finding and evaluating community structure in networks. *Phisical review E*, *69*(2), 026113.

Nguyen, N., Dinh, T., S.Tokala, & M.T.Thai. (2011, Sept). Overlapping communities in dynamic networks: Their detection and mobile applications. *Proceedings of the 17th annual international conference on Mobile computing and networking (MobiCom'11)*,

85-96.

Palla, G., Barabasi, A., & Vicsek, T. (2007, April). Quantifying social group evolution. *In Nature*, *446*, 664-667.

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

Porcel, C., Moreno, J., & Herrer-Viedma, E. (2009). A multi-disciplinar recommender system to advice research resources in university digital libraries. *Expert Systems with applications*, *36*(10), 12520-12528.

Qiang, H., & Yan, G. (2012, October). A method of personalized recommendation based on multi-label propagation for overlapping community detection. *Proceedings of the 3rd International Conference on System Science, Engineering Design and Manufacturing Informatization*, *1*, 360-364.

Qin, S., Menezes, R., & Silaghi, M. (2010). A recommender system for youtube based on its network of reviewers. *The IEEE International Conference on Social Computing*.

Raghavan, U., Albert, R., & S.Kumara. (2007). Near linear time algorithm to detect community structures in large-scale networks. *Physical Review E*, *76*(3), 036106.

Resnick, P., Iacovou, N., Suchak, M., Bergtrom, P., & Riedl, J. (1994). Grouplens: An open architecture for collaborative filtering of netnews. In *Proc. of the acm conference on computer supported cooperative work* (p. 175-186).

Sahebi, S., & Cohen, W. (2011). Community-based recommendations: a solution to the cold start problem. *In RSWEB'11*.

Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2000). Analysis of recommendation algorithms for e-commerce. In *Proceedings of the acm e-commerce. minneapolis, minn, usa.* (p. 158-167).

Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on world wide web, www'01* (p. 285-295).

Sarwar, B. M., Karypis, G., Konstan, J., & Riedl, J. (2002). *Recommender systems for large-scale e-commerce: Scalable neighborhood formation using clustering.*

Schafer, J. B., Konstan, J., & Riedl, J. (1999). Recommender systems in e-commerce. In *Proceedings of the 1st acm conference on electronic commerce* (p. 158-166). New York, NY, USA. Retrieved from `http://doi.acm.org/10.1145/336992.337035` doi: 10.1145/336992.337035

Soboroff, I., & Nicholas, C. (1999, August). Combining content and collaborative in text filtering. In *Proc. int'1 joint conf. artificial intelligence workshop: Machine learning for information filtering.*

Song, X., Lin, C., Tseng, B., & Sun, M. (2006). Modeling evolutionary behaviors for community-based dynamic recommendation. In *Proceedings of the 2006 siam inter-*

national conference on data mining.

Tang, L., Wang, X., & Liu, H. (2009). Uncovering groups via heterogeneous interaction analysis. In *Icdm.*

Ungar, L., & Foster, D. (1998). Clustering methods for collaborative filtering. In *Proceedings. workshop on recommendation systems. aaai press.*

Wen, Y., Liu, Y., Zhang, Z., Xiong, F., & Cao, W. (2014). Compare two community-based personalized information recommendation algorithms. *Physica A 398, 2014*, 199-209.

Xie, J., Chen, M., & Szymanski, B. (2013). Labelrankt: Incremental community detection in dynamic networks via label propagation. *arXiv preprint arXiv : 1305.2006v2*.

Yehuda, K. (2009). Collaborative filtering with temporal dynamics. In *Proceedings of the 15th acm sigkdd international conference on knowledge discovery and datamining* (p. 447-456). ACM.

Zhang, Y., & Koren, J. (2007). Efficient bayesian hierarchical user modeling for recommendation system. In *Proceedings of the 30th annual intern. acm sigir conf. on research and developement in information retrieval* (p. 47-54). ACM.

Zhao, G., Lee, M. L., Hsu, W., Chen, W., & Hu, H. (2013, October). Community-based user recommendation in uni-directional social networks. *Proceedings of the 22th ACM international conference on Conference on information knowledge management*, 189-198.

Abstract

With the increase of time-stamped data, the task of recommender systems becomes not only to fulfill users interests but also to model the dynamic behavior of their tastes. In this work, we propose a novel architecture, called Dynamic Community-based Collaborative filtering (D2CF), that combines both recommendation and dynamic community detection techniques in order to exploit the temporal aspect of the community structure in real-world networks and enhance the existing community-based recommendation. The efficiency of the proposed D2CF is studied via a comparative study with a recommendation system based on static community detection and item-based collaborative filtering. Experimental results show a considerable improvement of D2CF recommendation accuracy, whilst it addresses both of scalability and sparsity problems.

Keywords: Recommendation systems, Collaborative filtering, Dynamic Community Detection, Time varying graphs

Résumé

Avec l'augmentation des données horodatées, la tâche des systèmes de recommandation devient non seulement de satisfaire les intérêts des utilisateurs mais aussi de modéliser le comportement dynamique de leurs goûts. Dans ce travail de recherche, nous proposons une nouvelle architecture appelée "filtrage collaboratif basé sur des communautés dynamiques" (D2CF). Cette architecture combine la recommandation et les techniques de détection de communautés dynamiques en vue d'exploiter l'aspect temporel de la structure communautaire dans les réseaux du monde réel et d'améliorer les méthodes existantes de la recommandation basées sur les communautés. L'efficacité de l'architecture est évaluée par une étude comparative avec la recommandation basée sur les communautés statiques et un filtrage collaboratif basé sur les objets. Les résultats de l'expérience montrent une amélioration considérable au niveau de l'exactitude et la performance de la recommandation générée par D2CF. Outre, cette architecture est capable de traiter à la fois les problémes de l'évolutivité et la parcimonie des données.

Mots clés: Systèmes de recommandation, Filtrage collaboratif, Détection de communautés dynamiques, graphes qui varient dans le temps