Hindawi Publishing Corporation
Advances in Multimedia
Volume 2011, Article ID 852518, 19 pages
doi:10.1155/2011/852518

Research Article

From Community Detection to Mentor Selection in Rating-Free Collaborative Filtering

Armelle Brun, Sylvain Castagnos, and Anne Boyer

LORIA—Nancy-Université, 615 Rue du Jardin Botanique, 54506 Vandoeuvre-lès-Nancy, France

Correspondence should be addressed to Armelle Brun, armelle.brun@loria.fr

Received 5 October 2010; Revised 27 December 2010; Accepted 10 January 2011

Academic Editor: Andrea Prati

The number of items that users can now access when navigating on the Web is so huge that these might feel lost. Recommender systems are a way to cope with this profusion of data by suggesting items that fit the users needs. One of the most popular techniques for recommender systems is the collaborative filtering approach that relies on the preferences of items expressed by users, usually under the form of ratings. In the absence of ratings, classical collaborative filtering techniques cannot be applied. Fortunately, the behavior of users, such as their consultations, can be collected. In this paper, we present a new approach to perform collaborative filtering when no rating is available but when user consultations are known. We propose to take inspiration from local community detection algorithms to form communities of users and deduce the set of mentors of a given user. We adapt one state-of-the-art algorithm so as to fit the characteristics of collaborative filtering. Experiments conducted show that the precision achieved is higher then the baseline that does not perform any mentor selection. In addition, our model almost offsets the absence of ratings by exploiting a reduced set of mentors.

1. Introduction

The democratization of Internet and network technologies has resulted in a large increase of information, readily accessible to everybody. This growth had been an advantage during its first years as the access to information became generalized. However, the volume of information is now so enormous that users cannot easily get the information they search for and are drowned in the mass of resources. This overabundance has very often the effect of leading to unsatisfied users.

As a consequence, a critical issue of the current Web applications is the incorporation of mechanisms for delivering information that fits users' needs, whilst increasing their satisfaction.

Recommender systems (RSs) provide users with personalized recommendations of resources or items, based on the knowledge they have about users. A recent observation showed that users are now aware of their need to be assisted [1] and are prepared to adopt recommender systems [2]. The increasing popularity of these systems in information seeking or commercial e-services has meant that the need for quality, accuracy, and reliability of recommendations has become tremendous. Recommender systems generally fall into three categories: content-based systems which compute recommendations from the semantic content of items [3], knowledge-based systems where recommendations rely on knowledge about the domain, the users and preestablished heuristics [4], and at last collaborative filtering systems which compute recommendations by examining either users' preferences on items, or their past interactions with the system, called traces of *usage* [5].

The general principle behind collaborative filtering (CF) can be summarized into one sentence: an active user will probably be keen on the items that his like-minded users (also called neighbors or mentors) have previously liked, and that he or she has not yet consulted. Users usually express if they have liked an item or not under the form of ratings [6]. Based on this piece of information, CF identifies the users which have similar ratings with the active user: his mentors. The system then exploits the collective knowledge of mentors' preferences, and estimates the rating that this active user would assign to each item he or she has not rated yet. This way, the system is able to determine which item(s)

to recommend to a user: those with the highest estimated ratings.

CF has been adopted these last few years mainly due to the fact that no explicit information about users is required a priori: users do not have to fill any questionnaire or forms to get some recommendations. Moreover, no explicit information about items is required either: the tedious and time-consuming indexing task is avoided [7]. In addition, the accuracy of the CF-based recommendations provided to users is high in comparison with the content-based approach [8], as soon as the system gets a sufficient (and high) number of ratings. CF also has the advantage to recommend items not directly linked to those the active user has already liked. This feature is called novelty or serendipity [9].

One drawback of CF is the requirement of user-provided ratings. Relying on ratings may be penalizing since users often provide few ratings, not to say no ratings at all. Indeed, assigning a rating to an item is a complex and time consuming task, and users are not aware of the benefit they can get when rating items. Moreover, user-provided ratings may not be reliable [10]: on a predefined and usually small scale, which rating should be assigned to an item? For example, let suppose that a user has liked a given movie, on a scale of 1 (he did not like it) to 5 (he did really like it). Which rating should he/she assign to this movie? a 4 or a 5? In addition, in many application domains, the system cannot ask users to rate items. For example, when users navigate on a Web site or consult news, it is inappropriate to ask them voting, since it interrupts their information seeking process.

Standard CF relies on the computation of similarities of ratings (or preferences) between users to perform mentor selection. When no rating is available, no similarity of preferences can be precisely estimated and classical CF techniques cannot be used. Then, the question is: how to select mentors when no information about user preferences (ratings) and about users' similarities are known?

To answer this question, we propose to represent the information about users collected by the system in an unweighted graph. The nodes correspond to users. In the graph, we simply consider that two users are connected if they have common consultation behaviors. This connection does not reflect a similarity in preferences, but the fact that they have some commonly consulted items. Based on this representation, we investigate ways to identify mentors by using the only information available: the structure of the graph (as no weight is associated with the links).

Community detection algorithms are efficient in detecting communities or classes of nodes in graphs, focusing on the topology of the graph [11]. We thus take inspiration from community detection algorithms, specifically one local community detection algorithm from the state of the art [12], to solve the problem of mentor selection in collaborative filtering. To the best of our knowledge, the structure of the graph, community detection and local community detection algorithms have never been exploited in the frame of CF, in particular to perform mentor selection.

Section 2 focuses on CF and the way mentors are usually selected. In Section 3, community detection and local community detection algorithms are presented. Section 4 introduces our model that exploits local community detection algorithms to detect mentors in the frame of CF. Sections 5 and 6 are, respectively, dedicated to the evaluation of this new approach and to the discussion of the results. Last, we conclude this work.

2. Collaborative Filtering

2.1. Collaborative Filtering in General. Given a set of items I, and a set of users U, the input data of a CF system is the set of ratings $r(u,i)$ that users $u \in U$ have assigned to items $i \in I$, most of the time on a non binary rating scale (e.g., from 1 to 5). This set is stored under the form of a rating matrix $= U \times I$. Let a be the active user. To make recommendations to a, the CF system first estimates the ratings $r(a,i)$ that a would assign to all the items i he has not rated yet. Second, given this set of ratings, the system recommends the items with the highest estimated rating values. In the literature, "making recommendations" is equivalent to estimating the user's ratings for unrated items.

To estimate $r(a,i)$, the system adopts either an item-based or a user-based approach. The item-based approach supposes that a user will more likely appreciate items that are similar or related to the items he has already liked. At the opposite, the user-based approach exploits similarities between users and recommends a user the items that his like-minded users, or mentors, have appreciated. In this paper, we are interested in the user-based approach.

There are basically two approaches to implement a user-based collaborative filtering algorithm, respectively, the memory-based and the model-based approaches. The memory-based approach, also known as "lazy learning", simply exploits the user-item rating matrix without any transformation when the system is asked for a recommendation. We can say that the learning phase is skipped. On the contrary, the model-based approach first builds a model of the user-item rating matrix and then uses this model to make recommendations.

2.2. Memory-Based Collaborative Filtering. One classical way to estimate a rating in a memory-based approach is presented in (1)

$$r(a,i) = \overline{r(a)} + \sum_{u \in U_i} \mathrm{sim}(a,u) * \left(r(u,i) - \overline{r(u)}\right), \quad (1)$$

where U_i is the set of users that rated item i, $r(u,i)$ is the rating that user u has assigned to item i, $\overline{r(u)}$ is the average rating of user u and $\mathrm{sim}(a,u)$ is the similarity between users a and u. The estimated rating is the weighted average of the absolute difference between the ratings the other users assigned to the item i and their average rating, added to the average rating of user a. The more a user is similar to a, the higher his weight in the estimation of $r(a,i)$ is.

The similarity between two users is a similarity of ratings, which measures if both users rated (i.e., appreciated) items in

the same way. This similarity is classically computed with the Pearson correlation coefficient, presented in (2)

$$\mathrm{corr}(a,u) = \frac{\sum_{j\in I_{a\cap u}}\left(r(a,j)-\overline{r(a)}\right)*\left(r(u,j)-\overline{r(u)}\right)}{\sqrt{\sum_{j\in I_{a\cap u}}\left(r(a,j)-\overline{r(a)}\right)^2}*\sqrt{\sum_{j\in I_{a\cap u}}\left(r(u,j)-\overline{r(u)}\right)^2}}, \quad (2)$$

where $I_{a\cap u}$ is the set of commonly rated items by users a and u and $\overline{r(a)}$ is the average rating of user a. This correlation measure has been proved to be the most adequate measure in user-based approaches [13].

The cosine measure can also be used to compute the similarity between two users. The corresponding equation (3) presents the way the cosine measure is computed

$$\mathrm{sim}(a,u) = \frac{\sum_{j\in I_{a\cap u}} r(a,j)*r(u,j)}{\sqrt{\sum_{j\in I_a} r(a,j)^2}*\sqrt{\sum_{j\in I_u} r(u,j)^2}}, \quad (3)$$

where I_a is the set of items rated by a. The cosine measure has proved to be more efficient in the item-based approach, to compute the similarity between items [14].

The similarity values in (1) are instantiated by either the correlation or the similarity value in (2) and (3) above. Reference [15] presents an overview of equations that compute user similarities or estimate ratings.

Although accurate, memory-based approaches suffer from combinatory complexity problem, since it requires to compute the similarity between each pair of items or users.

2.3. Model-Based Collaborative Filtering. Model-based collaborative filtering techniques constitute a good alternative to reduce the time complexity of memory-based collaborative filtering [13]. These algorithms consist in creating descriptive models correlating users, items and associated ratings via a learning process. These models can take several forms, such as Bayesian networks, classes of items and/or users, and so forth. The basic idea is that a partial knowledge of data may be sufficient to provide accurate recommendations. As an example, in (1), only a subset of users is actually useful to estimate $r(a,i)$. This subset of users is referred to as mentors or the community of user a. In CF, a mentor is a like-minded user who is used as a representative user for computing estimated ratings. The choice of the mentors (and their number) highly influences three main features in a CF system: the computation time, the accuracy of the estimated ratings and the coverage. More concretely, we have the following.

(i) Mentor selection has the advantage to speed up the rating estimation process as only a subset of users is used to estimate ratings [16].

(ii) Mentor selection increases the accuracy of recommendations as lowly related users may lower the accuracy of the estimation [17]. Additionally, the accuracy of the rating estimation highly depends on the adequacy of the mentors.

(iii) Mentor selection influences the coverage [18]. The coverage reflects the ability of the recommender system to estimate a rating value for a given user a and a given item i. When no rating can be estimated, no recommendation can be performed. The lower the number of mentors, the lower the coverage, since the probability that a mentor has already rated the item is low. At the opposite, the larger the number of mentors, the higher the coverage. However, this increases the computation time.

Given these features, we can conclude that the choice of mentors is a tricky and important task that may highly influence the recommendations.

Two main approaches of mentor selection are often used in CF: direct neighbor selection and clustering.

2.3.1. Direct Neighbor Selection. Given a set of users that have a non-null similarity value with a, direct neighbor selection consists in keeping the users that comply with a given criterion.

This criterion can be a threshold value [19, 20]. A similarity threshold is fixed a priori and all users that have a similarity with the active user a above this threshold are selected as mentors of a. The resulting community is user-centered.

The criterion can also be an integer value K. The K nearest neighbors (KNN) of a are retained, K being fixed a priori [20]. The K neighbors with the highest similarity values are considered as the mentors of the active user (if less than K neighbors exist, all the neighbors are kept).

One main drawback of direct neighbor selection is scalability. Mentors are detected at runtime since the profile of all users has to be kept in memory; similarities between users are recomputed each time mentor selection and recommendations have to be performed. In addition, the choice of the K value or of the threshold value is tricky. At last, KNN is not robust to data sparsity: it is unable to form reliable neighborhoods in case of high sparsity level [21]. However, as similarities are up-to-date when recommendations are made and as the communities are user-centered, this direct neighbor selection approach is accurate.

2.3.2. Clustering. The second approach classically used to select mentors performs clustering. These clusters are computed offline and constitute the model that is periodically reused to generate recommendations. Depending on if an item-based or a user-based approach is chosen, the system builds either classes of items [14], or classes of users [16]. Item-based CF is known to be very accurate and for highly improving the scalability, as long as the relationships between items remain relatively static. Item-based approaches are widely deployed in industry and commonly studied in research settings. In this paper however and as mentioned previously, we focus on the user-based approach, as it is known to be more adapted for web applications where items are greatly volatile [16]: the set of items often change, since some items are deleted, new items constantly appear, and some others are modified.

Concretely, user-based clustering identifies groups of users with similar preferences, that is, who appear to have similar ratings [13]. Once the clusters are created, recommendations for the active user a can be made by exploiting the preferences (here ratings) of all users belonging to the cluster of a.

There are typically three kinds of clustering methods: partitioning [22], hierarchical [16], and fuzzy techniques [23]. Most of the time, CF relies on a partitioning techniques. In this case, users are grouped into separated clusters: each user belongs to exactly one cluster.

Many partitioning algorithms have been studied in the frame of CF, such as [24, 25]. Some of these algorithms require either an a priori number of clusters, or a fixed-size for clusters, others a similarity threshold, and so forth. These algorithms are parametric. For example, the K-means algorithm [22], computes K clusters so that the average pairwise similarity between users within clusters is maximized. Most of non-parametric algorithms aim at maximizing a given criterion (that usually represents the quality of the classification), without any parameter chosen by hand [26].

Partitioning algorithms provide a good recommendation accuracy among clustering techniques. When recommending items, exploiting clusters of users is robust to scalability problem as it reduces the memory requirements, and classes are computed offline. However, partitioning algorithms often suffer from convergence problems, as they are highly sensitive to initial starting conditions [27]. Moreover, the computation time require to execute such algorithms is high.

Hierarchical clustering deals with these issues by lowering the complexity, speeding up the convergence time, and allowing the distribution of computations. Castagnos and Boyer proposed in particular a recursive model that simplifies the selection of optimal initial points, by splitting the population into two subsets at each step [16]. Nevertheless, the quality is dependent of the expected number of clusters. If the number of clusters is low, the system will provide inaccurate recommendations. If this number is too large, there is a risk of overgeneralization.

An interesting way to avoid bias due to bad clusterization consists in relaxing the constraint of belonging to one and only one class. This is the principle of fuzzy collaborative filtering, in which allocation of data points to clusters is not hard [28].

Let us observe that whatever the approach adopted, a similarity value is exploited; for example it may be the one presented in (2) or (3).

In some approaches, the similarity value is used in combination with another criterion. For example, [29] has proposed a clustering algorithm based on the information about the shared nearest neighbors, in addition to their similarity. In this approach, two elements are grouped if they share many of their nearest neighbors, and if they are themselves nearest neighbors of the other element. This algorithm has been reused in many domains such as information retrieval [30], databases [31] and recently in data traffic [32].

2.4. Discussion. In this section we have presented the main approaches used in CF to select mentors. The direct neighbor selection method leads to accurate recommendations. However this approach is neither scalable nor robust to data sparsity. Clustering of users is one way to solve these two problems [33]. First, it has the advantage to only compute similarities between users and to select the mentors periodically, and not each time recommendations are needed. Second, two users may belong to the same class even if they have a null similarity value (due to a null number of corated items). Third, the size of the clusters does not have to be defined a priori, contrary to the KNN approach and the resulting clusters may have very different sizes.

These clustering algorithms have however several drawbacks.

(i) In most of the clustering methods, users belong to only one cluster. However, in some cases, users may need to belong to several groups of users. Some clustering techniques thus allow users to belong to several clusters [34]. The estimated rating is then an average across the clusters, weighted by degree of participation of the user in each cluster.

(ii) Some users may be border nodes (at the limit of the class they belong to). Thus, their mentors (those in the same class) may be really different of their actual nearest neighbors, which may decrease the recommendation performance. We can notice that this drawback does not occur when using the direct neighbor selection, as the community formed is user-centered.

(iii) Despite its ability to cope with the scalability problem, clustering is known to generate less-personal thus less accurate recommendations [35, 36].

(iv) Few works focused on user-centered clustering. Reference [17] proposed a decentralized algorithm to build user-centered clusters. However, the complexity remains high if computations are not distributed on a high number of processors.

Whatever the mentor selection method is, a similarity measure between users is required. As a consequence, they can thus not be used in the case no similarity value is available.

Based on the above statements, we can deduce that developing user-centered communities or clusters, where each user may belong to several communities (clusters), would lead to a high accuracy of recommendations and a solution to the sparsity and scalability problems.

2.5. Managing Unary Ratings. Most of the works on CF exploit user preferences under the form of ratings. As previously mentioned, in many application cases, ratings cannot be collected. Only the information about the user consultations may be known. These consultations can be viewed as unary ratings (has consulted/has not consulted).

When unary ratings are available, three main approaches may be used.

The First Approach. Aims at estimating the ratings that users would have assigned to the items if they had rated them. The user preferences are estimated from implicit feedback. In [37], Oard and Kim question whether it might be possible to substitute implicit feedback for explicit ratings. They propose three types of implicit feedback (Examination, Retention, and Reference) and a set of observable behaviors. Castagnos et al. have extended this work by defining a generic implicit modeling function to transform implicit feedback into estimated ratings [38, 39]. This approach relies on usage mining techniques, and infer preferences from usages. The generic function first collects every possible implicit feedback. It then groups user consultations per item and per user, and synthetizes data under the form of criteria. These criteria may be the duration or/and the frequency of consultation of the items. Finally, these criteria are used to provide an estimation of the ratings. This approach alleviates the sparsity problem and offers the advantage of providing estimated ratings that can be reused by collaborative filtering techniques.

However, rating estimation is also very intrusive into privacy of users and is quite imprecise due to the inference process under uncertainty. As an example, the active user can be on phone, which explains a long consultation duration but a low interest for an item. He can also save a document to read it later without having any opinion on it, or delete it by accident. Thus, implicit modeling functions generally lead to poor performance in term of accuracy.

The Second Approach. Skips the rating estimation step and directly applies similarity measures on unary ratings. Karypis [40], Linden et al. [41], and Miranda and Jorge [42] adopt an item-based point of view. They compute the similarities between pairs of items, using adapted versions of the cosine-based or conditional probability-based measures. As mentioned previously, item-based models are known to be accurate; for these reasons, the unary model presented in [40] will be evaluated in our experiment framework.

Recently, Redpath et al. have extended these works to user-based CF, where similarities between users are also computed with either the adapted cosine similarity measure or the Jaccard coefficient [43]. This approach assumes that the larger the number of items two users coconsult, the more similar they are. However, as no information about their preferences is known, this assumption may not always be true. This approach will also be evaluated in our experiment framework.

The Third Approach. Proposes to use one additional information about the user consultations, which is the order of consultation of the items. Sequences of consultations of items are mined [44, 45]. However, this approach may be quite unprecise, since it is extremely hard and time-consuming to identify typical robust usage patterns under uncertainty.

In the following section, we focus on community detection methods from the literature, to determine the adequacy between our unary rating framework and the literature.

3. Community and Local Community Detection

3.1. Community Detection. As presented in the previous section, a clustering process requires the information about the links existing between elements to be clustered, and the weights associated to these links. These elements and their links can also be represented under the form of weighted graphs, where the nodes are the elements to be clustered and the edges are their links (e.g., the link value may be the similarity between the elements) [46]. In our case, nodes of the graph represent users that have to be clustered.

Graph clustering has recently received much attention [47–49], especially due to the numerous domains where data can be represented under the form of graphs. The best known graph clustering algorithms attempt to optimize specific criteria related to the graph, such as k-median, minimum sum, minimum diameter, and so forth, [50]. Other algorithms are application-specific and may take advantage of known characteristics of the application data or of the application domain. In general, the approaches proposed to perform clustering in graphs cannot easily scale up to large problems due to their high time complexity [11].

The concept of community detection is linked to the concept of clustering objects in graphs. The notion of community can be seen in a broad sense: depending on the context, it can be synonymous of module, class, group, cluster, and so forth. The aim of community detection in graphs is to identify groups of objects, by only analyzing the topology. As a result a community in a graph is a set of nodes between which the interactions are (relatively) frequent. We assume that the nodes in a community probably share common properties or play similar roles within the graph. For example, communities of users in the blogspace often correspond to users sharing topics of interests. The community detection task may be defined as follows: "community detection involves the analysis of the network structure with the goal of identifying communities, that is, groups of objects (which are represented as nodes in the network) that are more densely connected (on the network) to each other than with the rest of the objects" [51].

Community detection methods have been applied in a wide range of scientific problems, for example, social networks to identify groups of friends [52, 53], citation networks to study the centrality and the significance of scientific disciplines and their interrelations [54, 55], the World Wide Web to identify and manage web page topics [56], biology and epidemiology to detect the diffusion of viruses [57, 58].

In the frame of social networks, Tang [59] asked for the actual difference between graph clustering and community detection, and makes a clear comparative presentation of what distinguishes the two processes. Clustering works on a distance or similarity matrix whereas community detection works on discrete data and manages an adjacency matrix. The community detection algorithms thus use the graph properties and exploit notions such as: k-clique, quasi-clique [60], node-betweenness, edge-betweeness [52, 61], and so forth. As a consequence, the majority of the community detection algorithms proposed in the literature have been

designed for undirected and unweighted graphs. However, some of them have been proposed for directed [56], weighted [62] or signed graphs [63].

The algorithms used to detect communities can be classified in similar categories with clustering: fuzzy-community-detection [64], hierarchical community detection [65] and partitioning [66]. For example, a hierarchical community detection algorithm either groups highly connected nodes into larger and larger communities, or divides the graph progressively into smaller and smaller disconnected subgraphs, identified as the communities. One example of hierarchical community detection algorithm is the one proposed by Girvan and Newman, called the GN algorithm [52]. This algorithm uses the edge betweenness measure, that expresses the importance of the edges when transmitting across the graph following paths of minimal length. An edge has a high betweenness value if (almost) all shortest paths connecting nodes of two communities run through it. The GN algorithm splits the graph into disconnected subgraphs, by removing the edges with the highest betweenness score. Subgraphs then undergo the same procedure, until the whole graph is divided into a set of isolated nodes.

Partitioning algorithms frequently use another kind of information: the number of links within a set of nodes, and the number of links between nodes of this set and the rest of the graph. A community is thus a set of nodes highly connected to each other, but not much connected to nodes outside the community. Thus the problem consists in dividing the nodes in groups, such that the number of edges between the groups is minimal. The number of groups has to be defined beforehand. A description of partitioning algorithms can be found in [67].

Let G be a graph, with an adjacency matrix $A_{j,l}$ making links within the graph explicit. Let j be a node of the graph and $k_j = \sum_{l \in G} A_{j,l}$ the degree of j. Let us consider D a subgraph of G, to which node j belongs. k_j, the degree of j, can be split in two elements, with regard to D:

$$k_j(D) = k_j^{\text{in}}(D) + k_j^{\text{out}}(D), \tag{4}$$

where $k_j^{\text{in}}(D) = \sum_{l \in D} A_{j,l}$ is the number of edges connecting node j to other nodes inside D and $k_j^{\text{out}}(D) = \sum_{l \notin D} A_{jl}$ is the number of connections toward nodes in the rest of the graph.

Radicchi et al. [68] proposed two definitions of communities: strong communities and weak communities. A strong community is defined as follows:

$$k_j^{\text{in}}(D) > k_j^{\text{out}}(D), \quad \forall j \in D. \tag{5}$$

D is a strong community if each node has more connections within the community than with the rest of the graph.

A weak community is defined as follows:

$$\sum_{j \in D} k_j^{\text{in}}(D) > \sum_{j \in D} k_j^{\text{out}}(D). \tag{6}$$

D is a weak community if the sum of all degrees within D is larger than the sum of all degrees toward the rest of the graph.

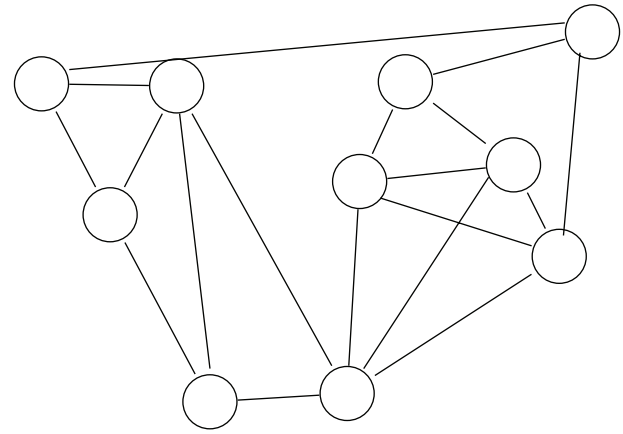

Figure 1: An example of simple graph.

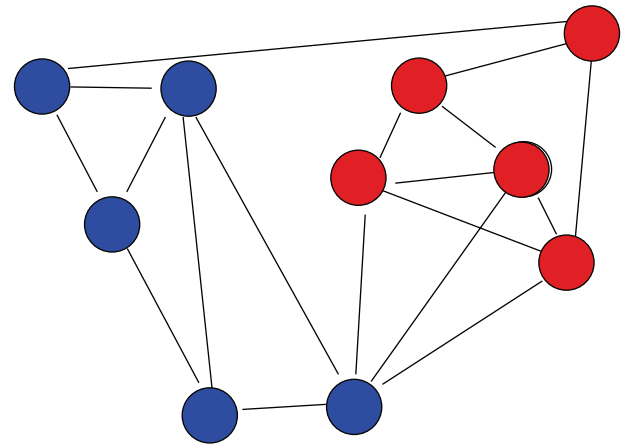

Figure 2: An example of two communities.

We can notice here that the definition of a strong community is more constraining than a weak community, as the connectivity measure applies to each node. Moreover, each strong community is also a weak community. Figure 1 presents an example of a graph. The latter is split into two communities (resp., in red and in blue) in Figure 2.

3.2. Local Community Detection. In the context of large-scale graphs, two problems may arise: first, the complete graph may not be known; second the graph may be too huge to be stored. For example, the Web is a so large and evolving structure that no graph can be easily constructed. In addition, as the Web is stored in a decentralized way, building such a graph is difficult. On the same principle, in social networks such as the Facebook application (http://www.facebook.com/), the number of users is so huge that the network cannot easily be stored.

In recommender system applications, specifically those based on collaborative filtering, graphs of items (item-based approach) or graphs of users (user-based approach) are managed; the corresponding graphs are huge and their storage is a complex task. Specifically, the stage of finding the set of nearest neighbors of a given user (his community)

in the graphs of users is not tractable in the frame of large graphs.

In such cases, community detection algorithms cannot be used, thus it is not feasible to determine the globally optimal community structure. Instead, the search may be limited to determining the local community structure in the neighborhood of a query node. The complexity of the search is thus reduced. Local-community detection algorithms are a way to detect communities when the graph is really huge, that is, not tractable.

At the opposite of community detection algorithms that have a global view of the whole graph, local community detection algorithms have the characteristic to rely on a local point of view and allow to detect communities when focusing on a specific node rather than on the whole graph.

Local community detection algorithms start from a query node (also called seed node) and iteratively add nodes to the community being discovered, based on the local view of the graph. At each step of the iteration process, the nodes actually added to the local community are the nodes that are highly connected to all the nodes of this community, that is, nodes that act as a bridge with another community (with a high betweenness value), and are connected to the whole community.

Local community detection algorithms have the advantage of decreasing the complexity of the process, compared to standard community detection algorithms, as only a subset of nodes are managed at a time. Local community detection algorithms have been mostly used in the frame of social networks [69–71].

In comparison with traditional graph clustering algorithms and community detection algorithms, that perform partitioning of edges, local community detection algorithms result in overlapping communities.

Some algorithms have been proposed to detect local communities. For example, Tian et al. [72] proposes an algorithm designed for both unweighted and weighted graphs.

The local community detection algorithms have thus the particularity of being iterative and of having a local point of view. We can remark that in graph clustering, some iterative or/and local algorithms have also been proposed. For example, Song and Kasabov [73] has presented a fast iterative one-pass algorithm for dynamic clustering. In [74], a local clustering algorithm has been proposed based on a stochastic approach.

Most of the local community detection algorithms manage three sets of nodes [71] as follows:

(i) D the community under construction, which is typically initialized with the query node,

(ii) N the Neighboring nodes not in D but sharing an edge with at least one element of D,

(iii) U the Unexplored nodes, that is, those not adjacent to D.

In some algorithms, the set D is divided into the core set C and the boundary set B. The core set C has no edge with a node in N, while the boundary set B has at least one edge with one node in N.

```
(1) D ← {queryNode}
(2) N ← neighbors(queryNode)
(3) repeat
(4)     select the "best" node n ∈ N
(5)     D ← D ∪ {n}
(6)     N ← (N − n) ∪ neighbors(n) − D
(7) until termination Criterion
(8) return filter(D)
```

Algorithm 1: Scheme of traditional local-community detection algorithms.

Any implementation of a local community detection algorithm requiresthe following:

(i) the instantiation of the selection of the next node to add to the community,

(ii) the termination criterion (when to stop adding nodes),

(iii) the filtering (which nodes, if any, have to be removed from the community).

Most of the local community detection algorithms follow the scheme presented in Algorithm 1.

Recently, Chen et al. [12] proposed an improvement of classical local-community detection algorithms to cope with the problem of outliers that tend to be included in the communities. The information about the number of connections considered to form the communities is replaced by the average number of nodes. The $L(D)$ metric represents the "quality" of a community D. It is computed with the following two terms. The first term $L_{\text{in}}(D)$ measures the average internal degree of nodes in D and is computed as follows:

$$L_{\text{in}}(D) = \frac{\sum_{j \in D} k_j^{\text{in}}(D)}{|D|}, \tag{7}$$

where $k_j^{\text{in}}(D)$ represents the number of edges between the node j and the nodes in D, and $|D|$ is the number of nodes in D.

The second term $L_{\text{ex}}(B)$ measures the average external degree of nodes in B

$$L_{\text{ex}}(D) = \frac{\sum_{j \in B} k_j^{\text{out}}(B)}{|B|}, \tag{8}$$

where $k_j^{\text{out}}(B)$ is the number of connections between node j and external nodes. As C is a subset of D that has no link with nodes out of D, $L_{\text{ex}}(D)$ is strictly equivalent to $L_{\text{ex}}(B)$.

The $L(D)$ metric is defined as

$$L(D) = \frac{L_{\text{in}}(D)}{L_{\text{ex}}(D)}. \tag{9}$$

The higher the $L(D)$ metric, the better the community D. In this algorithm, the resulting communities are weak

communities, according to the definitions of Radicchi et al. [68].

This algorithm has the same scheme than the one described in Algorithm 1.

(i) The algorithm starts with $B = D$, made up of a single node, a query node q and $C = \emptyset$.

(ii) At a given iteration step, the selection of the next node to be inserted in the community is made as following: the node that maximizes L is added to D.

(iii) The termination criterion is the evolution of the value of L, the algorithm stops when the value of L decreases.

The advantage of this algorithm is that no lower bound about the connectivity has to be fixed a priori. The algorithm automatically determines the bound, based on the value of L.

4. Community Detection Algorithms for Collaborative Filtering

Noting that community detection algorithms seem appropriate for our research framework, we chose to model Collaborative Filtering under the form of a graph: the nodes (the vertices) are the users, and the edges represent the existence of a link between two users. Using graph-theoretic approaches in collaborative filtering has been initially proposed by Aggarwal et al. in [75]. They rely on the same graph structure and the twin notions of horting and predictability to address the scalability problem. Exploring the graph allows to quickly identify neighbors and users with valuable experience. Additionally, Ekstrand et al. [76] use a graph structure with collaborative filtering to recommend research papers. In our case, this graph structure will be of high value to detect communities in a unary rating context.

As explained in introduction, the challenge we address is that of not being able to collect ratings, rendering the traditional CF approach powerless. Thus, the only information available in those cases is whether a given user has consulted an item (e.g., a Web page) or not. Although this data is less informative than ratings, a huge quantity of such data is available.

One major problem appearing in this case is that selecting nearest neighbors or clustering users is not an easy task, since no similarity value can be computed precisely enough. As seen before, one way to compute similarities between users relies on the use of the cosine similarity measure, adapted to unary values or the Jaccard coefficient [43].

However, computing similarities between all pairs of users is time consuming. It takes $O(M)$ time, between two users, where M is the huge number of items. (http://www.eecs.berkeley.edu/~pliang/cs294-spring08/lectures/collaborative/slides.pdf) In addition, such metrics assume that users who coconsult a high number of items, tend to be similar users. However, these users may have opposite preferences and may actually not be similar.

We thus decide to not compute such a similarity value, by simplifying its computation stage. We consider that if two users have coconsulted more than n (to be fixed) items, these users are similar, this similarity measure being fixed to a constant value 1. The resulting graph has the distinctive feature of having unweighted edges. Let us notice that the computation of this similarity value is not complex. First, it is a single count. Second, in the counting process, as soon as the count reaches n, the process stops and an edge is created. Besides, as the input data constantly evolves, the update of the edges is facilitated as the existing edges do not have to be reexamined.

As a result, the challenge we face is the selection of mentors of a given user, when no information about the similarity between users is available.

The first statement that can be made is that too many users are connected to the active user a. Moreover, as the edges are unweighted, the system cannot select nearest neighbors. However, one additional information may be used: the structure of the graph.

We thus propose to exploit a local community detection algorithm to detect mentors of the active user. Such an algorithm mainly exploits the structure (the topology) of the graph by finding sets of users (communities) highly connected with users within the community and not much connected with users out of the community. As previously noticed, the nodes within a community probably share common properties and/or play similar roles within the graph. As a consequence, in the frame of collaborative filtering, we can assume that users in a community have similar consultation behaviors and maybe similar preferences.

More specifically, we aim at exploiting a local community detection algorithm. A local community detection is preferred over a community detection algorithm for several reasons.

(i) The local community detection algorithm copes with the scalability problem of community detection algorithms.

(ii) Once a query node a is chosen, the users of the resulting community can be considered as the mentors of that starting node a.

(iii) The concept of mentor is local and asymmetric: a mentor is specific to a given user. For example a user b may be a mentor of a, but a may not be a mentor of b. A local community detection algorithm deals with such constraints, which is not the case of community detection algorithms.

(iv) When executing the algorithm for each possible a, the resulting communities overlap. Some nodes can thus belong to several (even many) communities. Each node can thus be a mentor of several users.

(v) The size of each community is not fixed a priori and the number of communities is equal to the number of users.

We propose to use an adapted and improved version of the local community detection algorithm proposed by Chen et al. in [12], as it has been designed to efficiently manages outliers, resulting in high-quality classes. The algorithm in [12] will be called LCD in the rest of the paper, and will serve as our baseline.

4.1. Selecting Only Direct Neighbors. Let us remind that our goal is to detect the appropriate set of mentors for each active user a. The LCD algorithm has the characteristic of building communities with elements that may not be directly connected to the query user a. The resulting communities may not be centered on a. As a result, a may be at a limit border of his community and many users in his community may not be connected to a, resulting in a decrease of the recommendation accuracy.

We propose to apply some modifications on this algorithm, so as to be properly used in the frame of collaborative filtering.

In the recommendation stage of standard CF, the estimated rating (or score) of an item i is computed as the weighted average of the neighbors' ratings on this item i. Weights are instantiated by the similarity values between the active user and his mentors. Therefore if the active user a's community is made up of users not connected to a (with a null similarity value), these users will not be useful in the recommendation stage. The consequence of this observation is that communities have to be strictly made up of users connected to a, meaning that the LCD algorithm in [12] is thus not an optimal local community detection algorithm.

During the iteration step of the LCD algorithm, all the neighbors of the current community D (the candidate nodes) are tested (the metric L is evaluated). The node maximizing L is then included in D. We propose to relax this constraint by reducing the set of candidate nodes to the set of users connected to a. As a result, the mentors that belong to the community are not only all connected to a, but are also highly connected with the users of the community and not so much connected with other nodes.

The resulting community will be directly comparable to the set of mentors used by a standard KNN approach.

This adapted version of LCD will be called "User-Centered LCD" or UC-LCD.

4.2. Filtering Out Subconnected Nodes. Despite the fact that LCD has been designed to cope with the unexpected inclusion of outliers in communities, some outliers may all the same be included in certain configurations of the graph. As an example, when focusing on the first iterations of the LCD and UC-LCD algorithms, we can notice that the node that maximizes L is the node that minimizes L_{ex}. In the worst case, this node may be connected to only one node out of the community. However, once this subconnected node is included in the community, it remains in the community till the end of the process. Nevertheless, as this node is subconnected, it will be subconnected to the nodes within the community. Thus, the quality of the community is lowered. In addition, such a node can be also integrated in further iterations, resulting in the integration of additional nodes potentially subconnected to the community.

To improve the quality of the resulting community, we propose to filter out the subconnected nodes from the community. At each iteration step, the connectivity of each node in the community D is computed and each node with a connection value below a predefined threshold θ is filtered out from the community. The filtering procedure is presented in Algorithm 2. The resulting communities will thus be only made up of nodes highly connected with the other nodes of the community. The new local community detection algorithm is presented in Algorithm 3.

Require: D: a community
(1) $N \leftarrow \emptyset$
(2) **for all** $(j \in D)$ **do**
(3) **if** $(k_j^{\text{in}}(D) < \theta \cdot |D|)$ **then**
(4) $N \leftarrow N \cup \{j\}$
(5) **end if**
(6) **end for**
(7) **for all** $j \in N$ **do**
(8) $V \leftarrow V \cup \{j\}$
(9) **end for**
(10) **return** (D)

Algorithm 2: Procedure FilterSubConnected(θ).

(1) $D \leftarrow \{queryNode\}$
(2) $N \leftarrow neighbors(queryNode)$
(3) $\theta \leftarrow Initialize\ Connectivity\ Thresold\ Value$
(4) **repeat**
(5) select the "best" node $n \in N$
(6) $D \leftarrow D \cup \{n\}$
(7) $N \leftarrow (N - n) \cup neighbors(n) - D$
(8) FilterSubConnected(θ)
(9) **until** termination Criterion
(10) **return**(D)

Algorithm 3: Scheme of the F-LCD Algorithm.

At the opposite of the filtering step presented in Algorithm 1, the filtering in this algorithm is made within the iterations. In addition, the nodes that can be filtered are all the nodes of the community, and not only the last node inserted in the community. Indeed, a given node may be connected to most of the nodes in the community at one iteration step, and this connectivity may decrease as the community increases. Of course, the connectivity threshold θ has to be fixed a priori.

This refined version of UC-LCD, that filters out subconnected nodes will be called F-LCD.

4.3. Recommending. Once the communities have been computed, the recommendation process can be performed. In classical approaches, the set of mentors is used, likewise their similarities with the active user. Here, no information about the similarity between the active user and his mentors is known. Thus, the classical rating estimation step of (1) cannot be used.

We propose to adapt this rating estimation step so as to be used when no similarity value is available. Assuming that the highest the number of a's mentors having consulted an item, the highest the probability a will like this item. We propose a recommendation equation that computes the score of an

item by applying democratic voting rules among the mentors of a. The mostly consulted items among the mentors will be those recommended. The resulting equation is presented in (10)

$$\text{score}(a, i) = \sum_{u \in D_{a,i}} 1. \tag{10}$$

$D_{a,i}$ is the set of users in the community of a that have consulted the item i. We can notice that, as in traditional selection of neighbors with clustering algorithms, the exact number of mentors that will be used is not known a priori. It is a subset of the community, but their number will depend on the item of interest in the estimation of the score.

5. Experiments

5.1. Experimental Data. The experiments conducted in this paper aim at evaluating the adequacy of community detection algorithms to identify mentors in CF. These experiments also aim at quantifying the recommended items' loss of accuracy when no rating is available, that is, when classical approaches cannot be used.

We choose to conduct our experiments on two corpora from the state of the art, that contain ratings. In these corpora, we substitute consultations for ratings. The experimental datasets we have chosen are the well-known MovieLens and Jester datasets.

The MovieLens dataset (http://movielens.org/) contains 100,000 explicit ratings, on 1682 movies (items) from 943 anonymous users. Each rating varies in the range from 1 (dislike) to 5 (like). Each user has rated at least 20 movies. The data sparsity on this dataset is 93.7%.

The Jester dataset (http://goldberg.berkeley.edu/jester-data/) is made up of 1.8 Million explicit ratings, on 100 jokes (items) from 24,983 anonymous users. Each rating varies in the continuous range from −10 to +10. Each user has rated at least 36 jokes. The data sparsity on this dataset is only 27.5%.

We used these two datasets in order to validate the algorithms we proposed, in different conditions. MovieLens allowed us to evaluate the robustness of the algorithms in a context of high sparsity, with a high number of items, and in a situation where ratings are mainly highly positive (the average rating value is 3.53 on a scale ranging from 1 and 5). However, this dataset suffers from a drawback: there is no direct mapping between the presence of a rating and an observable user action. On one hand, users have likely viewed other movies that they did not rate. On the other hand, ratings are frequently given directly on search result pages and not necessarily on item detail pages. In addition, users may rate a movie they have not actually watched, or that they watched a long time ago. Thus, we chose to validate the algorithms on an additional dataset. The Netflix dataset suffers from the same deficiency than MovieLens. In Jester on the contrary, users necessarily have to read a joke before being able to rate it, and they are able to read any of them at any moment. Moreover, at the opposite of MovieLens, the Jester dataset does not face any sparsity problem, the distribution of ratings is more homogeneous than MovieLens and the rating scale is quite larger. At last, there are many more users than items in Jester, which is complementary to MovieLens, where the number of items is almost twice the number of users.

Following commonly used test procedures (see Section "Cross-Validation Subset Generation Scripts", http://www.grouplens.org/system/files/README_10M100K.html), we divided these corpora into five parts, in order to perform a five-fold cross-validation; training is made up of 4 parts and test of the last part, repeated five-times by alternating the test part. On the MovieLens dataset, this division was provided along with the dataset; each part contains 20% of the ratings. On the Jester dataset, we performed this division; we randomly divided the dataset into 5 parts. Similarly to the MovieLens dataset, each part contains 20% of the ratings.

For both corpora, the ratings are transformed under the form of consultations; the value 1 is assigned if a user has rated an item (whatever is the rating value), and 0 else. As a result, 1 means that the user has consulted the item, and 0 means he/she has not.

To build the graph of users, and determine if an edge exists between two users, we choose to fix a minimum number of coconsulted items. On the MovieLens dataset, this value has been set to 20, as suggested by [77]. On the Jester dataset, it has been set to 36, as it is the minimum number of jokes each user has rated. Thus, if two users have coconsulted more than 20 movies or 36 jokes, a link between them is created, and the value of this link is equal to 1; else no link between these two users is created.

5.2. Evaluation. In classical collaborative filtering, the system relies on the set of ratings the users have assigned to items (in the training dataset). These ratings are exploited to estimate unknown preferences on the items the users have not rated yet. These missing ratings are then compared to the ratings from the test dataset to evaluate the accuracy of these estimations. This accuracy is often computed under the form of the Mean Average Error (MAE), that computes the mean difference between the estimated ratings and the actual ratings [14].

Herlocker et al. [9] demonstrate that this metric is in fact not very accurate when considering users' ratings, as a user is usually interested to know if he is going to like the item or not. As a consequence, several papers propose to use other classical information retrieval metrics, such as precision and recall [9, 78].

In our experiments, as no rating is available during training and test, no rating can be estimated. Thus, the MAE measure cannot be used in this context.

We propose to evaluate the performance of the various approaches presented in this paper in terms of precision. The precision reflects the ability of a system to recommend items that users will actually like.

During the recommendation process, the system computes a score for each item the active user has not consulted yet. This set of items is then sorted according to this score, and the top of the list is the set of recommended items. Obviously, the number of recommended items has to be fixed.

Concretely, the precision is defined as the percentage of items, among those recommended by the system, that the user has actually liked; it is defined as follows:

$$\text{Precision} = \frac{\text{Number of liked recommended items}}{\text{Number of recommended items}}. \tag{11}$$

The highest the precision value, the most accurate the system.

However, this evaluation requires the information about which items have been actually liked by users. In the test datasets we exploit, the users' ratings are actually known, thus the items liked by the users are also known. On the MovieLens dataset, we follow the work presented in [79]: a rating greater or equal to 4 reflects that a user has liked an item. On the Jester dataset, we consider that a rating greater or equal to 6 means that the user liked the item. Of course, this information about the interest of users on items will not be used to compute recommendations, it will be used only to evaluate the accuracy of the approaches.

In the experiments below, we fixed the number of items recommended by the system to 10 (size of the top-n list), which is a usual size for recommendation lists [80, 81].

5.3. Description of the Datasets. We present here some statistics about the datasets used to train and evaluate the approaches.

As told previously, on the MovieLens dataset, the number of users is 943. Each user in this dataset has rated at least 20 movies. On the 5 training datasets, on average 739 users have coconsulted/corated more than 20 items with at least one other user. The average number of connected users among these users is 165 (22.3%); the maximum number of neighbors is 612 (82.8%) and the minimum is 1.

On the Jester dataset, the number of users is 24,983. Each user in the dataset has rated at least 36 jokes. On the 5 training datasets, 20,145 users have coconsulted/covoted at least 36 jokes with at least one other user.

The average number of connections among these users is on average 15,644 (77.6%), the maximum number is 18,697 (92.8%) and the minimum is 1.

5.4. Baseline Models with Rating Values. In this section, we are interested in the accuracy of recommendations when the ratings are available. We focus on the user-based collaborative filtering. Two classical collaborative filtering approaches (see Section 2.3) will give us the reference precision values. We will then test our approaches that do not use ratings, with the aim of getting as close as possible to these reference precisions.

5.4.1. Classical Collaborative Filtering with KNN. We first evaluate the precision when mentor selection is performed by selecting direct neighbors (KNN) [20]. We relied on the Pearson Correlation Coefficient (2) to estimate the similarity value between users, since the literature shows it works well [82]. The number of neighbors K has been set to 50, for both corpora; this value represents the best compromise between precision and neighborhood size according to the experiments we conducted and is in accordance with what has been suggested in [83]. The resulting precision is presented in Figure 3. We can first notice that the precision on the Jester dataset is roughly 20% lower than the one on the MovieLens dataset.

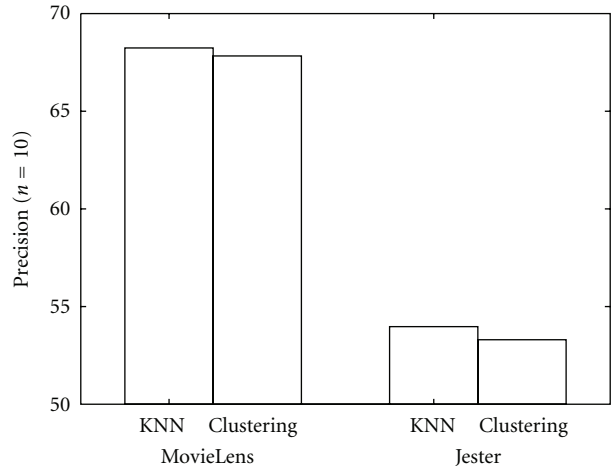

Figure 3: Precision values on MovieLens and Jester when using KNN and clustering approaches for neighbors selection, when ratings are available.

5.4.2. Classical Collaborative Filtering with Clustering. We also evaluate the precision when mentors are selected with a user-based clustering technique [22]. A K-means algorithm has been used and the number of classes has been set to 20. The resulting precisions are also presented in Figure 3.

When comparing the precisions of the two approaches, we can notice that the KNN-based collaborative filtering performs slightly better than the clustering-based collaborative filtering; an improvement of around 1% is achieved on both corpora, which is not statistically significant. This conclusion is in accordance with the state of the art.

The following sections are dedicated to the evaluation of the precision when only the information about consultations is available.

5.5. Classical Approaches When Ratings Are Not Available. In this section, we present two classical approaches when no rating is available.

5.5.1. Exploiting the Whole Set of Connected Users. When users' ratings are not available, no quantitative link can be easily computed. Thus, two users are either connected or not; this link is unweighted. As explained in Section 5.1, we decide to connect two users if they have coconsulted more than a fixed number of items. As a result, as all connected users have the same similarity value, neighbor selection cannot be easily made.

One way to build the set of mentors of the active user is to keep all users that are connected to him. In the recommendation stage, a democratic vote among the mentors is performed to estimate the score of a given item: the more the users connected to the active user have

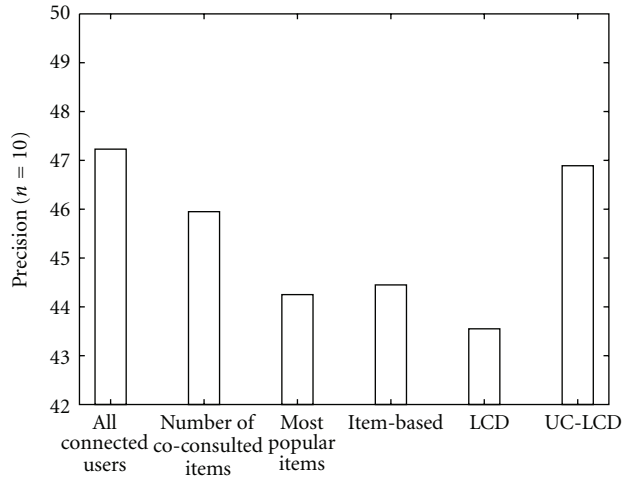

Figure 4: Precision values on MovieLens dataset for classical approaches for neighbors selection when ratings are not available.

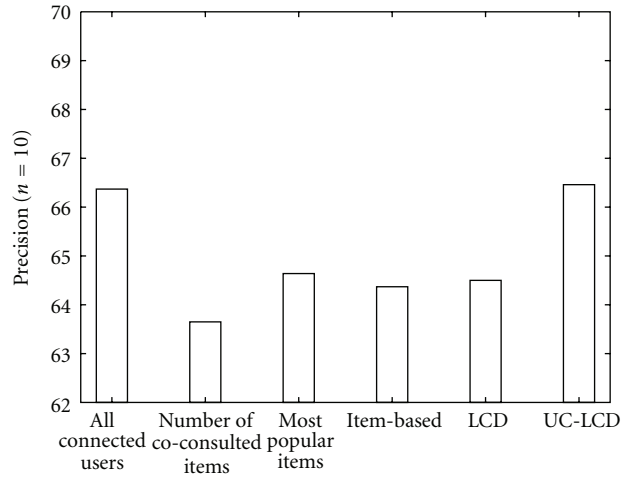

Figure 5: Precision values on Jester dataset for classical approaches for neighbors selection when ratings are not available.

consulted an item, the more the active user should like this item. The minimum number of coconsulted items has been set to 20 for the MovieLens dataset and 36 for the Jester dataset. The resulting precisions for both datasets are presented in the first bar "All Conn. Users" of Figures 4 and 5. In addition, the average size, the median and the quartiles of the resulting communities are presented in the first line of Tables 1 and 2.

On the MovieLens dataset, exploiting users' consultations, using the whole set of connected users as mentors, and a democratic vote among these mentors does not lead to a significant decrease of the precision: only 1.5% compared to the exploitation of users' ratings and KNN to select mentors. At the opposite, on the Jester dataset, a large decrease is achieved: 14%.

We performed an additional experiment to study if the large decrease on the Jester dataset is due to the exploitation of the consultations (loss of the rating information), or to the way the mentors are chosen. In this experiment, the set of mentors has been selected by exploiting the ratings, it is the same set than the one used in Section 5.5.1. A democratic vote has then been used to compute the recommendations. The resulting precision is 47.32, which corresponds to a decrease of 12% compared to the precision obtained with the use of ratings for training and for test. We can conclude that, on the Jester dataset, the rating values have a large influence on the precision of the recommendations. This may be explained by the rating scale used in Jester, which is not only continuous, but also larger than the one used on the MovieLens dataset.

Table 1: Settingup of the communities of the MovieLens dataset.

Way of selecting mentors	Avg number of mentors	Lower quartile	Median	Upper quartile
Whole set of connected neigh.	165	7	89	313
Algorithm LCD	240	12	387	434
Algorithm UC-LCD	99	1	30	190
Algorithm F-LCD (with $\theta = 0.5$)	48	1	2	80

Table 2: Settingup of the communities of the Jester dataset.

Way of selecting mentors	Avg number of mentors	Lower quartile	median	Upper quartile
Whole set of connected neigh.	15644	17178	17646	18062
Algorithm LCD	18376	17645	19564	20098
Algorithm UC-LCD	5037	7	7122	7663
Algorithm F-LCD (with $\theta = 1.0$)	172	5	18	155

5.5.2. Taking into Account the Number of Coconsulted Items. When no rating is available, the similarity between users can even so be computed. The number of items coconsulted by two users can be used as a basis to compute this similarity: one can assume that users with a high number of coconsulted items tend to be similar. The similarity measure we use is the one presented in [42].

The mentors of the active user are selected according to their similarity measure with the active user. The number of users kept is has been chosen experimentally as the number leadings to the highest precision value; it has been set to 50 on the MovieLens dataset and to 450 on the Jester dataset. The resulting precisions are presented in the second bar

"# Co-cons. Items" of Figures 4 and 5. On both corpora, when the set of mentors is selected based on their number of coconsulted items with the active user, the precision decreases compared to the use of the whole set of connected users. This decrease is particularly large on the Jester dataset: 6% compared to the use of the whole set of connected users. This decrease is only 2% on the MovieLens dataset. Thus, the choice of the mentors based on the rate of coconsulted items does not seem to be a good choice on a user-based approach. We can conclude that users who tend to consult the same items do not necessarly tend to share the same opinions.

Let us notice that, in both previous experiments, a democratic vote among mentors has been used in the recommendation stage. Thus the movies recommended (with the highest scores) may tend to be popular items. To verify that this democratic vote does not come down to recommending the most popular items, all users taken together, we propose to evaluate the precision of recommendations when the most popular items are recommended. The corresponding precisions are presented in the third bar "Most Pop. Items" of Figures 4 and 5. For both corpora, the resulting precisions are lower than those obtained when exploiting the whole set of connected users. As a consequence, we can say that the democratic vote does not come down to recommend the most popular items.

5.6. The Item-Based Approach. As shown in Section 5.5, exploiting the number of coconsulted items to compute the similarity between users lowers the precision in user-based approaches. However, this information has often been used in item-based approaches [40, 41]. We thus used this item-based metric and evaluated the corresponding precisions and, as suggested by [40], the vectors have been normalized. The fourth bar "Item-based" of Figures 4 and 5 represent the corresponding precisions. On the MovieLens dataset, the resulting precision is slightly lower than the one of the user-based approach; this decrease is about 2.5%. At the opposite, on the Jester dataset, the precision has been increased by 1.6%. These differences may be explained by the characteristics of the two datasets. The MovieLens dataset has much more items than users, the user-based approach leads to more accurate recommendations. At the opposite, the item-based approach performs better on the Jester dataset as the number of items is smaller than the number of users.

The highest precision values reached the previous experiments are those related to use of the whole set of connected users as mentors. These precision values will now be viewed as the baseline values when ratings are not available.

5.7. Community Detection Algorithms. We now focus on the use of community detection algorithms to detect mentors.

5.7.1. The Original Algorithm (LCD). Although the original algorithm proposed in [12] has the drawback of discovering communities that include users not directly connected to the active user a, we are all the same interested in studying the precision associated with the resulting communities. As in the previous experiments, a democratic vote among the users of the community is computed to estimate the score of each item (10) and perform recommendations. Let us remark that the users in the resulting communities who are not connected to the active user cannot be used in the recommendation stage. Despite the possible large number of users in the communities, the number of users actually useful for the recommendation may thus be reduced. In addition, as the LCD algorithm forms non user-centered communities, the active user may be on the border of his/her own community, which may result in a low precision.

The characteristics of the resulting communities are presented in the second line of Tables 1 and 2. As expected, the size of the communities is larger than the baseline (when considering all users directly connected to the active user). The average size is actually increased by 45% for the MovieLens dataset and 17% for the Jester dataset. The smaller increase on the Jester dataset is due to the high percentage of users connected to each active user.

The corresponding precision values are presented in the fifth bar "LCD" of Figures 4 and 5. The precision values are low: 63.55 on the MovieLens dataset and 44.9 on the Jester, which corresponds to a decrease of, respectively, 5.5% and 3.2%, compared to the use of the whole set of connected users. This decrease may be due to the reasons presented hereabove. The lower variation of the size of the communities in Jester compared to those in MovieLens may explain the smaller decrease of the precision.

To study the influence of the mentors not directly connected to the active user a, we conducted an additional experiment. The community detection algorithm was initialized with the whole set of direct neighbors (i.e., D was initially made up of the active user a and his connected users). The algorithm was then executed to discover additional neighbors in D. The resulting precision was below the baseline precision. We can conclude that indirect neighbors in this case are not useful mentors.

5.7.2. The User-Centered Algorithm UC-LCD. In this paper, we propose to adapt the LCD algorithm so as to detect user-centered communities, only made up of users connected the active user a. The recommendation stage remains the same than the one used in the previous experiments (10). Details about the resulting communities are presented in the third line of Tables 1 and 2. These communities are a subset of the communities used in Section 5.5.1. The corresponding precisions are presented in the sixth bar "UC-LCD" of Figures 4 and 5.

On both corpora, the precision reached by the UC-LCD algorithm exceeds the precision of the LCD algorithm, while reaching a similar precision to that of the baseline one (when using the whole set of neighbors). It is even slightly higher on the Jester dataset. In addition, this similar precision is achieved with a smaller set of mentors. On the MovieLens dataset, the communities are on average 40% smaller than the baseline ones and 68% smaller on the Jester dataset.

5.7.3. Filtering Subconnected Users in the Communities (Algorithm F-LCD). In Section 4.2, we put forward that the local community detection algorithm proposed by [12] has the drawback of inserting in the communities some users

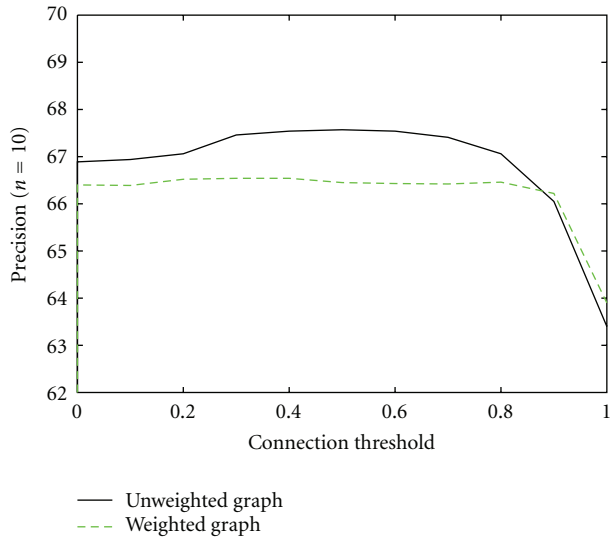

Figure 6: Precision values on MovieLens according to the connectivity threshold θ.

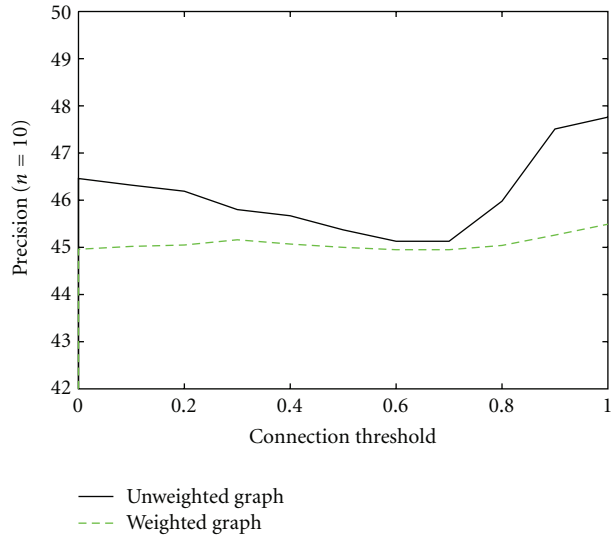

Figure 7: Precision values on Jester according to the connectivity threshold θ.

with few connections in the graph. These users are mainly inserted in the communities in the first iterations. As a result, these users lower the quality of the communities. Thus, we proposed to add one constraint to the composition of the communities, related to the connectivity of each element within the community. At a given step s, one user is removed from the community if his/her connectivity is below a threshold value θ fixed a priori. This threshold is called the connectivity threshold.

We performed several experiments, with the connectivity threshold θ ranging from 0.0 (no removal, i.e., UC-LCD) to 1.0 (a node is removed if it is not connected to all the nodes of the community). The resulting precisions, according to this threshold, are presented in Figures 6 and 7. Figures 8 and 9 present the evolution of the size of the communities, depending on this threshold.

In addition to these experiments, we evaluated the precision values with weighted edges. The weights associated with the links are those used in Section 5.5.2. The computation of the L_{in} and L_{ex} measures are thus adapted to take these weights into account. The number of links is no more used. It is replaced by the sum of the values of the edges. The resulting precisions are also presented in Figures 6 and 7. The evolution of the size of the corresponding communities, according to the threshold, are presented in Figures 8 and 9.

Figures 6 and 7 show that, on both corpora, the use of weighted links does not lead to a significant increase of the precision, compared to not filtering the communities (UC-LCD). At the opposite, when using F-LCD with unweighted links, an improvement of the precision is obtained. On the MovieLens dataset, the precision increases along with the threshold value, till a value $\theta = 0.5$. Above this value, the precision decreases. The maximal value is 67.57 which corresponds to an increase of 1% compare to the UC-LCD algorithm. This small increase is reached with a number of mentors 52% lower than UC-LCD. Compared

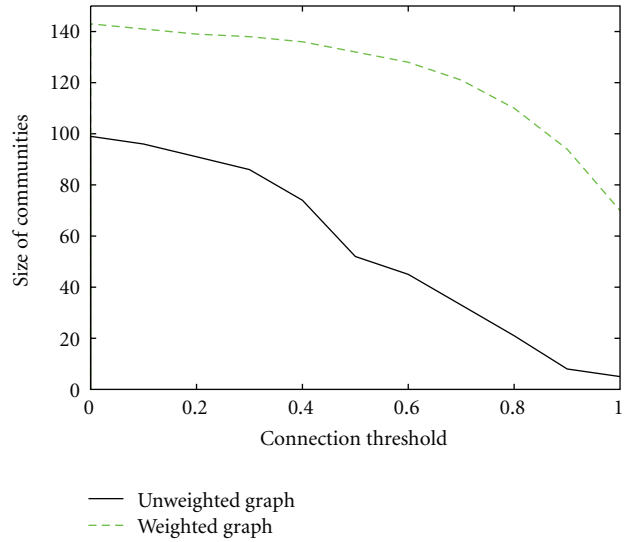

Figure 8: Size of communities on the MovieLens dataset, according to the connectivity threshold θ.

to the mentors from the baseline (All connected users), this decreases reaches 71%.

On the Jester dataset, the precision decreases with the threshold value until $\theta = 0.7$ and then increases. The optimal threshold value is $\theta = 1.0$. The corresponding precision value is 47.76, which corresponds to a increase of 3% in comparison with the UC-LCD algorithm, while reducing the number of mentors by 96%. This decrease is 99% when comparing to the set of mentors from the baseline.

The precision of F-LCD remains naturally lower than the one achieved when managing ratings. However, F-LCD reaches a precision value very close to that objective. The differences between F-LCD and the rating-based algorithms

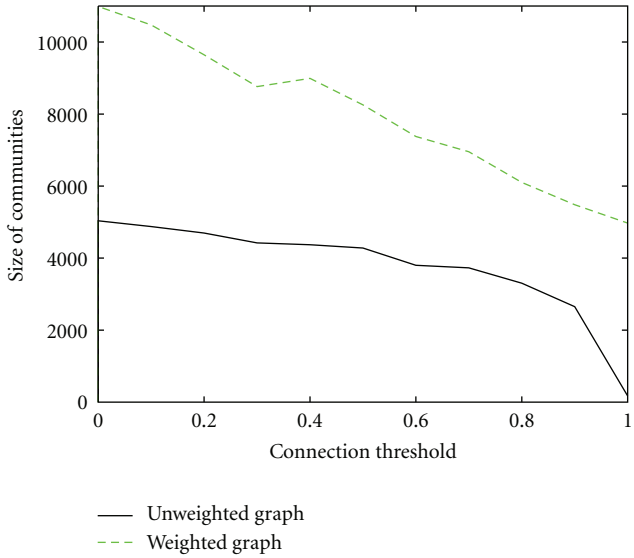

Figure 9: Size of communities on the Jester dataset, according to the connectivity threshold θ.

are even not statistically significant on the MovieLens dataset.

Figures 8 and 9 show that the size of the communities decreases in accordance with the value of the connectivity threshold. On the unweighted graph, this downtrend is especially strong on the MovieLens corpus: when the threshold value is fixed to $\theta = 0.5$, the size of the community is divided by 2, and made up on average of 52 users. This number is similar to the size of the communities used in the experiments we conducted with the KNN approach. At the opposite, when using the weighted graph, it decreases more slowly. For example, when the connectivity threshold is set to $\theta = 0.9$, the size of the community is only decreased by 35%, whereas it decreased by more than 90% when using the unweighted graph. The reason of this small decrease is that, when managing weighted edges, the algorithm tends to insert in the communities users with high-valued links. Users with high-valued links are users who tend to have consulted a lot of items, thus users highly connected.

A similar conclusion can be drawn on the Jester dataset. However, when θ is fixed to 0.5, the decrease of the size of the communities is lower, which is due to the high connectivity of the nodes. An important decrease is obtained when the threshold value exceeds 0.7, which corresponds to the threshold value above which the precision increases.

5.8. Similarities between Rating-Based and Community-Based Mentors. In the previous section, the experiments showed that, on the MovieLens dataset, the F-LCD algorithm, with an average community size of 48 users, leads to a precision value which is not very different to the one obtained when exploiting the KNN approach with ratings and $K = 50$. Thus we can ask in what measure the two sets of communities, that have similar average sizes, are formed by the same users.

Table 3: Quartile, median and maximal values of the number of communities a user belongs to, on the MovieLens dataset.

	1st quartile	Median	3rd quartile	Maximal
Unweighted edges	1	10	183	254
KNN	2	35	123	267

We can first say that the maximal size of communities when using KNN is 50, whereas it is 80 for the F-LCD algorithm (Table 1). To further analyze the communities, we propose to evaluate the average number of communities a user belongs to, for each approach. As the sizes of the communities are equal on average, the average number of communities a user belongs to is equivalent. However, the repartition, in terms of median and quartile numbers may be different. Table 3 presents this repartition.

The distribution of the number of communities the users belong to are different. When focusing on the maximal number of communities users belong to, both algorithms have a similar value: about 260, which represents 28% of the communities. This means that each user belongs to less than 28% of the communities. Nevertheless, when using the F-LCD algorithm, half of the users belong to less than 10 communities, whereas they belong to less than 35 communities with KNN. Thus, with F-LCD, users tend to belong to either few communities, or many communities, whereas the repartition seems to be more homogeneous with KNN.

On the Jester dataset, a similar experiment has been conducted. With KNN, we built communities with $K = 172$ (to have similar average sizes of communities than the optimal precision of F-LCD). The distribution of the number of users in the communities also differ. Specifically, the first quartile and the median are twice smaller with F-LCD compared to those with KNN.

6. Discussion

This paper was dedicated to the identification of a good way to detect mentors in a rating-free collaborative filtering system.

As presented in Section 2.5, in the absence of ratings, existing algorithms of the literature usually consider that the more two users have coconsulted two items, the more they are likely to have similar preferences [41, 42]. So as to confirm or invalidate this hypothesis, we compared the precision measures of two different mentor selection strategies.

(1) Our baseline consists in defining the mentors as the whole set of connected users (referred to as "All connected users" in Figures 4 and 5).

(2) K nearest mentors have been selected on the basis of their number of coconsulted items with the active user (see label "# Co-consulted Items" in Figures 4 and 5).

The resulting precisions of the second strategy are not increased in comparison with our baseline. They even

decrease the precision by 2% and 6% on MovieLens and Jester, respectively. We can thus deduce that the number of coconsultations between two users does not reliably reflect their similarity of preferences. This comes to disprove the optimality of existing approaches.

Following this observation, we decided to not manage any similarity information between users, and made several proposals and refinements to improve the mentor selection process through a graph-theoretic approach. We represented the set of users under the form of an unweighted graph, where users are linked if they have coconsulted a sufficient number of items. This number of coconsulted items was first used to prefilter mentors. Subsequently, we exploited the connectivity within this graph to include the most relevant candidates in the set of mentors. Our investigations led us to propose the new model UC-LCD inspired from local community detection algorithms. The precision reached by this model was similar to the one obtained when all connected users are taken into account (our baseline), while managing a set of mentors greatly smaller (a reduction of 40% and 68% is observed). We refined this model by discarding, during the process of community detection, the mentors subconnected within the set other mentors (model F-LCD). The resulting precisions of F-LCD have been increased in comparison with UC-LCD, while decreasing the set of mentors of 52% in MovieLens and 96% in Jester.

As presented in Section 2.5, the item-based approach is often used to compute recommendations in a rating-free context [40]. Thus, we also compared the previous precision measures (UC-LCD and F-LCD) to the one of an item-based approach (see label "Item-based" in Figures 4 and 5). Results show that UC-LCD and F-LCD are on average, respectively, 4.25% and 6.2% better than the item-based approach.

As a conclusion, if the number of coconsulted items did not appear to be a reliable measure to detect mentors, the connectivity (both between the active user and his/her mentors, and between each pair of mentors) has been proven to be much more adequate within the frame of these experiments. An improvement of the precision is obtained while requiring a small set of mentors.

The reason why we chose to conduct the experiments on both MovieLens and Jester datasets lies in their various characteristics. These datasets allow us to reliably validate the robustness and relevancy of our models. The MovieLens dataset has a high sparsity level (94%) whereas the sparsity level on Jester is only 27%. On the MovieLens dataset, each user has consulted at least 1% of the items whereas on the Jester dataset each user has consulted at least 30% of the items. This means that the connectivities between users are very different in MovieLens and in Jester. In addition, the rating scales are very different, the scale used in Jester is four times larger than the one in MovieLens. Among other noteworthy differences, we noticed exploiting the users' ratings, the precision values of the two corpora differ of roughly 20%. We also conducted an experiment to compare the precision of the recommendations with and without ratings. We showed that, on the Jester dataset, exploiting the rating values is of high importance and largely increases the precision value, compared to using only consultations. At the opposite, on the MovieLens dataset, ignoring the rating values has a smaller impact. This difference may be explained by the difference of the rating scales.

Despite these differences of characteristics, the improvement of the precision in a rating-free context thank to our models have been confirmed in these two corpora. Our rating-free model F-LCD leads to an increase in precision of 1% on MovieLens, and 3% on Jester, compared to the rating-free baseline, while decreasing the number of mentors by 71% and 99%, respectively. On MovieLens, the corresponding precision is even only 1% lower compared to the one obtained when managing ratings. We can conclude that the connectivity is a good information that better reflects similarity between users than the number of coconsulted items. Of course, the optimal connectivity threshold differs according to the graphs.

7. Conclusion

This paper focused on the mentor selection problem in the frame of user-based collaborative filtering. Classical approaches of mentor selection rely on a similarity value between users. This similarity is computed on the basis of user-provided ratings, that reflect their preferences on the items. Two main approaches of mentor selection are used in the literature. The first approach defines mentors of a user a as the users with the highest similarity value. The second approach clusters users thanks to their similarity value, and users are considered as mentors of each other within a cluster.

In this work, we have addressed the problem of mentor selection when no user-provided ratings are available. In that case, no similarity value between users can be precisely computed; thus no mentor selection can be easily made either. Nevertheless, the set of user consultations is available. Exploiting the number of coconsulted items is a way to estimate the similarity between users. We proposed to not exploit this information to deduce the similarity between users. The approach we proposed if made up of two stages. First, we considered that two users who have coconsulted more than a predefined number of items are potentially similar users; their similarity value is fixed to 1. This approach has the advantage to make the design of the similarity matrix easier than the classical approach. Indeed, for each pair of users, the computation of the value of the similarity (0 or 1) comes down to a simple count that can be stopped when the minimum number of coconsulted items has been reached.

Second, we represent the set of users under the form of an unweighted graph and we exploit local community detection algorithms to form communities of users, and deduce the mentors, within this context. Such algorithms exploit the structure of the graph and do not pay attention to the value of the edges. In addition, they have a local view of the graph, which allows to design a community for each user, resulting in overlapping communities. Used in the context of collaborative filtering, these algorithms have the advantage of both the direct neighbor selection and classification of users of classical approaches.

We adapted a state-of-the-art local community detection algorithm so as to discover communities that fit the characteristics of collaborative filtering: the communities have to be user-centered and have to be strictly made up of directly connected users. The users that belong to the community of a given user *a* are his mentors.

We then proposed to further refine the set of mentors by filtering out subconnected mentors in the communities so as to have communities made up of only highly connected users. We assumed that the more the set of mentors of a user are connected, the more the quality of the set of mentors is high.

To the best of our knowledge, the exploitation of the structure of the graph has been rarely studied in the frame of collaborative filtering, especially to perform mentor detection.

This approach has been tested on two datasets with various characteristics: rating scale, number of users and items, connectivity of the graph, and so forth. Experimental results have shown that our local community detection algorithm F-LCD improves the precision compared to the baseline model that uses the whole set of connected users (from 1% to 3%). In addition, the number of mentors used is dramatically decreased (up to 99%).

Thus, we have shown that, when the user-provided ratings are not available, mentor selection can however be performed by exploiting the connectivity between mentors in place of their similarity values, while reaching a good precision value.

As a future work, we plan to study the use of local community detection algorithms when ratings are available. The challenge is thus how to accurately exploit similarities of ratings in these algorithms.

Among perspectives, we also propose to extend our model to security issues. Collaborative Filtering is well-known for being very vulnerable to malicious attacks [84], since it uses the opinion of a community of similar users to predict the opinion of a current user. Thus, the problem consists in automatically making the difference—among the global set of users—between the leaders who helps building relevant recommendations, and attackers who aims at degrading the service or influencing users. Analyzing connectivity between users will help to reach this objective.

References

[1] I. ChoiceStream, "Choicestream personalization survey," 2006.

[2] S. Castagnos, N. Jones, and P. Pu, "Eye-tracking product recommender's usage," in *Proceedings of the 4th ACM Conference on Recommender Systems (RecSys '10)*, pp. 29–36, Barcelona, Spain, 2010.

[3] M. Pazzani and D. Billsus, "The adaptive web," in *Content-Based Recommendation Systems*, Springer, Berlin, Germany, 2007.

[4] R. D. Burke, K. J. Hammond, and B. C. Young, "Knowledge-based navigation of complex information spaces," in *Proceedings of the 13th National Conference on Artificial Intelligence (AAAI '96)*, vol. 1, pp. 462–468, Portland, Ore, USA, August 1996.

[5] D. Goldberg, D. Nichols, B. Oki, and D. Terry, "Using collaborative filtering to weave an information tapestry," *Communications of the ACM*, vol. 35, no. 12, pp. 61–70, 1992.

[6] J. Wang, A. P. de Vries, and M. J. T. Reinders, "Unifying user-based and item-based collaborative filtering approaches by similarity fusion," in *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 501–508, Seatttle, Wash, USA, August 2006.

[7] S. Bruninghaus and K. D. Ashley, "Toward adding knowledge to learning algorithms for indexing legal cases," in *Proceedings of the 7th International Conference on Artificial Intelligence and Law*, pp. 9–17, June 1999.

[8] K.-Y. Jung, D.-H. Park, and J.-H. Lee, "Hybrid collaborative filtering and content-based filtering for improved recommender system," in *Proceedings of the International Conference on Computational Science (ICCS '04)*, vol. 3036 of *Lecture Notes in Computer Science*, pp. 295–302, 2004.

[9] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl, "Evaluating collaborative filtering recommender systems," *ACM Transactions on Information Systems*, vol. 22, no. 1, pp. 5–53, 2004.

[10] A. Brun, A. Hamad, O. Buffet, and A. Boyer, "Towards preference relations in recommender systems," in *Proceedings of the Workshop on Preference Learning, European Conference on Machine Learning and Principle and Practice of Knowledge Discovery in Databases (ECML-PKDD '10)*, 2010.

[11] S. Fortunato, "Community detection in graphs," *Physics Reports*, vol. 486, no. 3–5, pp. 75–174, 2009.

[12] J. Chen, R. Zaane, and R. Goebel, "Local community identification in social networks," in *Proceedings of the Advances in Social Network Analysis and Mining*, pp. 237–242, 2009.

[13] J. Breese, D. Heckerman, and C. Kadie, "Empirical analysis of predictive algorithms for collaborative filtering," in *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence (UAI '98)*, 1998.

[14] B. M. Sarwar, G. Karypis, J. Konstan, and J. Reidl, "Item-based collaborative filtering recommendation algorithms," in *Proceedings of the 10th International Conference on World Wide Web (WWW '01)*, pp. 285–295, 2001.

[15] L. Candillier, F. Meyer, and M. Boullé, "Comparing state-of-the-art collaborative filtering systems," in *Proceedings of the 5th International Conference on Machine Learning and Data Mining in Pattern Recognition (MLDM '07)*, vol. 4571 of *Lecture Notes in Computer Science*, pp. 548–562, Leipzig, Germany, July 2007.

[16] S. Castagnos and A. Boyer, "A client/server user-based collaborative filtering algorithm: model and implementation," in *Proceedings of the 17th European Conference on Artificial Intelligence (ECAI '06)*, pp. 617–621, 2006.

[17] S. Castagnos and A. Boyer, "Personalized communities in a distributed recommender system," in *Proceedings of the 29th European Conference on IR Research (ECIR '07)*, vol. 4425 of *Lecture Notes in Computer Science*, pp. 343–355, Rome, Italy, April 2007.

[18] N. Lathia, S. Hailes, and L. Capra, "Trust-based collaborative filtering," *International Federation for Information Processing*, vol. 263, pp. 119–134, 2008.

[19] G. Amati, C. Carpineto, and G. Romano, "An effective threshold-based neighbor selection in collaborative filtering," in *Proceedings of European Conference on Information Retrieval (ECIR '07)*, pp. 712–715, 2007.

[20] J. Herlocker, J. Konstan, A. Borchers, and J. Riedl, "An algorithmic framework for performing collaborative filtering," in *Proceedings of the SIGIR Conference*, pp. 230–237, 1999.

[21] M. Grcar, B. Fortuna, and D. Mladenic, "kNN versus SVM in the collaborative filtering framework," in *Proceedings of the 7th WEBKDD Workshop on Knowledge Discovery from the Web (WEBKDD '05)*, 2005.

[22] B. Mobasher, R. Burke, and J. J. Sandvig, "Model-based collaborative filtering as a defense against profile injection attacks," in *Proceedings of the 21st National Conference on Artificial Intelligence and the 18th Innovative Applications of Artificial Intelligence Conference (AAAI '06)*, vol. 2, pp. 1388–1393, Boston, Mass, USA, July 2006.

[23] L. Terán and A. Meier, "A fuzzy recommender system for eElections," in *Proceedings of the 1st International Conference on Electronic Government and Information Systems Perspective (EGOVIS '10)*, vol. 6267 of *Lecture Notes in Computer Science*, pp. 62–76, Bilbao, Spain, August-September 2010.

[24] M. O'Connor and J. Herlocker, "Clustering items for collaborative filtering," in *Proceedings of the 22nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '99)*, 1999.

[25] L. Ungar and D. Foster, "Clustering methods for collaborative filtering," in *Proceedings of AAAI Workshop on Recommendation Systems*, 1998.

[26] A. M. Martinez, P. Mittrapiyanuruk, and A. C. Kak, "On combining graph-partitioning with non-parametric clustering for image segmentation," *Computer Vision and Image Understanding*, vol. 95, no. 1, pp. 72–85, 2004.

[27] P. S. Bradley and U. M. Fayyad, "Refining initial points for *k*-means clustering," in *Proceedings of the 15th International Conference on Machine Learning (ICML '98)*, pp. 91–99, Morgan Kaufmann, May 1998.

[28] P. Perny and J. Zucker, "Collaborative filtering methods based on fuzzy preference relations," in *Proceedings of the 4th Meeting of the Euro Working Group on Fuzzy Sets (EUROFUSE-SIC '99)*, 1999.

[29] R. A. Jarvis and E. A. Patrick, "Clustering using a similarity measure based on shared near neighbors," *IEEE Transactions on Computers*, vol. 22, no. 11, pp. 1025–1034, 1973.

[30] L. Ertöz, M. Steinbach, and V. Kumar, "Information retrivial and clustering," in *Finding Topics in Collections of Documents: A Shared Nearest Neighbor Approach*, W. Wu, H. Xiong, and S. Shekhar, Eds., 2002.

[31] S. Guha, R. Rastogi, and K. Shim, "CURE: an efficient clustering algorithm for large databases," in *Proceedings of the ACM-SIGMOD International Conference on Management of Data (SIGMOD '98)*, no. 2, pp. 73–84, 1998.

[32] J. Yin, X. Fan, Y. Chen, and J. Ren, "High-dimensional shared nearest neighbor clustering algorithm," in *Proceedings of the 2nd International Conference on Fuzzy Systems and Knowledge Discovery (FSKD '05)*, vol. 3614 of *Lecture Notes in Computer Science*, pp. 494–502, Changsa, China, August 2005.

[33] G. Xue, C. Lin, Q. Yang et al., "Scalable collaborative filtering using cluster-based smoothing," in *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '05)*, 2005.

[34] N. Srinivasa and S. Medasani, "Active fuzzy clustering for collaborative filtering," in *Proceedings of IEEE International Conference on Fuzzy Systems*, pp. 1697–1702, July 2004.

[35] J. B. Schafer, J. A. Konstan, and J. Riedl, "E-commerce recommendation applications," *Data Mining and Knowledge Discovery*, vol. 5, no. 1-2, pp. 115–153, 2001.

[36] X.-M. Jiang, W.-G. Song, and W.-G. Feng, "Optimizing collaborative filtering by interpolating the individual and group behaviors," in *Proceedings of the 8th Asia-Pacific Web Conference (APWeb '06)*, vol. 3841 of *Lecture Notes in Computer Science*, pp. 568–578, Harbin, China, January 2006.

[37] D. Oard and J. Kim, "Implicit feedback for recommender systems," in *Proceedings of the AAAI Workshop on Recommender Systems*, pp. 81–83, 1998.

[38] S. Castagnos, *Modélisation de comportements et apprentissage stochastique non supervisé de stratégies d'interactions sociales au sein de systèmes temps réel de recherche et d'accès à l'information*, Ph.D. thesis, Nancy University, 2008.

[39] S. Castagnos and A. Boyer, "Privacy concerns when modeling users in collaborative filtering recommender systems," in *Social and Human Elements of Information Security: Emerging Trends and Countermeasures*, 2008.

[40] G. Karypis, "Evaluation of item-based top-N recommendation algorithms," in *Proceedings of the ACM 10th International Conference on Information and Knowledge Management (CIKM '01)*, pp. 247–254, November 2001.

[41] G. Linden, B. Smith, and J. York, "Amazon.com recommendations: item-to-item collaborative filtering," *IEEE Internet Computing*, vol. 7, no. 1, pp. 76–80, 2003.

[42] C. Miranda and A. M. Jorge, "Incremental collaborative filtering for binary ratings," in *Proceedings of IEEE/WIC/ACM International Conference on Web Intelligence (WI '08)*, pp. 389–392, Sydney, Australia, December 2008.

[43] J. Redpath, D. H. Glass, S. McClean, and L. Chen, "Collaborative filtering: the aim of recommender systems and the significance of user ratings," in *Proceedings of the 32nd European Conference on Information Retrieval (ECIR '10)*, vol. 5993 of *Lecture Notes in Computer Science*, pp. 394–406, Milton Keynes, UK, March 2010.

[44] B. Mobasher, H. Dai, T. Luo, and M. Nakagawa, "Improving the effectiveness of collaborative filtering on anonymous web usage data," in *Proceedings of the IJCAI 2001 Workshop on Intelligent Techniques for Web Personalization (ITWP '01)*, 2001.

[45] G. Bonnin, A. Brun, and A. Boyer, "Web Intelligence and Intelligent Agents, chap. Skipping-Based Collaborative Recommendations Inspired from Statistical Language Modeling," Zeeshan-ul-Hassan Usmani, March 2010.

[46] G. W. Flake, R. Tarjan, and K. Tsioutsiouliklis, "Graph clustering and minimum cut trees," *Internet Mathematics*, vol. 1, no. 4, pp. 385–408, 2004.

[47] D. Zhou, J. Huang, and B. Schölkopf, "Learning from labeled and unlabeled data on a directed graph," in *Proceedings of the 22nd International Conference on Machine Learning (ICML'05)*, pp. 1036–1043, 2005.

[48] K. Tsuda and W. S. Noble, "Learning kernels from biological networks by maximizing entropy," *Bioinformatics*, vol. 20, no. 1, pp. 326–333, 2004.

[49] J. Callut, K. Franoisse, M. Saerens, and P. Dupont, "Semi-supervised classification in graphs using bounded random walks," in *Proceedings of the 17th Annual Machine Learning Conference of Belgium and the Netherlands (BENELEARN '08)*, pp. 67–68, 2008.

[50] S. E. Schaeffer, "Graph clustering," *Computer Science Review*, vol. 1, no. 1, pp. 27–64, 2007.

[51] S. Papadopoulos, Y. Kompatsiaris, and A. Vakali, "Leveraging collective intelligence through community detection in tag networks," in *Proceedings of Workshop on Collective Knowledge Capturing and Representation (CKCaR '09)*, 2009.

[52] M. Girvan and M. E. J. Newman, "Community structure in social and biological networks," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 99, no. 12, pp. 7821–7826, 2002.

[53] N. Mishra, R. Schreiber, I. Stanton, and R. E. Tarjan, "Clustering social networks," in *Proceedings of Weaving the Astronomy Web (WAW '07)*, Lecture Notes in Computer Science, pp. 56–67, 2007.

[54] P. Wanjantuk and J. A. Keane, "Finding related documents via communities in the citation graph," in *Proceedings of the IEEE International Symposium on Communications and Information Technologies (ISCIT '04)*, vol. 1, pp. 445–450, Sapporo, Japan, October 2004.

[55] M. Rosvall and C. T. Bergstrom, "Maps of random walks on complex networks reveal community structure," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 105, no. 4, pp. 1118–1123, 2008.

[56] G. W. Flake, S. Lawrence, and C. L. Giles, "Efficient identification of web communities," in *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '01)*, pp. 150–160, Boston, Mass, USA, August 2000.

[57] Y. Wang, D. Chakrabarti, C. Wang, and C. Faloutsos, "Epidemic spreading in real networks: an eigenvalue viewpoint," in *Proceedings of the 22nd International Symposium on Reliable Distributed Systems (SRDS '03)*, pp. 25–34, Florence, Italy, October 2003.

[58] Y. Qi, F. Balem, C. Faloutsos, J. Klein-Seetharaman, and Z. Bar-Joseph, "Protein complex identification by supervised graph local clustering," *BioInformatics*, vol. 24, no. 13, pp. 250–268, 2008.

[59] L. Tang, "Community detection in social networks," http://www.public.asu.edu/~huanliu/dmml_presentation/2008/.

[60] R. Alba, "A graph-theoretic definition of a sociometric clique," *Journal of Mathematical Sociology*, pp. 112–126, 1973.

[61] L. Freeman, "A set of measures of centrality based on betweenness," *Sociometry*, vol. 40, pp. 35–41, 1977.

[62] H. Ino, M. Kudo, and A. Nakamura, "Partitioning of web graphs by community topology," in *Proceedings of the 14th International Conference on World Wide Web*, pp. 661–669, 2005.

[63] B. Yang, W. K. Cheung, and J. Liu, "Community mining from signed social networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 10, pp. 1333–1348, 2007.

[64] S. Gregory, "An algorithm to find overlapping community structure in networks," in *Proceedings of the 11th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD '07)*, vol. 4702 of *Lecture Notes in Computer Science*, pp. 91–102, Warsaw, Poland, September 2007.

[65] J. Scott, *Social Network Analysis: A Handbook*, Sage, London, UK, 2nd edition, 2000.

[66] D. Chakrabarti, "AutoPart: parameter-free graph partitioning and outlier detection," in *Proceedings of the Principles and Practice of Knowledge Discovery in Databases Conference (PKDD '04)*, vol. 3202 of *Lecture Notes in Computer Science*, pp. 112–124, 2004.

[67] A. Pother, "Graph partitioning algorithms with applications to scientific computing," Tech. Rep., Norfolk, Va, USA, 1997.

[68] F. Radicchi, C. Castellano, F. Cecconi, V. Loreto, and D. Paris, "Defining and identifying communities in networks," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 101, no. 9, pp. 2658–2663, 2004.

[69] A. Clauset, M. Newmann, and C. Moore, "Finding community structure in very large networks," *Physical Review*, vol. 72, 2005.

[70] F. Luo, J. Z. Wang, and E. Promislow, "Exploring local community structures in large networks," in *Proceedings of IEEE/WIC/ACM International Conference on Web Intelligence (WI '06)*, pp. 233–239, December 2006.

[71] L. Branting, "Incremental detection of local community structure," in *Proceedings of the International Conference on Advances in Social Networks Analysis and Mining*, pp. 80–87, 2010.

[72] J. Tian, D. Chen, and Y. Fu, "A new local algorithm for detecting communities in networks," in *Proceedings of the 1st International Workshop on Education Technology and Computer Science (ETCS '09)*, vol. 2, pp. 721–724, Wuhan, China, March 2009.

[73] Q. Song and N. Kasabov, "Foundations of cognitive science," in *ECM—A Novel On-Line, Evolving Clustering Method and Its Applications*, M. I. Posner, Ed., pp. 631–682, 2001.

[74] J. G. Booth, G. Casella, and J. P. Hobert, "Clustering using objective functions and stochastic search," *Journal of the Royal Statistical Society B*, vol. 70, no. 1, pp. 119–139, 2008.

[75] C. C. Aggarwal, J. L. Wolf, K. lung Wu, and P. S. Yu, "Horting hatches an egg: a new graph-theoretic approach to collaborative filtering," in *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 201–212, ACM Press, 1999.

[76] M. D. Ekstrand, P. Kannan, J. A. Stemper, J. T. Butler, J. A. Konstan, and J. T. Riedl, "Automatically building research reading lists," in *Proceedings of the 4th ACM Conference on Recommender Systems (RecSys '10)*, pp. 159–166, Barcelona, Spain, September 2010.

[77] P. Viappiani, B. Faltings, and P. Pu, "Preference-based search using example-critiquing with suggestions," *Journal of Artificial Intelligence Research*, vol. 27, pp. 465–503, 2006.

[78] V. Schickel-Zuber and B. Faltings, "Using hierarchical clustering for learning theontologies used in recommendation systems," in *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '07)*, pp. 599–608, San Jose, Calif, USA, August 2007.

[79] L. Baltrunas and F. Ricci, "Dynamic item weighting and selection for collaborative filtering," in *Web Mining 2.0 Workshop, ECML-PKDD*, Springer, 2007.

[80] C. Ziegler, *Towards decentralized recommender systems*, Ph.D. thesis, Albert-Ludwigs-Universitt Freiburg, 2005.

[81] A. Brun, G. Bonnin, and A. Boyer, "History dependent recommender systems based on partial matching," in *Proceedings of the 17th International Conference on User Modeling, Adaptation, and Personalization (UMAP '09)*, vol. 5535 of *Lecture Notes in Computer Science*, pp. 343–348, Trento, Italy, June 2009.

[82] U. Shardanand and P. Maes, "Social information filtering: algorithms for automating "word of mouth"," in *Proceedings of the Conference on Human Factors in Computing Systems (CHI '95)*, vol. 1, pp. 210–217, May 1995.

[83] J. Herlocker, J. A. Konstan, and J. Riedl, "An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms," *Information Retrieval*, vol. 5, no. 4, pp. 287–310, 2002.

[84] R. Burke and B. Mobasher, "Trust and bias in multi-agent recommender systems," in *Proceedings of Workshop on Multi-Agent Information Retrieval and Recommender Systems, in Conjunction with the 19th International Joint Conference on Artificial Intelligence (IJCAI '05)*, Edinburgh, Scotland, 2005.