Physics Reports 519 (2012) 1–49

Contents lists available at SciVerse ScienceDirect

Physics Reports

journal homepage: www.elsevier.com/locate/physrep

Recommender systems

Linyuan Lü [a,b,c], Matúš Medo [b], Chi Ho Yeung [b,d], Yi-Cheng Zhang [b,*], Zi-Ke Zhang [a,b,c], Tao Zhou [a,b,c,e]

[a] Institute of Information Economy, Alibaba Business School, Hangzhou Normal University, Hangzhou, 310036, PR China
[b] Department of Physics, University of Fribourg, Fribourg, CH-1700, Switzerland
[c] Web Sciences Center, University of Electronic Science and Technology of China, Chengdu, 610054, PR China
[d] The Nonlinearity and Complexity Research Group, Aston University, Birmingham B4 7ET, United Kingdom
[e] Beijing Computational Science Research Center, Beijing, 100084, PR China

ARTICLE INFO

Article history:
Accepted 7 February 2012
Available online 6 March 2012
editor: I. Procaccia

Keywords:
Recommender systems
Information filtering
Networks

ABSTRACT

The ongoing rapid expansion of the Internet greatly increases the necessity of effective recommender systems for filtering the abundant information. Extensive research for recommender systems is conducted by a broad range of communities including social and computer scientists, physicists, and interdisciplinary researchers. Despite substantial theoretical and practical achievements, unification and comparison of different approaches are lacking, which impedes further advances. In this article, we review recent developments in recommender systems and discuss the major challenges. We compare and evaluate available algorithms and examine their roles in the future developments. In addition to algorithms, physical aspects are described to illustrate macroscopic behavior of recommender systems. Potential impacts and future directions are discussed. We emphasize that recommendation has great scientific depth and combines diverse research fields which makes it interesting for physicists as well as interdisciplinary researchers.

Contents

* Corresponding author at: Institute of Information Economy, Hangzhou Normal University, Hangzhou, 310036, PR China.
E-mail address: yi-cheng.zhang@unifr.ch (Y.-C. Zhang).

0370-1573/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.physrep.2012.02.006

1. Introduction

Thanks to computers and computer networks, our society is undergoing rapid transformation in almost all aspects. We buy online, gather information by search engines and live a significant part of our social life on the Internet. The fact that many of our actions and interactions are nowadays stored electronically gives researchers the opportunity to study socio-economical and techno-social systems at much better level of detail. Traditional "soft sciences", such as sociology or economics, have their fast-growing branches relying on the study of these newly available massive data sets [1,2]. Physicists, with their long experience with data-driven research, have joined this trend and contributed to many fields such as finance [3,4], network theory [5–9] and social dynamics [10] which are outside their traditional realm. The study of recommender systems and information filtering in general is no exception with the interest of physicists steadily increasing over the past decade. The task of recommender systems is to turn data on users and their preferences into predictions of users' possible future likes and interests. The study of recommender systems is at crossroads of science and socio-economic life and its huge potential was first noticed by web entrepreneurs in the forefront of the information revolution. While being originally a field dominated by computer scientists, recommendation calls for contributions from various directions and is now a topic of interest also for mathematicians, physicists, and psychologists. For instance, it is not a coincidence that an approach based on what psychologists know about human behavior scored high in a recent recommendation contest organized by the commercial company Netflix [11].

When computing recommendations for a particular user, the very basic approach is to select the objects favored by other users that are similar to the target user. Even this simple approach can be realized in a multitude of ways—this is because the field of recommendation lacks general "first principles" from which one could deduce the right way to recommend. For example, how best to measure user similarity and assess its uncertainty? How to aggregate divergent opinions from various users? How to handle users for whom little information is available? Should all data be trusted equally or can one detect reckless or intentionally misleading opinions? These and similar issues arise also when methods more sophisticated than those based on user similarity are used. Fortunately, there exist a number of real data sets that can be used to measure and compare performance of individual methods. In consequence, similarly to physics, it is the experiment what decides which recommendation approach is good and which is not.

It would be very misleading to think that recommender systems are studied only because suitable data sets are available. While the availability of data is important for empirical evaluation of recommendation methods, the main driving force comes from practice: electronic systems give us too much choice to handle by ourselves. The interest from industry is hardly surprising—an early book on the nascent field of recommendation, *Net Worth* by John Hagel III and Marc Singer [12], clearly pointed out the enormous economic impact of "info-mediaries" who can greatly enhance individual consumers' information capabilities. Most e-commerce web sites now offer various forms of recommendation—ranging from simply showing the

most popular items or suggesting other products by the same producer to complicated data mining techniques. People soon realized that there is no unique best recommendation method. Rather, depending on the context and density of the available data, different methods adapting to particular applications are most likely to succeed. Hence there is no panacea, and the best one can do is to understand the underlying premises and recommender mechanisms, then one can tackle many diverse application problems from the real life examples. This is also reflected in this review where we do not try to highlight any ultimate approach to recommendation. Instead, we review the basic ideas, methods and tools with particular emphasis on physics-rooted approaches.

The motivation for writing this review is multifold. Firstly, while extensive reviews of recommender systems by computer scientists already exists [13–15], the view of physicists is different from that of computer scientists by using more the complex networks approach and adapting various classical physics processes (such as diffusion) for information filtering. We thus believe that this review with its structure and emphasis on respective topics can provide a novel point of view. Secondly, the past decade has already seen a growing interest of physicists in recommender systems and we hope that this review can be a useful source for them by describing the state of the art in language which is more familiar to the physics community. Finally, the interdisciplinary approach presented here might provide new insights and solutions for open problems and challenges in the active field of information filtering.

This review is organized as follows. To better motivate the problem, In Section 2 we begin with a discussion of real applications of recommender systems. Next, in Section 3 we introduce basic concepts – such as complex networks, recommender systems, and metrics for their evaluation – that form a basis for all subsequent exposition. Then we proceed to description of recommendation algorithms where traditional approaches (similarity-based methods in Section 4 and dimensionality reduction techniques in Section 5) are followed by network-based approaches which have their origin in the random walk process well known to all physicists (in Section 6). Methods based on external information, such as social relationships (in Section 7), keywords or time stamps (in Section 8), are also included. We conclude with a brief evaluation of methods' performance in Section 9 and a discussion on the outlook of the field in Section 10.

2. Real applications of recommender systems

Thanks to the ever-decreasing costs of data storage and processing, recommender systems gradually spread to most areas of our lives. Sellers carefully watch our purchases to recommend us other goods and enhance their sales, social web sites analyze our contacts to help us connect with new friends and get hooked with the site, and online radio stations remember skipped songs to serve us better in the future (see more examples in Table 1). In general, whenever there is plenty of diverse products and customers are not alike, personalized recommendation may help to deliver the right content to the right person. This is particularly the case for those Internet-based companies that try to make use of the so-called long-tail [16] of goods which are rarely purchased yet due to their multitude they can yield considerable profits (sometimes they are referred to as "worst-sellers"). For example on Amazon, between 20% and 40% of sales is due to products that do not belong to the shop's 100,000 most sold products [17]. A recommender system may hence have significant impact on a company's revenues: for example, 60% of DVDs rented by Netflix are selected based on personalized recommendations.[1]

As discussed in [18], recommender systems not only help decide which products to offer to an individual customer, they also increase cross-sell by suggesting additional products to the customers and improve consumer loyalty because consumers tend to return to the sites that best serve their needs (see [19] for an empirical analysis of the impact of recommendations and consumer feedback on sales at Amazon.com).

Since no recommendation method serves best all customers, major sites are usually equipped with several distinct recommendation techniques ranging from simple popularity-based recommendations to sophisticated techniques many of which we shall encounter in the following sections. Further, new companies emerge (see, for example, string.com) which aim at collecting all sorts of user behavior (ranging from pages visited on the web and music listened on a personal player to "liking" or purchasing items) and using it to provide personalized recommendations of different goods or services.

2.1. Netflix prize

In October 2006, the online DVD rental company Netflix released a data set containing approximately 100 million anonymous movie ratings and challenged researchers and practitioners to develop recommender systems that could beat the accuracy of the company's recommendation system, Cinematch [20]. Although the released data set represented only a small fraction of the company's rating data, thanks to its size and quality it fast became a standard in the data mining and machine learning community. The data set contained ratings in the integer scale from 1 to 5 which were accompanied by dates. For each movie, title and year of release were provided. No information about users was given. Submitted predictions were evaluated by their root mean squared error (RMSE) on a qualifying data set containing over 2817,131 unknown ratings. Out of 20,000 registered teams, 2000 teams submitted at least one answer set. On 21 September 2009, the grand prize of $1000,000 was awarded to a team that overperformed the Cinematch's accuracy by 10%. At the time when the contest

[1] As presented by Jon Sanders (Recommendation Systems Engineering, Netflix) during the talk "Research Challenges in Recommenders" at the 3rd ACM Conference on Recommender Systems (2009).

Table 1
Popular sites using recommender systems. Besides, there are also some companies devoting themselves to recommendation techniques, such as Baifendian (www.baifendian.com), Baynote (www.baynote.com), ChoiceStream (www.choicestream.com), Goodrec (www.goodrec.com), and others.

Site	What is recommended
Amazon	Books/other products
Facebook	Friends
WeFollow	Friends
MovieLens	Movies
Nanocrowd	Movies
Jinni	Movies
Findory	News
Digg	News
Zite	News
Meehive	News
Netflix	DVDs
CDNOW	CDs/DVDs
eHarmony	Dates
Chemistry	Dates
True.com	Dates
Perfectmatch	Dates
CareerBuilder	Jobs
Monster	Jobs
Pandora	Music
Mufin	Music
StumbleUpon	Web sites

was closed, there were two teams that achieved the same precision. The prize was awarded to the team that submitted their results 20 min earlier than the other one. (See [11] for a popular account on how the participants struggled with the challenge.)

There are several lessons that we have learned in this competition [21]. Firstly, the company gained publicity and a superior recommendation system that is supposed to improve user satisfaction. Secondly, ensemble methods showed their potential of improving accuracy of the predictions.[2] Thirdly, we saw that accuracy improvements are increasingly demanding when RMSE drops below a certain level. Finally, despite the company's effort, anonymity of its users was not sufficiently ensured [25]. As a result, Netflix was sued by one of its users and decided to cancel a planned second competition.

2.2. Major challenges

Researchers in the field of recommender systems face several challenges which pose danger for the use and performance of their algorithms. Here we mention only the major ones:

1. *Data sparsity*. Since the pool of available items is often exceedingly large (major online bookstores offer several millions of books, for example), overlap between two users is often very small or none. Further, even when the average number of evaluations per user/item are high, they are distributed among the users/items very unevenly (usually they follow a power-law distribution [26] or a Weibull distribution [27]) and hence majority of users/items may have expressed/received only a few ratings. Hence, an effective recommender algorithm must take the data sparsity into account [28].
2. *Scalability*. While the data is mostly sparse, for major sites it includes millions of users and items. It is therefore essential to consider the computational cost issues and search for recommender algorithms that are either little demanding or easy to parallelize (or both). Another possible solution is based on using incremental versions of the algorithms where, as the data grows, recommendations are not recomputed globally (using the whole data) but incrementally (by slightly adjusting previous recommendations according to the newly arrived data) [29,30]. This incremental approach is similar to perturbation techniques that are widely used in physics and mathematics [31].
3. *Cold start*. When new users enter the system, there is usually insufficient information to produce recommendation for them. The usual solutions of this problem are based on using hybrid recommender techniques (see Section 8.4) combining content and collaborative data [32,33] and sometimes they are accompanied by asking for some base information (such as age, location and preferred genres) from the users. Another way is to identify individual users in different web services.

[2] The ensemble methods deal with the selection and organization of many individual algorithms to achieve better prediction accuracy. In fact, the winning team, called BellKor's Pragmatic Chaos, was a combined team of BellKor [22], Pragmatic Theory [23] and BigChaos [24] (of course, it was not a simple combination but a sophisticated design), and each of them consists of many individual algorithms. For example, the Pragmatic Theory solution considered 453 individual algorithms.

For example, Baifendian developed a technique that could track individual users' activities in several e-commerce sites, so that for a cold-start user in site A, we could make recommendation according to her records in sites B, C, D, etc [34].

4. *Diversity vs. accuracy*. When the task is to recommend items which are likely to be appreciated by a particular user, it is usually most effective to recommend popular and highly rated items. Such recommendation, however, has very little value for the users because popular objects are easy to find (often they are even hard to avoid) without a recommender system. A good list of recommended items hence should contain also less obvious items that are unlikely to be reached by the users themselves [35]. Approaches to this problem include direct enhancement of the recommendation list's diversity [36–38] and the use of hybrid recommendation methods [39].
5. *Vulnerability to attacks*. Due to their importance in e-commerce applications, recommender systems are likely targets of malicious attacks trying to unjustly promote or inhibit some items [40]. There is a wide scale of tools preventing this kind of behavior, ranging from blocking the malicious evaluations from entering the system to sophisticated resistant recommendation techniques [41]. However, this is not a easy task since the strategies of attackers also get more and more advanced as the developing of preventing tools. As an example, Burke et al. [42] introduced eight attacking strategies, which are further divided into four classes: basic attack, low-acknowledge attack, nuke attack and informed attack.
6. *The value of time*. While real users have interests with widely diverse time scales (for example, short term interests related to a planned trip and long term interests related to the place of living or political preferences), most recommendation algorithms neglect the time stamps of evaluations. It is an ongoing line of research whether and how value of old opinions should decay with time and what are the typical temporary patterns in user evaluations and item relevance [43,44].
7. *Evaluation of recommendations*. While we have plenty of distinct metrics (see Section 3.4), how to choose the ones best corresponding to the given situation and task is still an open question. Comparisons of different recommender algorithms are also problematic because different algorithms may simply solve different tasks. Finally, the overall user experience with a given recommendation system – which includes user's satisfaction with the recommendations and user's trust in the system – is difficult to measure in "offline" evaluation. Empirical user studies thus still represent a welcome source of feedback on recommender systems.
8. *User interface*. It has been shown that to facilitate users' acceptance of recommendations, the recommendations need to be transparent [45,46]: users appreciate when it is clear why a particular item has been recommended to them. Another issue is that since the list of potentially interesting items may be very long, it needs to be presented in a simple way and it should be easy to navigate through it to browse different recommendations which are often obtained by distinct approaches.

Besides the above long-standing challenges, many novel issues appear recently. Thanks to the development of methodology in related branches of science, especially the new tools in network analysis, scientists started to consider the effects of network structure on recommendation and how to make use of known structural features to improve recommendation. For example, Huang et al. [47] analyzed the consumer–product networks and proposed an improved recommendation algorithm preferring edges that enhance the local clustering property, and Sahebi et al. [48] designed an improved algorithm making use of the community structure. Progress and propagation of new techniques also bring new challenges. For example, the GPS equipped mobile phones have become mainstream and the Internet access is ubiquitous, hence the location-based recommendation is now feasible and increasingly significant.[3] Accurate recommendation asks for both the high predictability of human movements [49,50] and quantitative way to define similarities between locations and people [51,52]. Lastly, intelligent recommender systems should take into account the different behavioral patterns of different people. For example, new users tend to visit very popular items and select similar items, while old users usually have more specific interests [53,54], and users behave much differently between low-risk (e.g., collecting bookmarks, downloading music, etc.) and high-risk (e.g., buying a computer, renting a house, etc.) activities [55,56].

3. Definitions of subjects and problems

We briefly review in this chapter basic concepts that are useful in the study of recommender systems.

3.1. Networks

Network analysis is a versatile tool in uncovering the organization principles of many complex systems [5–9]. A network is a set of elements (called *nodes* or *vertices*) with connections (called *edges* or *links*) between them. Many social, biological, technological and information systems can be described as networks with nodes representing individuals or organizations and edges capturing their interactions. The study of networks, referred to as *graph theory* in mathematical literature, has a long history that begins with the classical *Königsberg bridge problem* solved by Euler in 18th century [57]. Mathematically speaking, a network G is an ordered pair of disjoint sets (V, E) where V is the set of nodes and the set of edges, E, is a subset of $V \times V$ [58]. In an *undirected network*, an edge joining nodes x and y is denoted by $x \leftrightarrow y$, and $x \leftrightarrow y$ and $y \leftrightarrow x$ mean exactly the same edge. In a *directed network*, edges are ordered pairs of nodes and an edge from x to y is denoted by $x \to y$.

[3] Websites like Foursquare, Gowalla, Google Latitude, Facebook, Jiapang, and others already provide location-based services and show that many people want to share their location information and get location-based recommendations.

Edges $x \to y$ and $y \to x$ are distinct and may be present simultaneously. Unless stated otherwise, we assume that a network does not contain a *self-loop* (an "edge" joining a node to itself) or a *multi-edge* (several "edges" joining the same pair of nodes). In a *multinetwork* both loops and multi-edges are allowed.

In an undirected network $G(V, E)$, two nodes x and y are said to be adjacent to each other if $x \leftrightarrow y \in E$. The set of nodes adjacent to a node x, the *neighborhood* of x, is denoted by Γ_x. Degree of node x is defined as $k_x = |\Gamma_x|$. The degree distribution, $P(k)$, is defined as the probability that a randomly selected node is of degree k. In a *regular network*, every node has the same degree k_0 and thus $P(k) = \delta_{k,k_0}$. In the classical Erdös–Rényi random network [59] where each pair of nodes is connected by an edge with a given probability p, the degree distribution follows a binomial form [60]

$$P(k) = \binom{N-1}{k} p^k (1-p)^{N-1-k}, \tag{1}$$

where $N = |V|$ is the number of nodes in the network. This distribution has a characterized scale represented by the average degree $\bar{k} = p(N-1)$. At the end of the last century, researchers turned to investigation of large-scale real networks where it turned out that their degree distributions often span several orders of magnitude and approximately follow a power-law form

$$P(k) \sim k^{-\gamma}, \tag{2}$$

with γ being a positive exponent usually lying between 2 and 3 [5]. Such networks are called *scale-free networks* as they lack a characteristic scale of degree and the power-law function $P(k)$ is scale-invariant [61]. Note that detection of power-law distributions in empirical data requires solid statistical tools [62,63]. For a directed network, the out-degree of a node x, denoted by k^{out}, is the number of edges starting at x, and the in-degree k^{in} is the number of edges ending at x. The in- and out-degree distribution of a directed network in general differ from each other.

Generally speaking, a network is said to be *assortative* if its high-degree nodes tend to connect with high-degree nodes and the low-degree nodes tend to connect with low-degree nodes (it is said to be disassortative if the situation is opposite). This degree–degree correlation can be characterized by the average degree of the nearest neighbors [64,65] or a variant of Pearson coefficient called *assortativity coefficient* [66,67]. The assortativity coefficient r lies in the range $-1 \le r \le 1$. If $r > 0$ the network is assortative; if $r < 0$, the network is disassortative. Note that this coefficient is sensitive to degree heterogeneity. For example, r will be negative in a network with very heterogeneous degree distribution (e.g., the Internet) regardless to the network's connecting patterns [68].

The number of edges in a path connecting two nodes is called length of the path, and *distance* between two nodes is defined as the length of the shortest path that connects them. The *diameter* of a network is the maximal distance among all node pairs and the *average distance* is the mean distance averaged over all node pairs as

$$\bar{d} = \frac{1}{N(N-1)} \sum_{x \neq y} d_{xy}, \tag{3}$$

where d_{xy} is the distance between x and y.[4] Many real networks display a so-called *small-world* phenomenon: their average distance does not grow faster than the logarithm of the network size [70,71].

The importance of triadic clustering in social interaction systems has been realized for more than 100 years [72]. In social network analysis [73], this kind of clustering is called *transitivity*, defined as three times the ratio of the total number of triangles in a network to the total number of connected node triples. In 1998, Watts and Strogatz [71] proposed a similar index to quantify the triadic clustering, called *clustering coefficient*. For a given node x, this coefficient is defined as the ratio of the number of existing edges between x's neighbors to the number of neighbor pairs,

$$c_x = \frac{e_x}{\frac{1}{2} k_x (k_x - 1)} \tag{4}$$

where e_x denotes the number of edges between k_x neighbors of node x (this definition is meaningful only if $k_x > 1$). The *network clustering coefficient* is defined as the average of c_x over all x with $k_x > 1$. It is also possible to define the clustering coefficient as the ratio of $3 \times$ number of triangles in the network to the number of connected triples of vertices, which is sometimes referred to as "fraction of transitive triples" [7]. Note that the two definitions can give substantially different results.

Fig. 1 illustrates the above definitions for a simple undirected network. For more information about network measurements, readers are encouraged to refer an extensive review article [74] on characterization of networks.

[4] When no path exists between two nodes, we say that their distance is infinite which makes the average distance automatically infinite too. This problem can be avoided either by excluding such node pairs from averaging or by using the harmonic mean [7,69].

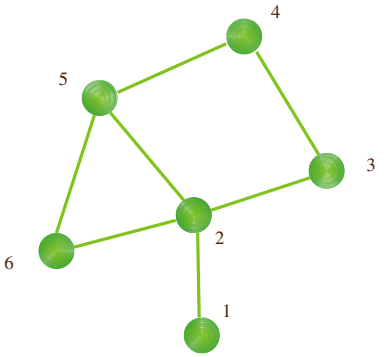

Fig. 1. A simple undirected network with 6 nodes and 7 edges. The node degrees are $k_1 = 1$, $k_2 = k_5 = 3$ and $k_3 = k_4 = k_6 = 2$, corresponding to the distribution $P(1) = 1/6$, $P(2) = 1/2$ and $P(3) = 1/3$. The diameter and average distance of this network are $d_{\max} = 3$ and $\bar{d} = 1.6$, respectively. The clustering coefficients are $c_2 = \frac{1}{6}$, $c_3 = c_4 = 0$, $c_5 = \frac{1}{3}$ and $c_6 = 1$, and the average clustering coefficient is $C = 0.3$.

3.2. Bipartite networks and hypergraphs

A network $G(V, E)$ is a *bipartite network* if there exists a partition (V_1, V_2) such that $V_1 \cup V_2 = V$, $V_1 \cap V_2 = \emptyset$, and every edge connects a node of V_1 and a node of V_2. Many real systems are naturally modeled as bipartite networks: the metabolic network [75] consists of chemical substances and chemical reactions, the collaboration network [76] consists of acts and actors, the Internet telephone network consists of personal computers and phone numbers [77], etc. We focus on a particular class of bipartite networks, called *web-based user–object networks* [53], which represent interactions between users and objects in online service sites, such as collections of bookmarks in *delicious.com* and purchases of books in *amazon.com*. As we shall see later, these networks describe the fundamental structure of recommender systems. Web-based user–object networks are specific by their gradual evolution where both nodes and links are added gradually. By contrast, this cannot happen in, for example, act–actor networks (e.g., one cannot add authors to a scientific paper after its publication).

Most web-based user–object networks share some structural properties. Their object-degree distributions obey a power-law-like form $P(k) \sim k^{-\gamma}$, with $\gamma \approx 1.6$ for the *Internet Movie Database* (IMDb) [78], $\gamma \approx 1.8$ for the music-sharing site *audioscrobbler.com* [79], $\gamma \approx 2.3$ for the e-commerce site *amazon.com* [53], and $\gamma \approx 2.5$ for the bookmark-sharing site *delicious.com* [53]. The form of the user-degree distribution is usually between an exponential and a power law [53], and can be well fitted by the Weibull distribution distribution [27] (also called the stretched exponential distribution in the literature [80])

$$P(k) \sim k^{\mu-1} \exp\left[-(k/k_0)^{\mu}\right] \tag{5}$$

where k_0 is a constant and μ is the stretching exponent. Connections between users and objects exhibit a disassortative mixing pattern [53,78].

A straightforward extension of the definition of bipartite network is the so-called *multipartite network*. For an r-partite network $G(V, E)$, there is an r-partition $V_1, V_2, \ldots, V_r$ such that $V = V_1 \cup V_2 \cup \cdots \cup V_r$, $V_i \cap V_j = \emptyset$ whenever $i \neq j$, and no edge joins two nodes in the same set V_i for all $1 \leq i \leq r$. The tripartite network representation has found its application in *collaborative tagging systems* (also called *folksonomies* in the literature) [81–84], where users assign tags to online resources, such as photographs in *flickr.com*, references in *CiteULike.com* and bookmarks in *delicious.com*.

Note that some information is lost in the tripartite representation. For example, given an edge connecting a resource and a tag, we do not know which user (or users) contributed to this edge. To resolve this, *hypergraph* [85] can be used to give an exact representation of the full structure of a collaborative tagging system. In a hypergraph $H(V, E)$, the *hyperedge* set E is a subset of the power set of V, that is the set of all subsets of V. Link e can therefore connect multiple nodes. Analogously to ordinary networks, node degree in a hypergraph is defined as the number of hyperedges adjacent to a node and the distance between two nodes is defined as the minimal number of hyperedges connecting these nodes. The clustering coefficient [82, 86] and community structure [86,87] can also be defined and quantified following the definitions in ordinary networks. Notice that there is a one-to-one correspondence between a hypergraph and a bipartite network. Given a hypergraph $H(V, E)$, the corresponding bipartite network $G(V', E')$ contains two node sets, as $V' = V \cup E$, and $x \in V$ is connected with $Y \in E$ if and only if $x \in Y$ (see Fig. 2 for an illustration).

Hypergraph representation has already found applications in ferromagnetic dynamics [88,89], population stratification [90], cellular networks [91], academic team formation [92], and many other areas. Here we are concerned more about the hypergraph representation of collaborative tagging systems [86,93,94] where each hyperedge joins three nodes (represented by a triangle-like in Fig. 3), user u, resource r and tag t, indicating that u has given t to r. A resource can be collected by many users and given several tags by a user, and a tag can be associated with many resources, resulting in small-world hypergraphs [86,94] (Fig. 3 shows both the basic unit and extensive description). Moreover, hypergraphs for collaborative

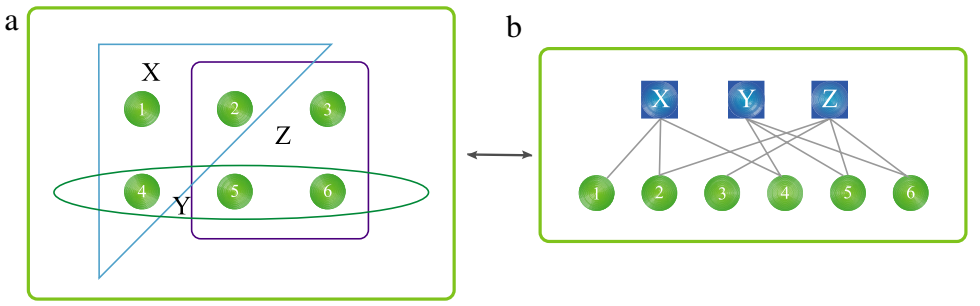

Fig. 2. An illustration of the one-to-one correspondence between a hypergraph (a) and a bipartite network (b). There are three hyperedges, $X = \{1, 2, 4\}$, $Y = \{4, 5, 6\}$ and $Z = \{2, 3, 5, 6\}$.

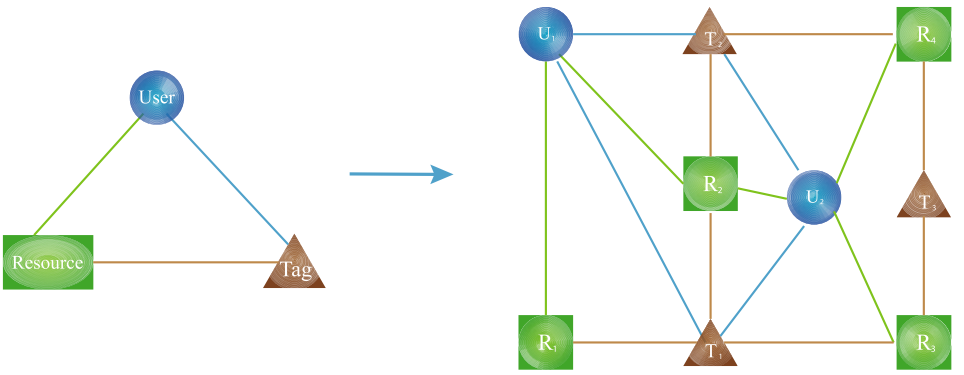

Fig. 3. A hypergraph illustration of collaborative tagging networks. (left) A triangle-like hyperedge [93], which contains three types of vertices, depicted by one blue circle, one green rectangle and one brown triangle which respectively represent a user, a resource and a tag. (right) A descriptive hypergraph consists of two users, four resources and three tags. Take user U_2 and resource R_1 for example, the measurements are denoted as: (i) U_2 has participated in six hyperedges, which means its hyperdegree is 6; (ii) U_2 has directly connected to three resources and three tags. As defined by Eq. (6), it suggests it possibly has $3 \times 3 = 9$ hyperedges in maximal. Thus its clustering coefficient equals $6/9 \approx 0.667$, where 6 is its hyperdegree; Comparatively, as defined by Eq. (7), its clustering coefficient is $D_h(U_2) = \frac{12-6}{12-4} = 0.75$; (iii) the shortest path from U_2 to R_1 is U_2–T_1–R_1, which indicates the distance between U_2 and R_1 is 2. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

tagging systems have been shown to be highly clustered, with heavy-tailed degree distribution[5] and of community structure [86,94]. A model for evolving hypergraphs can be found in [94].

Generally, to evaluate a hypergraph from the perspective of complexity science, the following quantities (Fig. 3 gives a detailed description of these quantities) can be applied:

(i) *hyperdegree*: The degree of a node in a hypergraph can be naturally defined as the number of hyperedges adjacent to it.

(ii) *hyperdegree distribution*: defined as the proportion that each hyperdegree occupies, where hyperdegree is defined as the number of hyperedges that a regular node participates in.

(iii) *clustering coefficients*: defined as the proportion of real number of hyperedges to all the possible number of hyperedges that a regular node could have [82]. e.g., the clustering coefficient for a user, C_u, is defined as

$$C_u = \frac{k_u}{R_u T_u}, \tag{6}$$

where k_u is the hyperdegree of user u, R_u is the number of resources that u collects and T_u is the number of tags that u possesses. The above definition measures the fraction of possible pairs present in the neighborhood of u. A larger C_u indicates that u has more similar topic of resources, which might also show that u has more concentrated on personalized or special topics, while smaller C_u might suggest that s/he has more diverse interests. Similar definitions can also be defined for measuring the clustering coefficient of resources and tags.

An alternative metric, named hyperedge density, is proposed by Zlatić et al. [86]. Taking a user node u again as an example, they define the coordination number of u as $z(u) = R_u + T_u$. Given $k(u)$, the maximal coordination number is

[5] The degrees of users, resources and tags are usually investigated separately. For *flickr.com* and *CiteULike.com*, the user and tag degree distributions are power-law-like, while the resource degree distributions are much narrower because in *flickr.com*, a photograph is only collected by a single user and in *CiteULike.com*, a reference is rarely collected by many users [86]. By contrast, in *delicious.com*, a popular bookmark can be collected by thousands of users and thus the resource degree distribution is of a power-law kind [94].

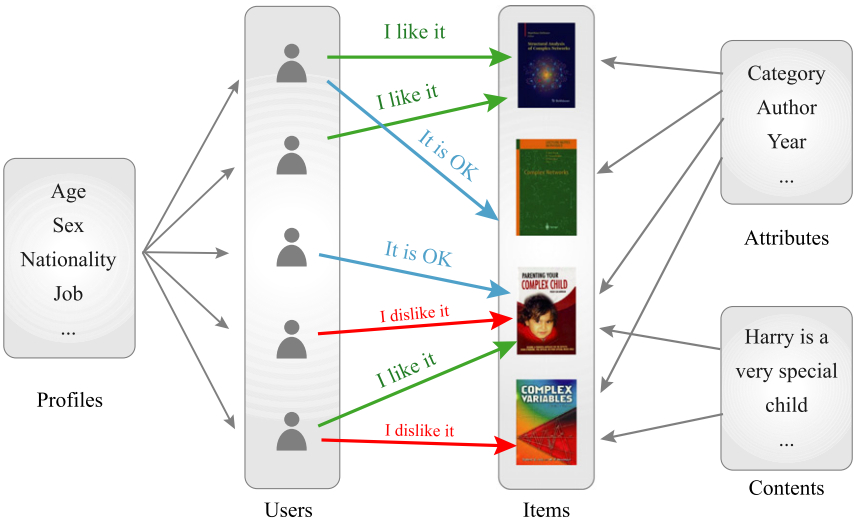

Fig. 4. Illustration of a recommender system consisted of five users and four books. The basic information contained by every recommender system is the relations between users and objects that can be represented by a bipartite graph. This illustration also exhibits some additional information frequently exploited in the design of recommendation algorithms, including user profiles, object attributes and object content.

$z_{\max}(u) = 2k(u)$, while the minimal coordination number is $z_{\min}(u) = 2n$ for $n(n-1) < k(u) \leq n^2$ and $z_{\min}(u) = 2n+1$ for $n^2 < k(u) \leq n(n+1)$, with n some integer. Obviously, a local tree structure leads to maximal coordination number, while the maximum overlap corresponds to the minimal coordination number. Therefore, they define the hyperedge density as [86]:

$$D_h(u) = \frac{z_{\max}(u) - z(u)}{z_{\max}(u) - z_{\min}(u)}, \quad 0 \leq D_h(u) \leq 1. \tag{7}$$

The definition of hyperedge density for resources and tags is similar. Empirical analysis indicates a high clustering behavior under both metrics [82,86]. The study of hypergraph for the collaborative tagging networks has just been unfolding, and how to properly quantify the clustering behavior, the correlations and similarities between nodes, and the community structure is still an open problem.

(iv) *average distance*: defined as the average shortest path length between two random nodes in the whole network.

3.3. Recommender systems

A recommender system uses the input data to predict potential further likes and interests of its users. Users' past evaluations are typically an important part of the input data. Let M be the number of users and let N be the number of all objects that can be evaluated and recommended. Note that *object* is simply as a generic term which can represent books, movies, or any other kind of consumed content. To stay in line with standard terminology, we sometimes use *item* which has the same meaning. To make the notation more clear, we restrict to Latin indices i and j when enumerating the users and to Greek indices α and β when enumerating the objects. Evaluation/rating of object α by user i is denoted as $r_{i\alpha}$. This evaluation is often numerical in an integer rating scale (think of Amazon's five stars)—in this case we speak of explicit ratings. Note that the common case of binary ratings (like/dislike or good/bad) also belongs to this category. When objects are only collected (as in bookmark sharing systems) or simply consumed (as in online newspaper or magazine without rating systems) or when "like" is the only possible expression (as on Facebook), we are left with unary ratings. In this case, $r_{i\alpha} = 1$ represents a collected/consumed/liked object and $r_{i\alpha} = 0$ represents a non-existing evaluation (see Fig. 4). Inferring users' confidence levels of ratings is not a trivial task, especially from the binary or unary ratings. Accessorial information about users' behavior may be helpful, for example, the users' confidence levels can be estimated by their watching time of television shows and with the help of this information, the quality of recommendation can be improved [95]. Even if we have explicit ratings, it does not mean we know how and why people vote with these ratings—do they have standards of numerical ratings or they just use ratings to present orders? Recent evidence [96] to some extent supports the latter ansatz.

The goal of a recommender system is to deliver lists of personalized "recommended" objects to its users. To this end, evaluations can be predicted or, alternatively, recommendation scores can be assigned to objects yet unknown to a given user. Objects with the highest predicted ratings or the highest recommendation scores then constitute the recommendation list that is presented to the target user. There is an extensive set of performance metrics that can be used to evaluate the resulting recommendation lists (see Section 3.4). The usual classifications of recommender systems is as follows [15]:

1. *Content-based recommendations:* Recommended objects are those with content similar to the content of previously preferred objects of a target user. We present them in Section 4.2.3.

Table 2
Recommendation process in a nutshell: to estimate the potential favorable opinion of Carol about Casablanca, one can use the similarity of her with those of Alice. Alternatively, one can note that ratings of Titanic and Casablanca follow a similar pattern, suggesting that people who liked the former might also like the latter.

	Alice	Bob	Carol
Titanic	5	1	5
2001: A Space Odyssey	1	5	2
Casablanca	4	2	?

2. *Collaborative recommendations:* Recommended objects are selected on the basis of past evaluations of a large group of users. An example is given in Table 2. They can be divided into:
   (a) *Memory-based collaborative filtering:* Recommended objects are those that were preferred by users who share similar preferences as the target user, or, those that are similar to the other objects preferred by the target user. We present them in Sections 4 (Standard similarity-based methods) and 7 (methods employing social filtering).
   (b) *Model-based collaborative filtering:* Recommended objects are selected on models that are trained to identify patterns in the input data. We present them in Sections 5 (dimensionality reduction methods) and 6 (diffusion-based methods).
3. *Hybrid approaches:* These methods combine collaborative with content-based methods or with different variants of other collaborative methods. We present them in Section 8.4.

3.4. Evaluation metrics for recommendation

Given a target user i, a recommender system will sort all i's uncollected objects and recommend the top-ranked objects. To evaluate recommendation algorithms, the data is usually divided into two parts: The training set E^T and the probe set E^P. The training set is treated as known information, while no information from the probe set is allowed to be used for recommendation. In this section we briefly review basic metrics that are used to measure the quality of recommendations. How to choose a particular metric (or metrics) to evaluate recommendation performance depends on the goals that the system is supposed to fulfill. Of course, the ultimate evaluation of any recommender system is given by the judgment of its users.

3.4.1. Accuracy metrics

Rating accuracy metrics. The main purpose of recommender systems is to predict users' future likes and interests. A multitude of metrics exist to measure various aspects of recommendation performance. Two notable metrics, *Mean Absolute Error* (MAE) and *Root Mean Squared Error* (RMSE), are used to measure the closeness of predicted ratings to the true ratings. If $r_{i\alpha}$ is the true rating on object α by user i, $\tilde{r}_{i\alpha}$ is the predicted rating and E^P is the set of hidden user–object ratings, MAE and RMSE are defined as

$$\mathrm{MAE} = \frac{1}{|E^P|} \sum_{(i,\alpha)\in E^P} |r_{i\alpha} - \tilde{r}_{i\alpha}|, \tag{8}$$

$$\mathrm{RMSE} = \left(\frac{1}{|E^P|} \sum_{(i,\alpha)\in E^P} (r_{i\alpha} - \tilde{r}_{i\alpha})^2 \right)^{1/2}. \tag{9}$$

Lower MAE and RMSE correspond to higher prediction accuracy. Since RMSE squares the error before summing it, it tends to penalize large errors more heavily. As these metrics treat all ratings equally no matter what their positions are in the recommendation list, they are not optimal for some common tasks such as finding a small number of objects that are likely to be appreciated by a given user (*Finding Good Objects*). Yet, due to their simplicity, RMSE and MAE are widely used in the evaluation of recommender systems.

Rating and ranking correlations. Another way to evaluate the prediction accuracy is to calculate the correlation between the predicted and the true ratings. There are three well-known correlation measures, namely the Pearson product-moment correlation [97], the Spearman [98] correlation and Kendall's Tau [99]. The Pearson correlation measures the extent to which a linear relationship is present between the two sets of ratings. It is defined as

$$PCC = \frac{\sum_{\alpha} (\tilde{r}_\alpha - \bar{\tilde{r}})(r_\alpha - \bar{r})}{\sqrt{\sum_{\alpha} (\tilde{r}_\alpha - \bar{\tilde{r}})^2} \sqrt{\sum_{\alpha} (r_\alpha - \bar{r})^2}}, \tag{10}$$

where r_α and $\tilde{r}_\alpha$ are the true and predicted ratings, respectively. The Spearman correlation coefficient ρ is defined in the same manner as the Pearson correlation, except that r_α and $\tilde{r}_\alpha$ are replaced by the ranks of the respective objects. Similarly

to the Spearman correlation, Kendall's Tau also measures the extent to which the two rankings agree on the exact values of ratings. It is defined as $\tau = (C - D)/(C + D)$ where C is the number of concordant pairs—pairs of objects that the system predicts in the correct ranked order and D is the number of discordant pairs—pairs that the system predicts in the wrong order. $\tau = 1$ when the true and predicted ranking are identical and $\tau = -1$ when they are exactly opposite. For the case when objects with equal true or predicted ratings exist, a variation of Kendall's Tau was proposed in [13]

$$\tau \approx \frac{C - D}{\sqrt{(C + D + S_T)(C + D + S_P)}}, \tag{11}$$

where S_T is the number of object pairs for which the true ratings are the same, and S_P is the number of object pairs for which the predicted ratings are the same. Kendall's Tau metric applies equal weight to any interchange of successively ranked objects, no matter where it occurs. However, interchanges at different places, for example between top 1 and 2, and between 100 and 101, may have different impacts. Thus a possible improved metric could give more weight to object pairs at the top of the true ranking.

Similar to Kendall's Tau, the normalized distance-based performance measure (NDPM) was originally proposed by Yao [100] to compare two different weakly ordered rankings. It is based on counting the number of contradictory pairs C^- (for which the two rankings disagree) and compatible pairs C^u (for which one ranking reports a tie while the other reports strict preference of one object over the other). Denoting the total number of strict preference relationships in the true ranking as C, NDPM is defined as

$$NDPM = \frac{2C^- + C^u}{2C}. \tag{12}$$

Since this metric does not punish the situation where the true ranks are tied, it is more appropriate than correlation metrics for domains where users are interested in objects that are good-enough.

Classification accuracy metrics. Classification metrics are appropriate for tasks such as "Finding Good Objects", especially when only implicit ratings are available (*i.e.*, we know which objects were favored by a user but not how much they were favored). When a ranked list of objects is given, the threshold for recommendations is ambiguous or variable. To evaluate this kind of systems, one popular metric is AUC (Area Under ROC Curve), where ROC stands for the receiver operating characteristic [101] (for how to draw a ROC curve see [13]). AUC attempts to measure how a recommender system can successfully distinguish the relevant objects (those appreciated by a user) from the irrelevant objects (all the others). The simplest way to calculate AUC is by comparing the probability that the relevant objects will be recommended with that of the irrelevant objects. For n independent comparisons (each comparison refers to choosing one relevant and one irrelevant object), if there are n' times when the relevant object has higher score than the irrelevant and n'' times when the scores are equal, then according to [102]

$$\text{AUC} = \frac{n' + 0.5n''}{n}. \tag{13}$$

Clearly, if all relevant objects have higher score than irrelevant objects, AUC $= 1$ which means a perfect recommendation list. For a randomly ranked recommendation list, AUC $= 0.5$. Therefore, the degree of which AUC exceeds 0.5 indicates the ability of a recommendation algorithm to identify relevant objects. Similar to AUC is a so-called *Ranking Score* proposed in [103]. For a given user, we measure the relative ranking of a relevant object in this user's recommendation list: when there are o objects to be recommended, a relevant object with ranking r has the relative ranking r/o. By averaging over all users and their relevant objects, we obtain the mean ranking score RS—the smaller the ranking score, the higher the algorithm's accuracy, and vice versa.

Since real users are usually concerned only with the top part of the recommendation list, a more practical approach is to consider the number of a user's relevant objects ranked in the top-L places. Precision and recall are the most popular metrics based on this. For a target user i, precision and recall of recommendation, $P_i(L)$ and $R_i(L)$, are defined as

$$P_i(L) = \frac{d_i(L)}{L}, \qquad R_i(L) = \frac{d_i(L)}{D_i} \tag{14}$$

where $d_i(L)$ indicates the number of relevant objects (objects collected by i that are present in the probe set) in the top-L places of the recommendation list, and D_i is the total number of i's relevant objects. Averaging the individual precision and recall over all users with at least one relevant object, we obtain the mean precision and recall, $P(L)$ and $R(L)$, respectively. These values can be compared with precision and recall resulting from random recommendation, leading to precision and recall enhancements as defined in [39]

$$e_P(L) = P(L)\,\frac{MN}{D}, \qquad e_R(L) = R(L)\,\frac{N}{L}, \tag{15}$$

where M and N are the number of users and objects, respectively, and D is the total number of relevant objects. While precision usually decreases with L, recall always grows with L. One may combine them into a less L-dependent metric [104,105]

$$F_1(L) = \frac{2PR}{P+R} \tag{16}$$

which is called F_1-score. Many other measurements which combine precision and recall are used to evaluate the effectiveness of information retrieval, but rarely applied to evaluate recommendation algorithms: *Average Precision*, *Precision-at-Depth*, *R-Precision*, *Reciprocal Rank* [106], *Binary Preference Measure* [107]. A detailed introduction and discussion of each combination index can be found in [108].

3.4.2. Rank-weighted indexes

Since users have limited patience on inspecting individual objects in the recommended lists, user satisfaction is best measured by taking into account the position of each relevant object and assign weights to them accordingly. Here we introduce three representative indexes that follow this approach. For a detailed discussion of their strengths and weaknesses see [13].

Half-life utility. The *half-life utility* metric attempts to evaluate the utility of a recommendation list to a user. It is based on the assumption that the likelihood that a user examines a recommended object decays exponentially with the object's ranking. The expected utility of recommendations given to user i hence becomes [109]

$$HL_i = \sum_{\alpha=1}^{N} \frac{\max(r_{i\alpha} - d, 0)}{2^{(o_{i\alpha}-1)/(h-1)}}, \tag{17}$$

where objects are sorted by their recommendation score $\tilde{r}_{i\alpha}$ in descending order, $o_{i\alpha}$ represents the predicted ranking of object α in the recommendation list of user i, d is the default rating (for example, the average rating), and the "half-life" h is the rank of the object on the list for which there is a 50% chance that the user will eventually examine it. This utility can be further normalized by the maximum utility (which is achieved when the user's all known ratings appear at the top of the recommendation list). When HL_i is averaged over all users, we obtain an overall utility of the whole system.

Discounted cumulative gain. For a recommendation list of length L, DCG is defined as [110]

$$DCG(b) = \sum_{n=1}^{b} r_n + \sum_{n=b+1}^{L} \frac{r_n}{\log_b n}, \tag{18}$$

where r_n indicates the relevance of the n-th ranked object ($r_n = 1$ for a relevant object and zero otherwise) and b is a persistence parameter which was suggested to be 2. The intention of DCG is that highly ranked relevant objects give more satisfaction and utility than badly ranked ones.

Rank-biased precision. This metric assumes that users always check the first object and progress from one object to the next one with certain (persistence) probability p (with a complementary probability $1 - p$, the examination of the recommendation list ends). For a list of length L, the rank-biased precision metric is defined as [108]

$$RBP = (1-p)\sum_{n=1}^{L} r_n p^{n-1}, \tag{19}$$

where r_n is the same as in DCG. RBP is similar to DCG, the difference is that RBP discounts relevance via a geometric sequence, while DCG does so using a log-harmonic form.

3.4.3. Diversity and novelty

Even a successfully recommended relevant object has little value to a user when it is notorious. To complement the above accuracy-probing metrics, several diversity- and novelty-probing metrics have been proposed recently [35,39,111] and we introduce them here.

Diversity. Diversity in recommender systems refers to how different the recommended objects are with respect to each other. There are two levels to interpret diversity: one refers to the ability of an algorithm to return different results to different users—we call it *Inter-user diversity* (*i.e.*, the diversity between recommendation lists). The other one measures the extent to which an algorithm can provide diverse objects to each individual user—we call it *Intra-user diversity* (*i.e.*, the diversity within a recommendation list). Inter-user diversity [112] is defined by considering the variety of users' recommendation lists. Given users i and j, the difference between the top L places of their recommendation lists can be measured by the Hamming distance

$$H_{ij}(L) = 1 - \frac{Q_{ij}(L)}{L}, \tag{20}$$

where $Q_{ij}(L)$ is the number of common objects in the top-L places of the lists of users i and j. If the lists are identical, $H_{ij}(L) = 0$, while if their lists are completely different, $H_{ij}(L) = 1$. Averaging $H_{ij}(L)$ over all user pairs, we obtain the mean Hamming distance $H(L)$. The greater its value, the more diverse (more personalized) recommendation is given to the users.

Denoting the recommended objects for user i as $\{o_1, o_2, \ldots, o_L\}$, similarity of these objects $s(o_\alpha, o_\beta)$ can be used to measure the intra-user diversity (this similarity can be obtained either directly from the input ratings or from object metadata) [113]. The average similarity of objects recommended to user i,

$$I_i(L) = \frac{1}{L(L-1)} \sum_{\alpha \neq \beta} s(o_\alpha, o_\beta), \tag{21}$$

can be further averaged over all users to obtain the mean intra-similarity of the recommendation lists, $I(L)$. The lower is this quantity, the more diverse objects are recommended to the users. Notably, intra-list diversity can be used to enhance improve recommendation lists by avoiding recommendation of excessively similar objects [37]. The rank-sensitive version can be obtained by introducing a discount function of the object's rank in recommendation list [111].

Novelty and surprisal. The novelty in recommender systems refers to how different the recommended objects are with respect to what the users have already seen before. The simplest way to quantify the ability of an algorithm to generate novel and unexpected results is to measure the average popularity of the recommended objects

$$N(L) = \frac{1}{ML} \sum_{i=1}^{M} \sum_{\alpha \in O_R^i} k_\alpha, \tag{22}$$

where O_R^i is the recommendation list of user i and k_α denotes the degree of object α (*i.e.*, the popularity of object α). Lower popularity indicates higher novelty of the results. Another possibility to measure the unexpectedness is using the self-information (surprisal) [114] of recommended objects. Given an object α, the chance that a randomly-selected user has collected it is k_α/M and thus its self-information is

$$U_\alpha = \log_2(M/k_\alpha). \tag{23}$$

A user-relative novelty variant can be defined by restricting the observations to the target user, namely calculating the mean self-information of target user's top-L objects. Averaging over all users we obtain the mean top-L surprisal $U(L)$. With a similar resulting formula, a discovery-based novelty was proposed in [111] by considering the probability that an object is known or familiar to a random user.

3.4.4. Coverage

Coverage measures the percentage of objects that an algorithm is able to recommend to users in the system. Denoting the total number of distinct objects in top L places of all recommendation lists as N_d, the L-dependent coverage is defined as

$$COV(L) = N_d/N. \tag{24}$$

Low coverage indicates that the algorithm can access and recommend only a small number of distinct objects (usually the most popular ones) which often results in little diverse recommendations. On the contrary, algorithms with high coverage are more likely to provide diverse recommendations [115]. From this viewpoint, coverage can be also considered as a diversity metric. In addition, coverage is helpful to better evaluate results of accuracy metrics [116]: recommending popular objects is likely to be of high accuracy but of low coverage. A good recommendation method is expected to be of both high accuracy and coverage.

The choice of a particular metric (or metrics) to evaluate a recommender system depends on the goals that the system is supposed to fulfill. In practice, one may specify different goals for new and experienced users which further complicates the evaluation process. For a better overview, Table 3 summarizes the described metrics for evaluation of recommender systems.

4. Similarity-based methods

Similarity-based methods represent one of the most successful approaches to recommendation. They have been studied extensively and found various applications in e-commerce [117,118]. This class of algorithms can be further divided into methods employing user and item similarity, respectively. The basic assumption of a method based on user similarity is that people who agree in their past evaluations tend to agree again in their future evaluations. Thus, for a target user, the potential evaluation of an object is estimated according to the ratings from users (''taste mates'') who are similar to the target user (see Fig. 5 for a schematic illustration). Different from user similarity, an algorithm based on item similarity recommends a user the objects that are similar to what this user has collected before. Note that, sometimes the opinions from dissimilar users [119] or the negative ratings [120,121] can play a significant (even positive) role in determining the recommendation, especially when the data set is very sparse and thus the information about relevance is more important than that about correlation [122]. For additional information see the recent review articles [123,124], and [125] is a nice survey that contains a number of similarity indices.

Table 3
Summary of the presented recommendation metrics. The third column represents the preference of the metric (*e.g.*, smaller MAE means higher rating accuracy). The fourth column describes the scope of the metric. The last two columns show whether the metric is obtained from a ranking and whether it depends on the length of the recommendation list L.

Name	Symbol	Preference	Scope	Rank	L
MAE	*MAE*	Small	Rating accuracy	No	No
RMSE	*RMSE*	Small	Rating accuracy	No	No
Pearson	*PCC*	Large	Rating correlation	No	No
Spearman	ρ	Large	Rating correlation	Yes	No
Kendall's Tau	τ	Large	Rating correlation	Yes	No
NDPM	*NDPM*	Small	Ranking correlation	Yes	No
Precision	$P(L)$	Large	Classification accuracy	No	Yes
Recall	$R(L)$	Large	Classification accuracy	No	Yes
F_1-score	$F_1(L)$	Large	Classification accuracy	No	Yes
AUC	*AUC*	Large	Classification accuracy	No	No
Ranking score	*RS*	Small	Ranking accuracy	Yes	No
Half-life utility	$HL(L)$	Large	Satisfaction	Yes	Yes
Discounted Cumulative Gain	$DCG(b, L)$	Large	Satisfaction and precision	Yes	Yes
Rank-biased Precision	$RBP(p, L)$	Large	Satisfaction and precision	Yes	Yes
Hamming distance	$H(L)$	Large	Inter-diversity	No	Yes
Intra-similarity	$I(L)$	Small	Intra-diversity	No	Yes
Popularity	$N(L)$	Small	Surprisal and novelty	No	Yes
Self-information	$U(L)$	Large	Unexpectedness	No	Yes
Coverage	$COV(L)$	Large	Coverage and diversity	No	Yes

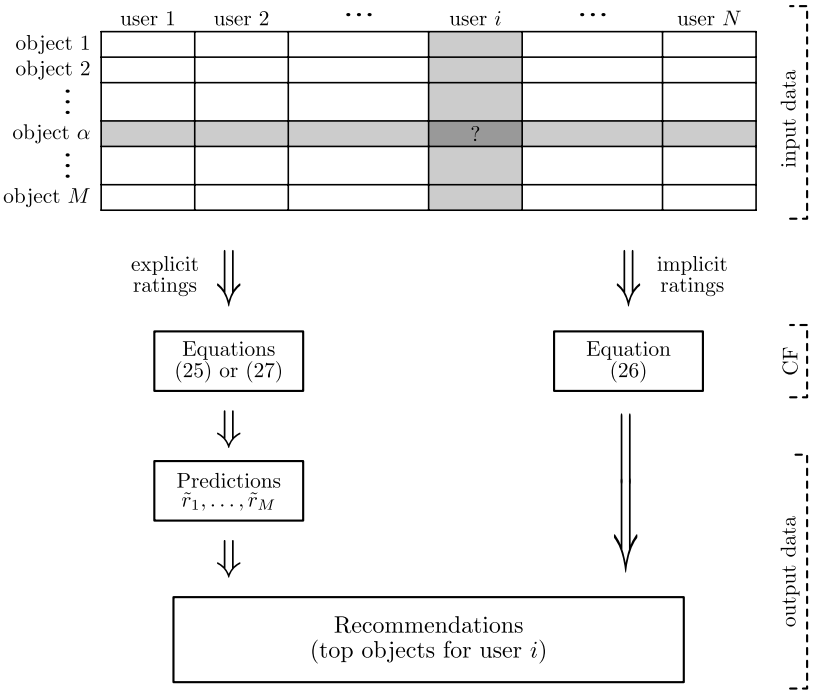

Fig. 5. A schematic representation of the collaborative-filtering (CF) recommendation method: rating prediction for a given user–object pair is based on the user's and object's past ratings.

4.1. Algorithms

Here we briefly introduce the conventional similarity-based algorithms which are often referred to as memory-based collaborative filtering techniques. The term "collaborative filtering" was introduced by creators of the first commercial recommender system, Tapestry [126], derives from the fact that it requires collaboration of multiple agents who share their data to obtain better recommendation. In the following sections, we describe basic algorithms as well as main approaches to the computation of similarity which is a critical component of the recommendation process.

4.1.1. User similarity

The goal is to make automated prediction of user preferences by collecting evaluation data from many other users, especially those whose evaluations are similar to evaluations from the target user. Denote the rating from user u on object α as $r_{u\alpha}$ and let Γ_u be the set of objects that user u has evaluated. The average rating given by u is $\bar{r}_u = \frac{1}{|\Gamma_u|}\sum_{\alpha\in\Gamma_u} r_{u\alpha}$. According to the standard collaborative filtering, the predicted rating of user u on object α is

$$\tilde{r}_{u\alpha} = \bar{r}_u + \kappa \sum_{v\in\hat{U}_u} s_{uv}(r_{v\alpha} - \bar{r}_v) \tag{25}$$

where $\hat{U}_u$ denotes the set of users that are most similar to user u, s_{uv} denotes the similarity between user u and user v and $\kappa = \frac{1}{\sum_v |s_{uv}|}$ is a normalization factor. If instead of explicit ratings, only the sets of objects collected by individual users are known (implicit ratings), we aim at predicting the objects which are most likely to be collected by a user in the future. According to [119], Eq. (25) should be replaced with

$$p_{u\alpha} = \sum_{v\in\hat{U}_u} s_{uv} a_{v\alpha} \tag{26}$$

where $p_{u\alpha}$ is the recommendation score of object α for user u and $a_{v\alpha}$ is an element of the adjacency matrix of the user–object bipartite network ($a_{v\alpha} = 1$ if user v has collected object α and $a_{v\alpha} = 0$ otherwise).

As has already been made explicit by Eqs. (25) and (26), only users most similar to given user u are usually considered. To obtain $\hat{U}_u$, two neighborhood selection strategies are usually applied: (i) correlation threshold [127] is based on selecting all users v whose similarity s_{uv} surpasses a given threshold, (ii) maximum number of neighbors [128] consists of selecting those k users that are most similar to u (here k is a parameter of the algorithm). Restricting computation to the most similar users is not only computationally advantageous but in general it leads to superior results [129].

4.1.2. Item similarity

In this case, item–item similarity $s_{\alpha\beta}$ is employed instead of user–user similarity s_{uv}. The simplest way is to estimate unknown ratings using the weighted average [130]

$$\tilde{r}_{u\alpha} = \frac{\sum_{\beta\in\Gamma_u} s_{\alpha\beta} r_{u\beta}}{\sum_{\beta\in\Gamma_u} |s_{\alpha\beta}|} \tag{27}$$

where Γ_u is the set of items evaluated by user u. Techniques limiting the computation of $\tilde{r}_{u\alpha}$ to items that are most similar to α can be applied similarly as described above for user similarity. One of the advantages of this approach is that similarity between items tends to be more static than similarity between users, allowing its values and neighborhoods to be computed offline (*i.e.*, before recommendation for a particular user is requested—this allows to shorten the time needed to obtain the recommendation). Hybrid collaborative filtering algorithms combining user-, item- or attribute-based similarity were proposed [131,132]. Their results show that this approach not only improves the prediction accuracy but it is also more robust to data sparsity.

4.1.3. Slope One predictor

Slope One predictor with the form $f(x) = x + b$, where b is a constant and x is a variable representing the rating values, is the simplest form of item-based collaborative filtering based on ratings [133]. It subtracts the average ratings of two items to measure how much more, on average, one item is liked than another. This difference is used to predict another user's rating of one of these two items, given his rating of the other. For example, consider a case where user i gave score 1 to item α and score 1.5 to item β while user j gave score 2 to item α. Slope One then predicts that user j will rate item β with $2 + (1.5 - 1) = 2.5$ (see Fig. 6 for an illustration).

The Slope One scheme takes into account both information from other users who rated the same item and from the other items rated by the same user. In particular, only ratings by users who have rated some common items with the target user and only ratings of items that the target user has also rated are involved in the prediction process. Denoting the set of users who rated items α and β as $S(\alpha, \beta)$, the average deviation of item β with respect to item α is defined as

$$\mathrm{dev}_{\beta\alpha} = \frac{\sum_{i\in S(\alpha,\beta)} r_{i\beta} - r_{i\alpha}}{|S(\alpha,\beta)|}. \tag{28}$$

Given a known rating $r_{u\alpha}$, Slope One predicts u's rating on item β as $r_{u\alpha} + \mathrm{dev}_{\beta\alpha}$. By varying α in Eq. (28), we obtain different predictions. A reasonable overall predictor is their average value

$$\tilde{r}_{u\alpha} = \frac{1}{|R(u,\alpha)|}\sum_{\alpha\in R(u,\alpha)} (r_{u\beta} + \mathrm{dev}_{\alpha\beta}), \tag{29}$$

	object α		object β
user i	1		1.5
user j	2		x

Fig. 6. In the depicted case, the Slope One prediction would be $x = 2 + (1.5 - 1) = 2.5$.

where $R(u, \alpha)$ is the set of items that have been both rated by u and co-rated with item β. Note that predictions obtained from different items α have equal weight no matter how many users have co-rated α with β. To take into account the fact that the credibility of $\mathrm{dev}_{\alpha\beta}$ depends crucially on $|S(\alpha, \beta)|$ (the larger the overlap, the more trustful the value), one can introduce a Weighted Slope One prediction as

$$\tilde{r}^{w}_{u\alpha} = \frac{\sum_{\alpha} |S(\alpha, \beta)|(r_{u\beta} + \mathrm{dev}_{\alpha\beta})}{\sum_{\alpha} |S(\alpha, \beta)|}. \tag{30}$$

Another improvement of the basic Slope One algorithm is based on dividing the set of all items into items liked and disliked by a given user (a straightforward criterion to identify liked and disliked items is to check whether their rating were higher or lower than the average rating awarded by the given user). From these liked and disliked items, two separate predictions are then derived which are combined into one prediction at the very end. Denote by $S^{+1}(\alpha, \beta)$ and $S^{-1}(\alpha, \beta)$ the sets of users who like and dislike, respectively, both α and β. The deviations for liked and disliked items are

$$\mathrm{dev}^{+1}_{\beta\alpha} = \frac{1}{|S^{+1}(\beta, \alpha)|} \sum_{i \in S^{+1}(\beta,\alpha)} (r_{i\beta} - r_{i\alpha}), \qquad \mathrm{dev}^{-1}_{\beta\alpha} = \frac{1}{|S^{-1}(\beta, \alpha)|} \sum_{i \in S^{-1}(\beta,\alpha)} (r_{i\beta} - r_{i\alpha}). \tag{31}$$

The prediction for the rating of item β based on the rating of item α is either $r_{j\alpha} + \mathrm{dev}^{+1}_{\beta\alpha}$ or $r_{j\alpha} + \mathrm{dev}^{-1}_{\beta\alpha}$ depending on whether the target user j likes or dislikes item α respectively. The Bi-Polar Slope One is thus given by

$$p^{\mathrm{bi}}_{j\beta} = \frac{\sum_{\alpha} |S^{+1}(\beta, \alpha)|(r_{j\alpha} + \mathrm{dev}^{+1}_{\beta\alpha}) + \sum_{\alpha} |S^{-1}(\beta, \alpha)|(r_{j\alpha} + \mathrm{dev}^{-1}_{\beta\alpha})}{\sum_{\alpha} |S^{+1}(\beta, \alpha)| + \sum_{\alpha} |S^{-1}(\beta, \alpha)|} \tag{32}$$

where the weights are chosen similarly as for Weighted Slope One.

It was shown that Slope One can outperform linear regression (*i.e.*, estimation by $f(x) = ax + b$) while having half the number of regressors [116,133]. This simple approach also reduces storage requirements and latency of the recommender system. Slope One has been used as a building block to improve other algorithms [134–136]. For instance, it can be combined with user-based collaborative filtering to address the data sparsity problem via filling the vacant ratings of the user–item matrix by the Slope One scheme, and thus improving the prediction accuracy [134].

4.2. How to define similarity

The key problem of similarity-based algorithms is how to define similarity between users or objects. When explicit ratings are available, similarity is usually defined using a correlation metric such as Pearson, for example (two users are considered as similar when they tend to give similar ratings to the objects they rate). When there is no rating information available, similarity can be inferred from the structural properties of the input data (two users are considered as similar when they liked/bought many objects in common). Besides, external information such as users' attributes, tags and objects' content meta information can be utilized to estimate similarity better.

4.2.1. Rating-based similarity

In many online e-commerce services, users are allowed to evaluate the consumed objects by ratings. For example, in Yahoo Music, users vote each song with one to five stars representing "Never play again" (⋆), "It is ok" (⋆⋆), "Like it" (⋆⋆⋆), "Love it" (⋆⋆⋆⋆) and "Can't get enough" (⋆⋆⋆⋆⋆). With explicit rating information we can measure the similarity between two users or between two objects by *Cosine index* [15,137] which is defined as

$$s^{\cos}_{xy} = \frac{\boldsymbol{r}_x \cdot \boldsymbol{r}_y}{|\boldsymbol{r}_x||\boldsymbol{r}_y|}. \tag{33}$$

For quantifying the similarity between users, $\boldsymbol{r}_x$, $\boldsymbol{r}_y$ are rating vectors in the N-dimensional object space while for similarity between objects, $\boldsymbol{r}_x$, $\boldsymbol{r}_y$ are vectors in M-dimensional user space. Note that, in the calculation of rating-based similarity, it is necessary to eliminate the rating tendencies of users and/or on items, otherwise the similarity is less meaningful. Actually,

according to a recently reported smart method, in some rating systems, via proper usage of rating tendencies, one could predict the unknown ratings with remarkably higher accuracy than the simply similarity-based methods [116].

The rating correlation can also be measured by *Pearson coefficient* (PC) [15,137]. To quantify similarity between users u and v, it reads

$$s_{uv}^{PC} = \frac{\sum_{\alpha \in O_{uv}} (r_{u\alpha} - \bar{r}_u)(r_{v\alpha} - \bar{r}_v)}{\sqrt{\sum_{\alpha \in O_{uv}} (r_{u\alpha} - \bar{r}_u)^2} \sqrt{\sum_{\alpha \in O_{uv}} (r_{v\alpha} - \bar{r}_v)^2}}, \tag{34}$$

where $O_{uv} = \Gamma_u \cap \Gamma_v$ indicates the set of objects rated by both u and v. A *constrained Pearson coefficient* proposed by Shardanand and Maes [127] consists of substituting the user mean in Eq. (34) with a "central" rating (for example, in the scale from 1 to 5, one can set the central rating to be 3). The idea is to take into account the difference between positive (above the central rating) and negative ratings (below the central rating). A *weighted Pearson coefficient* is based on the idea of capturing the confidence that can be placed on similarity values (when two users evaluated only a few objects in common, their potentially high similarity should not be trusted as much as for a pair of users with many overlapping objects). It was proposed [138] to weight the Pearson coefficient as

$$s_{uv}^{WPC} = \begin{cases} s_{uv}^{PC} \dfrac{|O_{uv}|}{H} & \text{for } |O_{uv}| \leq H, \\ s_{uv}^{PC} & \text{otherwise} \end{cases} \tag{35}$$

where H is a threshold, determined experimentally, beyond which the correlation measure can be trusted.

Analogically, Pearson similarity between objects α and β reads

$$s_{\alpha\beta}^{PC} = \frac{\sum_{u \in U_{\alpha\beta}} (r_{u\alpha} - \bar{r}_\alpha)(r_{u\beta} - \bar{r}_\beta)}{\sqrt{\sum_{u \in U_{\alpha\beta}} (r_{u\alpha} - \bar{r}_\alpha)^2} \sqrt{\sum_{u \in U_{\alpha\beta}} (r_{u\beta} - \bar{r}_\beta)^2}}, \tag{36}$$

where $U_{\alpha\beta}$ is the set of users who rated both α and β, and $\bar{r}_\alpha$ is the average rating of object α. Experiments have shown that the Pearson coefficient performs better than the cosine vector index [109]. When only binary ratings are available (like or dislike, purchase or no purchase, click or no click, etc.), the cosine and Pearson coefficient can still be applied to quantify the similarity of vectors with binary elements. For example, Amazon's patented algorithm [117] computes the cosine similarity between binary vectors representing users' purchases and use it in item-based collaborative filtering.

4.2.2. Structural similarity

As we have mentioned above, similarity can be defined using the external attributes such as tag and content information. However, the required data is usually very difficult to collect. Another simple and effective way to quantify the similarity, *structural similarity* [139], is based solely on the network structure of the data. Recent research shows that the structural-based similarity can produce better recommendations that the Pearson correlation coefficient, especially when the input data is very sparse [122].

To calculate the structural similarity between users or objects, we generally project the user–object bipartite network which contains the complete information about the system into a monopartite user–user or object–object network (for more information on this aspect of similarity see [103]). In the simplest case, two users are considered similar if they have voted at least one common object (analogically, two objects are considered similar if they have been co-voted by at least one user). More refined similarity metrics that can be roughly categorized as node-dependent vs. path-dependent, local vs. global, parameter-free vs. parameter-dependent, and so on—here we review some of them.

(i) *Node-dependent similarity*. The simplest weighted similarity index is Common Neighbors (CN) where the similarity of two nodes is directly given by the number of common neighbors (think of the number of users who bought both objects α and β or the number of objects shared by users u and v). By considering degrees of the two target nodes, six variations of CN were derived: Salton Index [140], Jaccard Index [141], Sørensen Index [142], Hub Promoted Index (HPI) [143], Hub Depressed Index (HDI) and Leicht–Holme–Newman Index (LHN1[6]) [144]. One can further take into account degrees of respective common neighbors to reward less-connected neighbors with a higher weight as in Adamic–Adar Index (AA) [145] and Resource Allocation Index (RA) [102]. Note that since AA uses a logarithmic weighting, it penalizes high-degree common neighbors less than RA. Finally, Preferential Attachment Index (PA) builds on the classical preferential attachment rule in network science [146]. This index has been used to quantify the functional significance of links subject to various network-based dynamics, such as percolation [147], synchronization [148] and transportation [149]. Note that these similarity can be computed also for a bipartite network where common neighbors are objects and users when considering user and object similarity, respectively. A summary of mathematical definitions of these similarity indices is shown in Table 4.

[6] We use the abbreviation LHN1 to distinguish this index to another index named as LHN2 also proposed by Leicht, Holme and Newman.

Table 4
Mathematical definitions of the described node-dependent similarity indices. Γ_x denotes the set of neighbors of node x (which can be either a user or an object node) and k_x is the degree of node x.

Index	Definition
CN	$s_{xy} = \lvert\Gamma_x \cap \Gamma_y\rvert$
Salton	$s_{xy} = \lvert\Gamma_x \cap \Gamma_y\rvert / \sqrt{k_x k_y}$
Jaccard	$s_{xy} = \lvert\Gamma_x \cap \Gamma_y\rvert / \lvert\Gamma_x \cup \Gamma_y\rvert$
Sørensen	$s_{xy} = 2\lvert\Gamma_x \cap \Gamma_y\rvert / (k_x + k_y)$
HPI	$s_{xy} = \lvert\Gamma_x \cap \Gamma_y\rvert / \min\{k_x, k_y\}$
HDI	$s_{xy} = \lvert\Gamma_x \cap \Gamma_y\rvert / \max\{k_x, k_y\}$
LHN1	$s_{xy} = \lvert\Gamma_x \cap \Gamma_y\rvert / (k_x k_y)$
AA	$s_{xy} = \sum_{z \in \Gamma_x \cap \Gamma_y} 1/\ln k_z$
RA	$s_{xy} = \sum_{z \in \Gamma_x \cap \Gamma_y} 1/k_z$
PA	$s_{xy} = k_x k_y$

(ii) *Path-dependent similarity*. The basic assumption here is that two nodes are similar if they are connected by many paths. Since elements of an n-th power of the adjacency matrix, A^n, are equal to the number of distinct paths between respective pairs of nodes, path-dependent similarity metrics can be usually written in a compact form such as

$$s_{xy}^{LP} = (\mathsf{A}^2)_{xy} + \epsilon(\mathsf{A}^3)_{xy} \tag{37}$$

for the Local Path Index [150] where only paths of length two and three count and ϵ is a damping parameter. (Note that in a bipartite network, only paths of an even length can exist between nodes of the same kind.) By including paths of all lengths, we obtain the classical Katz similarity [151] which is defined as

$$s_{xy}^{Katz} = \beta A_{xy} + \beta^2(\mathsf{A}^2)_{xy} + \beta^3(\mathsf{A}^3)_{xy} + \cdots, \tag{38}$$

where β is a damping factor controlling the path weights. This can be written as $s^{Katz} = (I - \beta\mathsf{A})^{-1} - I$. A variant of the Katz index, *Leicht–Holme–Newman Index* (LHN2) [144], was proposed where the term $(\mathsf{A}^l)_{xy}$ is replaced with $(\mathsf{A}^l)_{xy}/\mathrm{E}[(\mathsf{A}^l)_{xy}]$ where $\mathrm{E}[X]$ is the expected value of X.

(iii) *Random-walk-based similarity*. Another group of methods is based on random walks on networks.

Average commute time: The average commute time between nodes x and y is defined as the average number of steps required by a random walker starting from node x to reach node y plus that from y to x. It can be obtained in terms of the pseudoinverse of the network's Laplacian matrix, L^+, as [152,153]

$$n(x, y) = \left((\mathsf{L}^+)_{xx} + (\mathsf{L}^+)_{yy} - 2(\mathsf{L}^+)_{xy}\right)E, \tag{39}$$

where E is the number of edges in the network. Assuming that two nodes are similar if they have small average commute time, similarity between nodes x and y can be defined as the reciprocal of their average commute time

$$s_{xy}^{ACT} = \frac{1}{(\mathsf{L}^+)_{xx} + (\mathsf{L}^+)_{yy} - 2(\mathsf{L}^+)_{xy}} \tag{40}$$

where the constant factor E has been removed.

Cosine-based on L^+: This index is an inner-product-based measure. In the Euclidean space spanned by $\boldsymbol{v}_x = \Lambda^{\frac{1}{2}}\mathsf{U}^T\boldsymbol{e}_x$ where U is an orthonormal matrix composed of the eigenvectors of L^+ ordered in a decreasing order of their eigenvalues λ_x, $\Lambda = \mathrm{diag}(\lambda_x)$, $\boldsymbol{e}_x$ is a column base vector ($(\boldsymbol{e}_x)_y = \delta_{xy}$) and T is matrix transposition, elements of the pseudoinverse of the Laplacian matrix are the inner products of the node vectors, $(\mathsf{L}^+)_{xy} = \boldsymbol{v}_x^T\boldsymbol{v}_y$. Consequently, cosine similarity is defined as [153]

$$s_{xy}^{\cos+} = \frac{\boldsymbol{v}_x^T\boldsymbol{v}_y}{|\boldsymbol{v}_x||\boldsymbol{v}_y|} = \frac{(\mathsf{L}^+)_{xy}}{\sqrt{(\mathsf{L}^+)_{xx}(\mathsf{L}^+)_{yy}}}. \tag{41}$$

Random walk with restart: This index is a direct application of the PageRank algorithm [154]. Consider a random walker starting from node x recursively moves to a random neighbor with probability c and returns to node x with probability $1-c$. Denoting by q_{xy} the resulting stationary probability that the walker is located at node y, we can write

$$\boldsymbol{q}_x = c\mathsf{P}^T\boldsymbol{q}_x + (1 - c)\boldsymbol{e}_x \tag{42}$$

where P is the transition matrix with elements $P_{xy} = 1/k_x$ if x and y are connected and $P_{xy} = 0$ otherwise. The solution to this equation is

$$\boldsymbol{q}_x = (1 - c)(I - c\mathsf{P}^T)^{-1}\vec{e}_x. \tag{43}$$

Finally, the similarity index is defined as

$$s_{xy}^{RWR} = q_{xy} + q_{yx}. \tag{44}$$

A fast algorithm to calculate this index was proposed [155] and the application to recommender systems was studied in [122] where it was found that this similarity performs better than the Pearson correlation coefficient.

SimRank: This index is defined based on the assumption that two nodes are similar if they are connected to similar nodes. This allows us to define SimRank in a self-consistent way [156] as

$$s_{xy}^{SimRank} = C \frac{\sum_{z\in\Gamma_x}\sum_{z'\in\Gamma_y} s_{zz'}^{SimRank}}{k_x k_y}, \tag{45}$$

where $s_{xx} = 1$ and $C \in [0, 1]$ is a free parameter. SimRank can also be interpreted by the random-walk process: $s_{xy}^{SimRank}$ measures how fast are two random walkers, who respectively start at nodes x and y, expected to meet at a certain node.

Matrix forest index: This index introduces similarity between x and y as the ratio of the number of spanning rooted forests such that nodes x and y belong to the same tree rooted at x to all spanning rooted forests of the network (for details see [157]). Its mathematical definition

$$s^{MFI} = (I + \mathsf{L})^{-1} \tag{46}$$

can be further parametrized to obtain a variant of MFI

$$s^{PMFI} = (I + \alpha\mathsf{L})^{-1}, \quad \alpha > 0. \tag{47}$$

According to the authors, $\alpha > 0$ determines the proportion of accounting for long connections between vertices of the graph versus short ones.

Local random walk: To measure similarity between nodes x and y, a random walker is introduced in node x and thus the initial occupancy vector is $\boldsymbol{\pi}_x(0) = \boldsymbol{e}_x$. This vector evolves as $\boldsymbol{\pi}_x(t+1) = \mathsf{P}^T\boldsymbol{\pi}_x(t)$ for $t \geq 0$. The LRW index at time step t is defined [158] as

$$s_{xy}^{LRW}(t) = q_x\pi_{xy}(t) + q_y\pi_{yx}(t) \tag{48}$$

where q is the initial configuration function and t denotes the time step. In [158] it was suggested to use a simple approach where q is determined by node degree: $q_x = k_x/M$. Note that in bipartite networks, an even time step must be used to obtain similarity between nodes of the same kind.

Superposed random walk: Similar to the RWR index, in [158] they proposed another index where the random walker is continuously released at the starting point, resulting in a higher similarity between the target node and its nearby nodes. The mathematical expression reads

$$s_{xy}^{SRW}(t) = \sum_{\tau=1}^{t} s_{xy}^{LRW}(\tau) = \sum_{\tau=1}^{t}[q_x\pi_{xy}(\tau) + q_y\pi_{yx}(\tau)]. \tag{49}$$

In [153], several random-walk-based similarity indices, such as ACT, cos + and MFI, were applied in collaborative filtering. Their experimental results show that in general, Laplacian-based similarities perform well.

4.2.3. Similarity involving external information

Besides the fundamental user–object relations and the ratings, additional information can be exploited to define or improve the node similarity.

(i) *Attributes*. The dimension and elements of the attribute vectors are defined in advance by some domain experts, and is identical to all users (objects) in the system. The similarity between two users (objects) is obtained by calculating the correlation of their corresponding attributes vectors. For example, user profiles, usually including age, sex, nationality, location, career, etc., can be simply applied to quantify the similarity between users based on the assumption that two users are similar when they have many common features. In [159], a hybrid method considering both attributes of objects and the ratings was shown to provide better recommendations than when these two sources of information are used independently. However, the application of attributes represents some risks to user privacy—as shown by a recent work on de-anonymization of large data sets [25], collection and utilization of attribute data poses several sensitive issues. For more information on the issues of user privacy see [160].

(ii) *Contents*. Modern information retrieval techniques allow us to automatically extract content and meta information of the available objects. Object similarity can hence be calculated based on the content comparison of the given objects. This is usually referred as content-based recommendation in literature [161]. Unlike collaborative filtering, in a content-based algorithm, recommendations are made based solely on the profile built up by analyzing the content of objects that the target user has rated in the past. The recommendation problem hence becomes a search for objects whose content is most similar to the content of objects already preferred by the target user. The classical method to weigh content is TF–IDF

(term frequency-inverse document frequency) [140], which is a weighing metric often used in information retrieval and text mining. A term, t, in a given document, d, is weighted as,

$$W_{t,d} = \mathrm{tf}(t, d) \times \log \frac{|D|}{|d : t \subseteq d|}, \tag{50}$$

where $\mathrm{tf}(t, d)$ is the frequency of t in document d, $|D|$ is number of all observed documents. Then $W_{t,d}$ can be used to measure the similarly of objects defined in Section 4.2.2. In addition, if two users have collected objects with similar content, we may assume that these two users are similar.

Since both content-based method and collaborative filtering have their individual limitations, such as CF systems do not explicitly incorporate feature information and face the sparsity and cold-start problems, while content-based systems do not necessarily incorporate the information in preference similarity across individuals (see summaries and discussions in Refs. [15,123]), many hybrid algorithms are proposed to avoid certain weaknesses in each approach and thereby improve the recommendation performance. The combination methods can be classified into four categories: (i) implement separate collaborative and content-based methods and then combine their predictions [162,163]; (ii) add content-based characteristics to collaborative models [164–166]; (iii) add collaborative characteristics to content-based models [167] and (iv) develop a general unified model that integrate both content-based and collaborative characteristics [32,168–171]. However, these methods are only effective if the objects contain rich content information that can be automatically extracted. This is the case for recommendation of books, articles and bookmarks, but not for videos, music tracks or pictures.

(iii) *Tags*. Collaborative tagging systems emerged with the advent of Web2.0 [94]. Different from traditional taxonomy with hierarchical structure, tagging systems allow users to freely assign keywords (which are usually referred to as *tags*) to manage their own collections without the limitation of a preset vocabulary. Tags provide a rich source of information for recommendation purposes. With the tagging information, algorithms can be easily designed to calculate user similarity and object similarity by considering tag vectors in user and object space, respectively. To alleviate the effects of spam and magnify personalized user preferences, weighting techniques are often applied to measure the importance of each element in a given tag vector.

5. Dimensionality reduction techniques

Dimensionality reduction aims at downsizing the amount of relevant data while preserving the major information content. It is often applied in areas such as data mining, machine learning and cluster analysis. Most techniques of dimensionality reduction involve feature extraction which makes use of hidden variables, or so called latent variables, to describe the underlying causes of co-occurrence data. In the context of movie selection, potential viewers may consider genres such as action, romance or comedy features in a movie, which constitute the latent variables. These latent variables are usually represented by multi-dimensional vectors. A simplified picture of two-dimensional vectors of action and romance is shown in Fig. 7, which shows that user *Peter* has a preference in action movies, while user *Mary* prefers romantic content. Given these vectors and the corresponding vectors of movies, we can define the expected rating of a user on a move as the scalar product of their vectors. For instance, we expect *Peter* prefers movie β rather than α, while the opposite is true for *Mary*. Recommendations can thus be made once the vectors are computed. If K hidden variables are used, the latent vectors are K-dimensional and dimensionality reduction is achieved if $K(N + M) < NM$, since the number of relevant variables is reduced. In practice, these techniques are particularly suitable for large data sets which are costly to store and manipulate.

Instead of introducing latent variables which describe interests and genres, users and objects can also be assigned to individual classes which leads to reduction of data dimension. In this case, the co-occurrence of a user–object pair is explained by the relation between the classes to which the user and the object belong to. Though the original intention for such classification is not to reduce the data dimension, the number of classes used is usually significantly smaller than the number of users and objects, which then results in reduction of dimensionality.

Dimensionality reduction is in particular well applicable in collaborative filtering (it is sometimes referred to as model-based collaborative filtering), as for most applications only a small fraction of user–object pairs are observed such that the number of relevant variables can be significantly reduced. Reductions in dimensionality effectively preserve the information content while drastically decreasing the computation complexity and memory requirements for making recommendations. In this section, several techniques of dimensionality reduction with implementation to recommender systems are discussed, including singular value decomposition (SVD) [172], Bayesian clustering [173], probabilistic latent semantic analysis (pLSA) [174] and latent Dirichlet allocation (LDA) [175].

5.1. Singular value decomposition (SVD)

We start with a $N \times M$ matrix R whose element $r_{i\alpha}$ corresponds to the rating of user i to object α (if the rating has not yet been given, the corresponding element of R is zero). In the case without numeric ratings, R becomes the adjacency matrix as $r_{i\alpha} = 0, 1$ for connected and unconnected user–object pair, respectively. Recommendation process then aims to determine

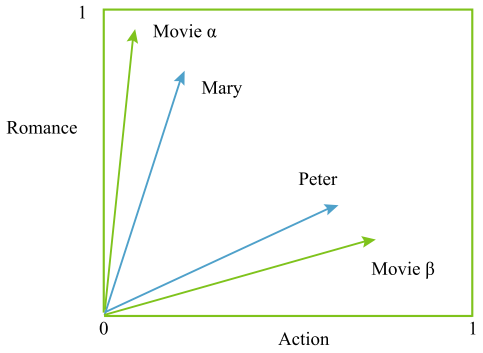

Fig. 7. An example of a user's movie selection process with hidden variables.

which presently zero entries of R have high chance to be non-zero in the future. Note that R is a sparse matrix for most applications because only a small fraction of all its elements are different from zero.

Dimensionality reduction is achieved by introducing K hidden variables which categorize tastes of users and attributes of objects. The original R is approximated as the product of two matrices

$$R \approx WV \tag{51}$$

where W and V are respectively matrices with dimension $N \times K$ and $K \times M$. They contain the taste information for users and content information for objects, respectively, expressing them in terms of K hidden variables. Because of these hidden variables, SVD belongs to a broad class of latent semantic analysis (LSA) techniques. From the product of W and V in Eq. (51), we see that objects are selected by users based on the overlap between a user's tastes and a movie's attributes. When the number of hidden variables is smaller than N and M, the number of parameters needed to describe the system reduces from NM for the original R to $NK + KM$ for the product WV. This approach is also known as matrix factorization (MF) as R is factorized into a product of matrices.

To obtain W and V, singular value decomposition (SVD) is a common algebraic tool in LSA which results in downsizing of relevant variables and, at the same time, finding a good approximation of R. In SVD, R is factorized as

$$R = W\Sigma V \tag{52}$$

where Σ is a $K \times K$ diagonal matrix, and equality in the above factorization holds with $K = \min(N, M)$. The matrix Σ contains the so-called *singular values* of R, which are indeed the square root of the eigenvalues of RR^* (or R^*R). To benefit from dimensionality reduction, we put $K < \min(N, M)$ which corresponds to the K-rank approximation in SVD and include only the K largest singular values in Σ and replace the others by zero. Equality in Eq. (52) no longer holds and R is approximated with $\tilde{R}$ given by

$$R \approx \tilde{R} = W\tilde{\Sigma}V \tag{53}$$

where $\tilde{\Sigma}$ is the K-rank approximation of Σ. Given certain conditions on the matrix R, for instance a low-rank R, quantities include the lower and the upper bound of the prediction errors, and the smallest number of non-zero entries in R to achieve an accurate prediction have been studied and proved analytically [176–179]. An estimate of the rank K is also given in [180]. These works also lead to useful algorithms. We refer readers to [176–180] and the references therein for details.

Here we show schematically the approach of SVD following [172]. First of all, it can be shown that $\tilde{R}$ can be found by minimizing the Frobenius norm of the matrix $R - \tilde{R}$, that is

$$\tilde{R} = \underset{\{\tilde{R}'\}}{\operatorname{argmin}} \|R - \tilde{R}'\| \tag{54}$$

with the rank of $\tilde{R}'$ restricted to be K. The Frobenius norm is given by

$$\|R - \tilde{R}\| = \sqrt{\sum_{i\alpha} (r_{i\alpha} - \tilde{r}_{i\alpha})^2}, \tag{55}$$

which corresponds to the root square error of $\tilde{R}$ with respect to R, with $\tilde{r}_{i\alpha}$ denoting the element (i, α) in $\tilde{R}$. SVD thus provides a simple cost function for measuring the agreement between R and $\tilde{R}$. To obtain $\tilde{R}$ explicitly for a particular R, an simple iterative approach [172] based on gradient descent can be employed. Σ in Eq. (53) is first absorbed into either W or V to obtain $\tilde{R} = WV$ as in Eq. (51). Our task is to obtain the optimal W and V for which $\tilde{R} = WV$ minimizes the norm in Eq. (55). We now express $\tilde{r}_{i\alpha}$ as

$$\tilde{r}_{i\alpha} = \sum_{k=1}^{K} w_{ik} v_{k\alpha}. \tag{56}$$

After substitution of the above expression, we minimize the difference between $r_{i\alpha}$ and $\tilde{r}_{i\alpha}$ for an *observed* pair $(i\alpha)$ by differentiation of $(r_{i\alpha} - \tilde{r}_{i\alpha})^2$, namely

$$\frac{\partial}{\partial w_{ik}}(r_{i\alpha} - \tilde{r}_{i\alpha})^2 = -2w_{ik}\left(r_{i\alpha} - \sum_{k=1}^{K} w_{ik}v_{k\alpha}\right), \tag{57}$$

$$\frac{\partial}{\partial v_{k\alpha}}(r_{i\alpha} - \tilde{r}_{i\alpha})^2 = -2v_{k\alpha}\left(r_{i\alpha} - \sum_{k=1}^{K} w_{ik}v_{k\alpha}\right). \tag{58}$$

Since the norm is non-negative by definition, we minimize the norm square which leads to the same result while encountering simpler expressions in the process. The obtained gradients can be used to write a gradient descent-based updating procedure for w_{ik} and $v_{k\alpha}$ in the form

$$w_{ik}(t+1) = w_{ik}(t) + 2\eta w_{ik}(t)e_{i\alpha}(t), \tag{59}$$

$$v_{k\alpha}(t+1) = v_{k\alpha}(t) + 2\eta v_{k\alpha}(t)e_{i\alpha}(t), \tag{60}$$

where t denotes the iteration step and

$$e_{i\alpha}(t) = r_{i\alpha} - \sum_{k=1}^{K} w_{ik}(t)v_{k\alpha}(t). \tag{61}$$

The learning rate $\eta > 0$ should be small to avoid big jumps in the solution space. With random initial conditions on w_{ik} and $v_{k\alpha}$, these equations are iterated until the squared norm shows no further decrease. Other procedures such as the variants of stochastic gradient descent may be applied to improve computational efficiency [181]. As suggested by [172], subtracting a small term from the gradient can prevent large resulting weights of w_{ik} and $v_{k\alpha}$ (this is equivalent to minimizing $\|R - \tilde{R}\|^2 + \frac{\lambda}{2}\|W\|^2 + \frac{\lambda}{2}\|V\|^2$, or similar to using the Tikhonov regularization which gives preference to solutions with small norms in ill-posed problems), which leads to the following update rule

$$w_{ik}(t+1) = w_{ik}(t) + \eta\big(2w_{ik}(t)e_{i\alpha}(t) - \lambda w_{ik}(t)\big), \tag{62}$$

$$v_{k\alpha}(t+1) = v_{k\alpha}(t) + \eta\big(2v_{k\alpha}(t)e_{i\alpha}(t) - \lambda v_{ik}(t)\big) \tag{63}$$

where a new parameter $\lambda \geq 0$ is introduced to this end; $\lambda > 0$ usually leads to better accuracy of the results. Using the resulting w_{ik} and $v_{k\alpha}$, we can compute $\tilde{R} = WV$ which has non-zero values on entries where the input matrix R are zero (representing unexpressed evaluations)—element $\tilde{r}_{i\alpha}$ then predicts the possible rating given by user i to object α.

Note that while $\|R - \tilde{R}\|$ measures the "error" of $\tilde{R}$ with respect to R, it is usually very different from (greater than) the error of the ultimate rating estimates. The reason for this, of course, lies in the fact that $\|R - \tilde{R}\|$ is minimized while knowing R, and the $K(N+M)$ free parameters usually allow us to achieve very low value of $\|R - \tilde{R}\|$. To measure performance of this method correctly, the available data must be divided into a training set (which is used to "learn" W and V) and a test set (see Section 3.3 for details). Increasing K does not automatically improve the results: over-fitting the data (providing too many free parameters) can lead to inferior accuracy. The use of this and other machine-learning methods hence requires a certain amount of tests and/or experience.

In addition to a simple iteration procedure, SVD also enjoys a flexibility in dealing with additional data. For instance, one can easily include the influence of individual rating bias in the framework. Suppose user i tends to give an average of b_i more score for all his items when compared to other users, while object α tends to receive b_α more scores when compared to other items, the predicted scores can be expressed as [182]

$$\tilde{r}_{i\alpha} = \mu + b_i + b_\alpha + \sum_{k=1}^{K} w_{ik}v_{k\alpha}, \tag{64}$$

where μ is the average value among all user–object pairs. In this case, $\|R - \tilde{R}\|$ is minimized with respect to b_i, b_α, w_{ik} and $v_{k\alpha}$ for all i, k and α. Other than individual bias, there are other variants of SVD which utilizes the relations between users in social network, for instance the similarity in taste between friends, to improve the recommendation accuracy by the factorized matrices [183].

Apart from the case with user–object ratings as the only input data, the above-described procedure can be generalized to incorporate additional information [172]. Suppose $d_{i\alpha}$ is the additional information (*e.g.* date) associated with a given user–object pair. Transforming $d_{i\alpha}$ into positive integers l with $1 \leq l \leq L$, the elements y_{kl} in the $K \times L$ matrix Y contain the relation between each of these additional data and the hidden variables. For example, a large number of movies with romantic content are reviewed on the Valentine day, leading to a large value in the corresponding entries of romance and Valentine day in Y. With the additional information, the matrix $\tilde{R}$ can be expressed as

$$\tilde{r}_{i\alpha} = \sum_{k=1}^{K} w_{ik}v_{k\alpha}y_{kd_{i\alpha}}. \tag{65}$$

Similar derivations based on gradient descent provide updating procedures for w_{ik}, $v_{k\alpha}$ and $y_{k,d_{i\alpha}}$ which can then be used to obtain $\tilde{R}$ with the smallest error with respect to R.

5.2. Bayesian clustering

Before describing a probabilistic version of LSA, we first introduce the Bayesian clustering method which is also probabilistic but simpler in formulation. In Bayesian networks, the value of a variable depends only on the value of its parent variables. For example, the probability of a variable x is described by the conditional probability $P(x|\mathrm{pa}_x)$ where pa_x is the parent variable of x. The joint probability of several independent variables can be factorized as $P(x_1, \ldots, x_N) = \prod_{i=1}^{N} P(x_i|\mathrm{pa}_{x_i})$ which represents the dependency structure of the variables. However, obtaining the most relevant dependency structure, *i.e.* the dependency relation between different nodes in the Bayesian network, is not a trivial task.

For the purpose of personalized recommendation, we describe a two-sided clustering [173,184] which is easy to implement. To obtain the rating for an unobserved user–object pair, one classifies users and objects respectively into K_{user} and K_{object} classes. The values of K_{user} and K_{object} are parameters on the algorithm, similarly to K which was a parameter of SVD. We assume that there is a simple Bayesian network that underlies the input data—the simplest assumption is that $r_{i\alpha}$ depends only on the user's class c_i and the object's class c_α. The probability of $r_{i\alpha}$ can then be written as

$$P(r_{i\alpha}) = \sum_{c_i=1}^{K_{\mathrm{user}}} \sum_{c_\alpha=1}^{K_{\mathrm{object}}} P(r_{i\alpha}|c_i, c_\alpha) P(c_i) P(c_\alpha).$$

To obtain an estimate of $r_{i\alpha}$, we need to find $P(c_i)$, $P(c_\alpha)$ and $P(r_{i\alpha}|c_i, c_\alpha)$, which is effectively $P(r|x, y)$ as the rating r is dependent merely on the user class x and the object class y. This can be done by applying the inference methods including the marginal estimation by belief propagation [185,186] and likelihood maximization by expectation maximization [187]. Here we describe another simple scheme, known as the Gibbs sampling method [188], which is similar to the heat bath algorithm in statistical physics [189,190]. Gibbs sampling is useful when the joint distribution of all variables is difficult to sample (in our case, $P(\{r|(x, y)\}, \{c_i\}, \{c_\alpha\})$), while sampling the conditional probability of individual variables given all other variable is comparatively easy (*e.g.*, $P(c_{i'}|\{r|(x, y)\}, \{c_i\}_{-i'}, \{c_\alpha\})$). It is similar to the heat bath algorithm as it models the state of a system moving in a phase space and samples at certain time intervals the required (physical) quantities.

Here we describe the Gibbs sampling scheme suggested in [173] to sample the state $(\{r|(x, y)\}, \{c_i\}, \{c_\alpha\})$ of the system which allows us to evaluate the predicted ratings for any unobserved user–object pair. This scheme was developed to sample binary ratings so we assume that $r_{i\alpha} = 1$ when user i has collected α or rated object by a score above a certain threshold; $r_{i\alpha} = 0$ otherwise. In this case, one can represent $P(r|x, y)$ by a single value variable P_{xy} corresponding to the probability that a user from category x likes an object from category y.

We start the algorithm with a random latent class assignment for all users and objects and evaluate N_x, N_y and N_{xy}, respectively correspond to the number of users in class x, the number of objects in class y, and the number of observed user–object pairs (*i.e.*, $r_{i\alpha} = 1$) for all pairs of classes. We then draw values of P_{xy} from the beta distribution with parameters $(N_{xy}+1, N_xN_y-N_{xy}+1)$, or simply approximate P_{xy} by its mean $P_{xy} = N_{xy}/N_xN_y$. Similarly, the variables P_x and P_y, respectively defined as the probability that a random user or object is classified in class x or y, are drawn from Dirichlet distributions or simply approximated by $P_x = N_x/N$ and $P_y = N_y/M$. All these values of P_{xy}, P_x and P_y are used to evaluate the transition probability of the system from the present state to another state in the phase space. We then successively pick either a random user or a random object and update its latent class as [173]

$$P(c_i = x) \propto P_x \prod_{y=1}^{K_{\mathrm{object}}} P_{xy}^{\sum_\alpha r_{i\alpha}\delta_{c_\alpha,y}} (1 - P_{xy})^{\sum_\alpha (1-r_{i\alpha})\delta_{c_\alpha,y}}, \tag{66}$$

for users, and

$$P(c_\alpha = y) \propto P_y \prod_{x=1}^{K_{\mathrm{user}}} P_{xy}^{\sum_i r_{i\alpha}\delta_{c_i,x}} (1 - P_{xy})^{\sum_i (1-r_{i\alpha})\delta_{c_i,x}} \tag{67}$$

for objects. These equations involve large powers of probability values and may lead to inaccurate numerics during the computation. Instead of computating the probability directly, one can first compute the powers and convert them to probability during the update of latent classes. The values of N_x, N_y and N_{xy}, and thus of P_x, P_y and P_{xy}, are updated after each update of user class or object class. After a sufficient number of iterations, we can start sampling estimated ratings of unobserved user–object pairs at regular time intervals which should be long enough to ensure low correlation between consecutive sampled states. For instance, the predicted rating for user i on object α can be obtained by

$$\tilde{r}_{i\alpha} = \sum_t P_{xy}(t_c + tT)\delta_{x,c_i(t_c+tT)}\delta_{u,c_\alpha(t_c+tT)}, \tag{68}$$

where t_c and T are respectively the convergence time and the sampling time interval. We can also store the state $(\{P_{xy}\}, \{c_i\}, \{c_\alpha\})$ at each sampling time and use the above equation to obtain the predictions afterward.

We remark that the above two-sided Bayesian network corresponds to the simplest dependency structure which relates ratings to merely user and object classes. More comprehensive Bayesian relation can be derived, for instance, to include the individual rating preference for objects [191] or to model the possibility of mixed membership [192]. Another class of

extensions is to build a probabilistic relational model (PRM) [184] to predict ratings by utilizing other meta-data including age, occupation and gender of users, or category, price and origin of objects. Although this comprehensive information generally improves recommendation results, to determine a valid dependency structure of meta-data is a non-trivial task in PRM.

5.3. Probabilistic latent semantic analysis (pLSA)

Probabilistic latent semantic analysis (pLSA) is similar to LSA in the sense that hidden variables are introduced in to explain the co-occurrence pairs of data. Unlike the algebraic SVD employed in LSA, pLSA is a statistical technique based on a probabilistic model. Well developed inference methods including likelihood maximization [187] and Gibbs sampling [188] can thus be employed in pLSA. pLSA models the relations between users and objects through the implicit overlap of genres, as compared to the two-sided Bayesian clustering where each user and object belong to a single specific category. In pLSA, the co-occurrence probability $P(i, \alpha)$ of user i and object α is expressed using the conditional probability given a hidden variable k

$$P(i, \alpha) = \sum_{k=1}^{K} P(i|k)P(\alpha|k)P(k). \tag{69}$$

Since $P(i|k)P(k) = P(k|i)P(i)$, this can be written as

$$P(i, \alpha) = P(i) \sum_{k=1}^{K} P(\alpha|k)P(k|i) \tag{70}$$

which leads to the conditional probability $P(\alpha|i)$ of an object α to be collected given the user i,

$$P(\alpha|i) = \sum_{k=1}^{K} P(\alpha|k)P(k|i), \tag{71}$$

which is already a quantity useful for personalized recommendation. Unlike the Bayesian clustering approach where the co-occurrences of users and objects are characterized by the coupled probability P_{xy} between the classes, users and objects are rendered independent in pLSA given the hidden variables—the co-occurrence probabilities are factorized. Our task is to obtain suitable forms of $P(\alpha|k)$ and $P(k|i)$ which provide accurate predictions of collected and recommended objects through $P(\alpha|i)$. We note that instead of $P(\alpha|i)$, $P(i, \alpha)$ can also be expressed as

$$P(i, \alpha) = P(\alpha) \sum_{k=1}^{K} P(i|k)P(k|\alpha) \tag{72}$$

to obtain $P(i|\alpha)$ which can be of interest for some purposes.

To obtain $P(\alpha|k)$ and $P(k|i)$, one can adopt a variational approach described in [174] to maximize the per-link log-likelihood of the observed data set which is given by

$$L(\boldsymbol{\phi}, \boldsymbol{\theta}) = \frac{1}{E} \sum_{(i,\alpha)} \log P(\alpha|i) = \frac{1}{E} \sum_{(i,\alpha)} \log\left(\sum_{k=1}^{K} P(\alpha|k)P(k|i) \right) \tag{73}$$

with respect to vectors $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$ which parametrize $P(\alpha|k)$ and $P(k|i)$ by $P(\alpha|k) = \phi_{\alpha}^{(k)}$ and $P(k|i) = \theta_{k}^{(i)}$. Here E is the total number of user–object links. We remark that the sum over (i, α) includes only the observed user–object pairs. We then employ the expectation maximization (EM) algorithms [187] to find the value of θ that maximize $L(\boldsymbol{\phi}, \boldsymbol{\theta})$. To achieve the goal, one can introduce the variational probability distribution $Q(k|i, \alpha)$ in Eq. (73) for each observed pair (i, α), with the constraint $\sum_{k=1}^{K} Q(k|i, \alpha) = 1$, which allows us to rewrite $L(\boldsymbol{\phi}, \boldsymbol{\theta})$ as

$$L(\boldsymbol{\phi}, \boldsymbol{\theta}) = \frac{1}{E} \sum_{(i,\alpha)} \log\left(\sum_{k=1}^{K} Q(k|i, \alpha) \frac{P(\alpha|k)P(k|i)}{Q(k|i,\alpha)} \right) \tag{74}$$

$$\geq \frac{1}{E} \sum_{(i,\alpha)} \sum_{k=1}^{K} Q(k|i, \alpha) \log \frac{P(\alpha|k)P(k|i)}{Q(k|i,\alpha)} := \mathcal{F}(Q, \boldsymbol{\phi}, \boldsymbol{\theta}) \tag{75}$$

with the inequality is justified by Jensens' inequality. $\mathcal{F}(Q, \boldsymbol{\phi}, \boldsymbol{\theta})$ can be written as

$$\mathcal{F}(Q, \boldsymbol{\phi}, \boldsymbol{\theta}) = \frac{1}{E} \sum_{(i,\alpha)} \left\{ \sum_{k=1}^{K} Q(k|i, \alpha) \log[P(\alpha|k)P(k|i)] + S_{i\alpha}(Q) \right\} \tag{76}$$

with $S_{i\alpha}(Q)$ being the entropy of the probability distribution Q for the pair (i, α) which is

$$S_{i\alpha}(Q) = -\sum_{k=1}^{K} Q(k|i,\alpha)\log Q(k|i,\alpha). \tag{77}$$

Since $\mathcal{F}$ serves as the lower bound of likelihood function $L(\boldsymbol{\phi}, \boldsymbol{\theta})$, we maximize $\mathcal{F}$ with respect to Q, $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$. The expectation maximization algorithm thus maximizes $\mathcal{F}$ by finding the optimal Q, $\boldsymbol{\phi}$, $\boldsymbol{\theta}$ alternatively. We first obtain the distribution Q which maximizes $\mathcal{F}$ by assuming a particular form of $P(\alpha|k)$ and $P(k|i)$, *i.e.* holding $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$ constant. Maximization of $\mathcal{F}$ in this step is subject to the normalization of $Q(k|i,\alpha)$ for every observed user–object pair which leads us to the Lagrangian

$$\mathcal{L}(Q, \boldsymbol{\phi}_t, \boldsymbol{\theta}_t) = \mathcal{F}(Q, \boldsymbol{\phi}_t, \boldsymbol{\theta}_t) + \sum_{(i,\alpha)} \lambda_{i\alpha}\left(\sum_{k=1}^{K} Q(k|i,\alpha) - 1\right) \tag{78}$$

where $\boldsymbol{\phi}_t$ and $\boldsymbol{\theta}_t$ denotes respectively $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$ after t iteration steps, or equivalently, $P_t(\alpha|k)$ and $P_t(k|i)$. $\mathcal{L}(Q, \boldsymbol{\phi}_t, \boldsymbol{\theta}_t)$ can then be differentiated to obtain the optimal Q for every observed user–object pair in the form

$$Q_t(k|i,\alpha) = \frac{P_t(\alpha|k)P_t(k|i)}{\sum_{k'=1}^{K} P_t(\alpha|k')P_t(k'|i)} = \frac{\phi_{\alpha,t}^{(k)}\theta_{k,t}^{(i)}}{\sum_{k'=1}^{K} \phi_{\alpha,t}^{(k')}\theta_{k',t}^{(i)}}. \tag{79}$$

This optimal Q is obtained from $\boldsymbol{\theta}_t$ and $\boldsymbol{\phi}_t$, thus we label it as Q_t.

We then proceed to obtain the distributions $P(\alpha|k)$ and $P(k|i)$, *i.e.* the value of $\phi_\alpha^{(k)}$ and $\theta_k^{(i)}$, by assuming Q is held fixed at $Q = Q_t$ obtained from Eq. (79). Due to the normalization of $P(\alpha|k)$ for all α and $P(k|i) = 1$ for all i, the corresponding Lagrangian has the form

$$\mathcal{L}(Q_t, \theta) = \mathcal{F}(Q_t, \theta) + \sum_k \lambda_k\left(\sum_{\alpha=1}^{M} P(\alpha|k) - 1\right) + \sum_i \lambda_i\left(\sum_{k=1}^{K} P(k|i) - 1\right). \tag{80}$$

After differentiation, the optimal $P(\alpha|k)$ and $P(k|i)$ can be found

$$P_{t+1}(\alpha|k) = \phi_{i,t+1}^{(k)} = \frac{\sum_i Q_t(k|i,\alpha)}{\sum_{\alpha'}\sum_i Q_t(k|i,\alpha')}, \tag{81}$$

$$P_{t+1}(k|i) = \theta_{k,t+1}^{(i)} = \frac{\sum_\alpha Q_t(k|i,\alpha)}{\sum_{k'}\sum_\alpha Q_t(k'|i,\alpha)} \tag{82}$$

where the summation involving i and α runs only over the observed user–object pairs. The optimal $P(\alpha|k)$ and $P(k|i)$ are obtained from $Q = Q_t$. We label them as $P_{t+1}(\alpha|k)$ and $P_{t+1}(k|i)$, respectively, because they constitute the basis for the next iteration step where Q_{t+1} is found. After stationary values of Q_t, $\boldsymbol{\phi}_t$ and $\boldsymbol{\theta}_t$ are found, Eq. (71) is used to obtain personalized recommendations.

As the above pLSA model considers a multinomial distribution $\phi_\alpha^{(k)}$ which can be used to model only binary preferences, one may consider the generalized pLSA [193] which allows for numeric ratings. The fast increasing number of independent variables used by pLSA (there are $K(N + M)$ of them) and the cold-start problem for new objects can be alleviated by assuming prior distributions on $\phi^{(k)}$ and $\theta^{(i)}$, such as Dirichlet priors discussed in the following section [175].

5.4. Latent Dirichlet Allocation (LDA)

Latent Dirichlet Allocation (LDA) [175] is similar to pLSA in the sense that hidden variables are present in a probabilistic way. While pLSA does not assume a specific prior distribution over $P(k|i)$, LDA assumes that priors that have the form of the Dirichlet distribution. LDA was applied to predict review scores based on the content of reviews [194] and to uncover implicit community structures in a social network [195]. It can be also extended to include general meta-data of users and objects [196].

For each user i there is a distribution $P(k|\theta^{(i)})$ where $\theta^{(i)}$ is a K-dimensional multinomial distribution with $\theta_k^{(i)}$ is the probability of user i belonging to the latent class k, *i.e.* $P(k|i) = \theta_k^{(i)}$. Unlike pLSA, the variable $\theta^{(i)}$ in LDA has a Dirichlet prior distribution with a K-dimensional parameter $\boldsymbol{a}$. The probability of observing user i with the collected object set $\{\alpha\}_i$ is

$$P(\{\alpha\}_i|\boldsymbol{a}, \boldsymbol{\phi}) = \int \mathrm{d}\theta^{(i)} P(\theta^{(i)}|\boldsymbol{a})\left[\prod_{\mu=1}^{k_i}\sum_{k=1}^{K} P(\alpha_{i,\mu}|k)P(k|\theta^{(i)})\right] \tag{83}$$

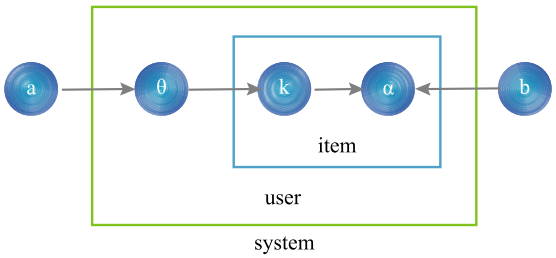

Fig. 8. A so-called plate notation representing the LDA recommendation model: $\boldsymbol{a}$ and $\boldsymbol{b}$ represent the model's parameters at the system level, which characterize respectively the Dirichlet distribution of $\theta_k^{(i)}$ and $\phi_\alpha^{(k)}$, where $P(k|i) = \theta_k^{(i)}$ and $P(\alpha|k) = \phi_\alpha^{(k)}$.

where k_i is the number of objects collected by user i and $\alpha_{i,\mu}$ is the μ-th object collected by i. The Dirichlet prior distribution $P(\theta^{(i)}|\boldsymbol{a})$ is

$$P(\theta^{(i)}|\boldsymbol{a}) = \frac{\Gamma\left(\sum_{k=1}^{K} a_k\right)}{\prod_{k=1}^{K}\Gamma(a_k)} \prod_{k=1}^{K}\left(\theta_k^{(i)}\right)^{a_k-1}, \tag{84}$$

where $\Gamma(x)$ is the Gamma function, the constraint $\sum_{k=1}^{K}\theta_k^{(i)} = 1$ holds and $\theta_{i,k} > 0$ for all i and k. Prior distributions for all $\theta^{(i)}$ share the same parameter $\boldsymbol{a}$. The LDA model can be represented by a so-called plate notation which is shown in Fig. 8. The probability of the observed data $\{(i,\alpha)\}$,

$$P(\{(i,\alpha)\}|\boldsymbol{a},\boldsymbol{\phi}) = \prod_i \int \mathrm{d}\theta^{(i)} P(\theta^{(i)}|\boldsymbol{a}) \left[\prod_{\mu=1}^{k_i}\sum_{k=1}^{K} P(\alpha_{i,\mu}|k) P(k|\theta^{(i)})\right], \tag{85}$$

depends on the parameter vectors $\boldsymbol{a}$ and $\boldsymbol{b}$.

In the original formulation of LDA [175], $P(\alpha|k)$ is given by a multinomial distribution parametrized by $\boldsymbol{\phi}$ such that $P(\alpha|k) = \phi_\alpha^{(k)}$ as in the case of pLSA. Some variants of LDA consider $P(\alpha|k)$ following a Dirichlet prior distribution, which is known as a smoothed version of LDA [197]. In this case, $\boldsymbol{b}$ which characterizes the prior distribution of $P(\alpha|k) = \phi_\alpha^{(k)}$ in Eq. (83) corresponds to a M-dimensional parameter of the Dirichlet prior distribution

$$P(\phi^{(k)}|\boldsymbol{b}) = \frac{\Gamma\left(\sum_{\alpha=1}^{M} b_\alpha\right)}{\prod_{\alpha=1}^{M}\Gamma(b_\alpha)} \prod_{\alpha=1}^{M}\left(\phi_\alpha^{(k)}\right)^{b_\alpha-1}. \tag{86}$$

Prior distributions for all $\phi^{(k)}$ share the same parameter $\boldsymbol{b}$.

In order to obtain rating predictions for unobserved user–object pairs, one has to find $P(\{\phi^{(k)}\},\{\theta^{(i)}\}|\{(i\alpha)\},\boldsymbol{a},\boldsymbol{b})$ and use $\{\phi^{(k)}\}$ and $\{\theta^{(i)}\}$ to make personalized predictions for user i. For instance, the predicted score for an unobserved pair (i,α) is given by $\tilde{r}_{i\alpha} = \sum_{k=1}^{K}\phi_\alpha^{(k)}\theta_k^{(i)}$. Since the distribution of $\{\phi^{(k)}\}$, $\{\theta^{(i)}\}$ is in general intractable, one can follow [175] and adopt a variational approach to maximize likelihood—similarly as we did in the case of pLSA. Here we describe the Gibbs sampling for the smoothed LDA as an alternative inference method [189]. The procedures are similar to the Gibbs sampling in Bayesian networks, except that one assigns a latent class for each user–object pair, instead of classes for individual users and objects.

To derive an equation for Gibbs sampling in LDA, we first assign an index μ for each observed user–object pair (i,α) so that k_μ is a latent variable drawn from the multinomial distribution $\theta^{(i_\mu)}$ and α_μ is an object drawn from the multinomial distribution $\phi^{(k_\mu)}$. As shown in [189,198], the inference method is much simplified by assuming a symmetry Dirichlet prior with homogeneous $\boldsymbol{a}$ and $\boldsymbol{b}$, *i.e.* $a_1 = \cdots = a_K := a$ and $b_1 = \cdots = b_M := b$. Then one can show that the conditional probability for an observed user–object pair μ' characterized by $k_{\mu'}$ is given by

$$\begin{aligned} P\left(k_{\mu'} = k|\{k_\mu\}_{-\mu'},\{(i_\mu,\alpha_\mu)\}\right) &\propto P\left(\alpha_{\mu'}|k_{\mu'} = k,\{k_\mu\}_{-\mu'},\{(i_\mu,\alpha_\mu)\}_{-\mu'}\right) P\left(k_{\mu'} = k|\{k_\mu\}_{-\mu'}\right) \\ &\propto \frac{n^{(k)}_{-\mu',\alpha_{\mu'}} + b}{n^{(k)}_{-\mu'} + Mb} \frac{n^{(i_{\mu'})}_{-\mu',k} + a}{n^{(i_{\mu'})}_{-\mu'} + Ka} \end{aligned} \tag{87}$$

where all $n_{-\mu'}$'s are evaluated in the absence of pair μ': $n^{(k)}_{-\mu'}$ is the number of observed pairs characterized by latent class k, $n^{(k)}_{-\mu',\alpha}$ is the number of observed pairs of object α characterized by latent class k, $n^{(i)}_{-\mu'}$ is the number of observed pairs of user i (degree of i) and $n^{(i)}_{-\mu',k}$ is the number of observed pairs with user i characterized by latent class k.

The Gibbs sampling process runs as follows. We first start with a random assignment of latent class to each observed user–object pair and successively pick a random user–object pair to update its latent class according to Eq. (87). This corresponds to a shift of the system state from one to another; $n^{(k)}$, $n^{(k)}_{\alpha_{\mu'}}$, $n^{(i_{\mu'})}$ and $n^{(i_{\mu'})}_k$ are updated after each new assignment of the latent class. After a sufficient number of iterations, one can sample $\phi^{(k)}$ and $\theta^{(i)}$ at a regular time interval

$$\phi^{(k)}_\alpha = \frac{n^{(k)}_\alpha + b}{n^{(k)} + Mb}, \qquad \theta^{(i)}_k = \frac{n^{(i)}_k + a}{n^{(i)} + Ka}. \tag{88}$$

With these samples of $\phi^{(k)}$ and $\theta^{(i)}$, the predicted score for an unobserved user–object pair can be computed as

$$\tilde{r}_{i\alpha} = \sum_t \sum_{k=1}^{K} \phi^{(k)}_\alpha (t_c + tT)\theta^{(i)}_k (t_c + tT), \tag{89}$$

where t_c and T are respectively the convergence time and the sampling time interval. As in the Gibbs sampling of Bayesian clustering, states of $\phi^{(k)}$ and $\theta^{(i)}$ can be stored and use to compute the predicted scores later. When the input data is large, one can distribute the Gibbs sampling to several processors to shorten the computation time [199].

6. Diffusion-based methods

Similarly as the classical PageRank algorithm [154] brought the order to the Internet by analyzing the directed network of links among web pages, one could aim to obtain recommendations using a network representation of the input data with user preferences. The algorithms presented in this section are all based on specific transformations (projections) of the input data to object–object networks. Personalized recommendations for an individual user are then obtained by using this user's past preferences as "sources" in a given network and propagating them to yet unevaluated objects.

6.1. Heat diffusion algorithm (HDiff)

This algorithm is based on projecting the input data on a simple object network characterized by a symmetric adjacency matrix A with elements either one (for similar objects) or zero (for dissimilar objects). It recommends objects to an individual user by a process motivated by heat diffusion: objects liked and disliked by this user are represented as hot and cold spots respectively, and recommendation is made according to the equilibrium "temperature" of the nodes in the networks [200]. The discrete Laplace operator of the network has the form $\mathsf{L} = \mathsf{1}_N - \mathsf{D}^{-1}\mathsf{A}$ where D is the network's diagonal degree matrix with elements $D_{\alpha\beta} = k_\alpha \delta_{\alpha\beta}$. This operator is a discrete analog of the heat diffusion operator $-\nabla^2$ which is well-known in physics. The resulting temperature vector for user i, $\boldsymbol{h}_i$, is the solution of the heat diffusion equation

$$\mathsf{L}\boldsymbol{h}_i = \boldsymbol{f}_i \tag{90}$$

and has both variable part (which we seek) and fixed part. Fixed elements of $\boldsymbol{h}_i$ correspond to objects already evaluated by user i; they are set to 1 (objects liked by the user—they act as heat sources) or 0 (objects disliked by the user—they act as heat sinks). Mathematically this corresponds to the Dirichlet boundary condition. The external flux vector $\boldsymbol{f}_i$ is non-zero only for objects evaluated by user i and allows for fixed values attributed to sources and sinks. Eq. (90) can be solved using the Green's function method and the involved computational cost can be lowered by utilizing various algebraical properties of L [200]. At the same time, it is straightforward to find the equilibrium $\boldsymbol{h}_i$ iteratively by setting the initial temperature vector $\boldsymbol{h}^{(0)}_i$ to contain only the fixed heat sources and sinks and iterate

$$\boldsymbol{h}^{(n+1)}_i = \mathsf{L}'\boldsymbol{h}^{(n)}_i \tag{91}$$

where L'_i is the same as the Laplace operator above except that it keeps elements in $\boldsymbol{h}_i$ corresponding to i's evaluated objects unchanged.

Given this mathematical framework, it is still an open question how exactly to apply it to a given rating matrix R. The procedure adopted in [200] is that the Pearson's correlation coefficient for ratings of objects α and β, $C_{\alpha\beta}$, is compared with a specified threshold C_t and $A_{\alpha\beta} = 1$ when $C_{\alpha\beta} \geq C_t$ and it is zero otherwise. The threshold C_t is set so that the resulting number of links is the same as the number of non-zero entries in $\mathsf{R}^T\mathsf{R}$ (which is equivalent to the number of object pairs co-evaluated by at least one user). The boundary condition for user i is composed of the ratings given by this user to different objects (note that the heat diffusion equations above are not constrained to the binary case like/dislike-hot/cold and can be used to arbitrary-valued ratings).

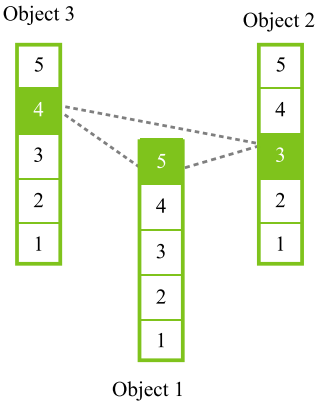

Fig. 9. Graphical representation of the links created by a user who has rated only objects 1 (rating 5), 2 (rating 3) and 3 (rating 4).

6.2. Multilevel spreading algorithm (MultiS)

This algorithm can be applied when ratings $r_{i\alpha}$ are given in a discrete scale.[7] For example, Amazon.com employs a five-star scale where one and five stars correspond to the worst and best rating, respectively. For the sake of simplicity, we assume a five-level rating scale in the rest of this subsection (generalization to a different number of levels is straightforward). As in other diffusion-based methods, the recommendation process starts with the preparation of a particular object–object projection of the rating data. To eliminate the loss of information in the projection, instead of merely creating a link between two objects, in this multilevel spreading algorithms, links are created between ratings given to a pair of objects [201]. As a result we obtain $5^2 = 25$ separate connections (channels) for each object pair. This is illustrated in Fig. 9 on an example of a user who has rated three movies; as a result, three links are created between the given movies. When all data are processed, contributions from all users accumulate and a weighted object–object network is created. Note that splitting the connection between two objects into multiple separate channels aggravates the data sparsity problem and can lead to inferior performance of this algorithm in some cases.

Between a given pair of objects we create multiple links which are conveniently stored in a 5×5 matrix. By representing integer ratings $r_{i\alpha}$ with column vectors in 5 dimensional space (unknown rating with $\boldsymbol{v}_{i\alpha} = (0, 0, 0, 0, 0)^T$, rating $r_{i\alpha} = 1$ with $\boldsymbol{v}_{i\alpha} = (1, 0, 0, 0, 0)^T$, rating $r_{i\alpha} = 2$ with $\boldsymbol{v}_{i\alpha} = (0, 1, 0, 0, 0)^T$, etc.), connection matrix for objects α and β has the form

$$\mathsf{W}_{\alpha\beta} = \sum_{i=1}^{M} \frac{\boldsymbol{v}_{i\alpha}\boldsymbol{v}_{i\beta}^T}{k_i - 1}. \tag{92}$$

Weights of individual users are inversely proportional to their number of evaluations—this aims to compensate the quadratically growing number of links created by an individual user, $k_i(k_i - 1)/2$, hence in total we get a linear relation between user's number of evaluations and the cumulative weight of user's ratings.[8]

Matrices $\mathsf{W}_{\alpha\beta}$ form a symmetric matrix W with dimensions $5N \times 5N$. By the column normalization of W we obtain an asymmetric matrix Ω which describes a diffusion process on the underlying network with the outgoing weights from any node in the graph normalized to unity (if one chooses the row normalization instead, the resulting process is equivalent to heat conduction in the network; for a mathematically-oriented review of flows in networks see [202]). Elements with large weights in Ω represent strong patterns in user ratings (*e.g.*, most of those who rated movie X with 5 gave 3 to movie Y). Similarly as for other diffusion-based methods, we obtain personal recommendations for user i by combining the aggregate matrix Ω with opinions already expressed by i. This opinions are stored in a $5N$-dimensional vector $\boldsymbol{h}_i^{(0)}$ (the first 5 elements correspond to object 1, next 5 elements to object 2, etc.). As in Section 6.1, we seek the stationary solution of the equation

$$\Omega_i \boldsymbol{h}_i = \boldsymbol{h}_i, \tag{93}$$

where Ω_i is the same as Ω except it keeps the elements corresponding to the objects evaluated by user i unchanged. Resulting vectors $\boldsymbol{h}_i^{(n)}$ contain information about objects unrated by user i. This information can be used to obtain rating predictions by the standard weighted average. For example, if for a given object in $\boldsymbol{h}_i$ we obtain the 5-tuple $(0.0, 0.2, 0.4, 0.4, 0.0)^T$, the rating prediction is $0.2 \times 2 + 0.4 \times 3 + 0.4 \times 4 = 3.2$. Numeric tests presented in [201]

[7] It is also possible to apply a binning procedure to continuous-valued ratings and hence transform them into an integer scale but that has not been used in practice yet.

[8] Since the users who have evaluated only one object add no links to the object–object network, the divergence of the weight $1/(k_i - 1)$ at $k_i = 1$ is not an obstacle.

suggest that $\boldsymbol{h}_i^{(1)}$ is a good enough predictor. Sophisticated techniques to avoid multiple iterations in Eq. (93) [201] are hence not fundamental for practical applications of this algorithm. An alternative way is to map each object to several channels with the number of channels being equal to the number of different ratings. So that if a user i has collected an object α with rating 2, her will only connect to $\alpha^{(2)}$. After that, one can directly apply the probabilistic spreading process (see the next subsection) to obtain the similarity and then integrate it into the collaborative filtering framework to obtain better recommendation [203].

6.3. Probabilistic spreading algorithm (ProbS)

This algorithm is suitable for data without explicit ratings, i.e. only the sets of object collected/visited by each user is known. Elements of the rating matrix R are hence $r_{i\alpha} = 1$ (when user i has collected/visited object α) or $r_{i\alpha} = 0$ (otherwise). More explicit preference indicators can be easily mapped to this form, albeit losing information in the process, whereas the converse is not so.

The spreading recommendation algorithm proposed in [103] is based on a projection of the input data (which can be represented by an unweighted user–object network) to an object–object network. In this projection, the weight $W_{\alpha\beta}$ can be considered as the importance of node α with respect to node β and in general it differs from $W_{\beta\alpha}$. A suitable form of $W_{\alpha\beta}$ can be obtained by studying the original bipartite network where a certain amount of a resource (a scalar quantity which reflects, for example, social influence in a recommender system) is assigned to each object node. Since the network is unweighted, the unbiased allocation of the initial resource is split equally among all its neighboring user-nodes. Consequently, resources collected by user-nodes are equally redistributed back to their neighboring object-nodes. This is equivalent to random walk from the initial source nodes to a distance of two in the user–object bipartite graph. An illustration of this resource-allocation process for a simple bipartite network is shown in Fig. 10.

Denoting the initial object resource values as x_α, the two resource-distribution steps can be merged to one and the final resource values read $\tilde{x}_\alpha = \sum_{\beta=1}^{N} W^P_{\alpha\beta} x_\beta$ where

$$W^P_{\alpha\beta} = \frac{1}{k_\beta} \sum_{i=1}^{M} \frac{r_{i\alpha} r_{i\beta}}{k_i}. \tag{94}$$

The superscript P stands for "probabilistic" and serves to distinguish the current spreading process from its modifications that we shall discuss later. Note that the resulting $N \times N$ transition matrix is column normalized with $W^P_{\alpha\beta}$ representing the fraction of the initial β's resource transferred to α. Recommendations for a given user i are obtained by setting the initial resource vector $\boldsymbol{h}^i$ in accordance with the objects the user has already collected, that is, by setting $h^i_\beta = r_{i\beta}$. Recommendation scores of objects are then obtained by

$$\tilde{h}^i_\alpha = \sum_{\beta=1}^{N} \mathsf{W}^P_{\alpha\beta} h^i_\beta.$$

Objects recommended to user i are then selected according to $\tilde{h}^i_\alpha$ (the higher the value, the better).

The original ProbS algorithm has been later improved in various directions. In [112], the authors suggested a heterogeneous distribution of the initial resources among the nodes, $h_\alpha = a_{i\alpha} k^\theta_\alpha$, and showed that when the parameter θ is tuned appropriately, it can help increase the accuracy of recommendations and it also makes the recommendations more personalized. The optimal value of θ is close to -1 (e.g., -0.8 to -1.0 for MovieLens, depending on the size of the selected data set [112,204]), indicating that each item should be assigned more or less the same amount of total initial resource. In [113], it was proposed to construct the transition matrix as $\mathsf{W} + \eta\mathsf{W}^2$ where W is defined by Eq. (94) and η is a free parameter. By effectively removing redundant correlations (the optimal value of η is usually negative), this method succeeded in outperforming ProbS and other derived methods in terms of accuracy and diversity of recommendations. Similar method can also be applied in designing more accurate similarity index for collaborative filtering [205] and link prediction [150]. In [206], they proposed to increase the method's accuracy by giving preference to objects with degree similar to the average degree of objects collected by a given user. In addition, the degree correlation [207], users' tastes [208], user behavior patterns [54] can also be accounted to improve the recommendation accuracy.

Finally, we introduce a preferential diffusion method, which was proposed to enhance the algorithm's ability to find unpopular and niche objects [115]. The basic idea is that at the last step (*i.e.*, diffusion from users to objects), the amount of resource that object α receives is proportional to k^ε_α where $\varepsilon \leq 0$ is a free parameter. When $\varepsilon = 0$, this method is identical to ProbS. It was shown that PD not only provides more accurate recommendations than ProbS but it also generates more diverse and novel recommendations by recommending relevant unpopular items. The authors further compared the intra-similarity of recommended items with that of the whole system. As shown in Fig. 11, they draw a line that divides the parameter space into two phases: In the left region, especially the area corresponding to smaller ε and larger L, PD is like a concave lens that broadens the user's vision, while in the right region, corresponding to larger ε and smaller L, PD likes a convex lens that narrows the user's vision. Of course, we prefer the former case since it embodies the merit of personalization.

Note that the resource-allocation process can also be applied in unipartite networks. Considering a pair of nodes i and j, i can send some resource to j with their common neighbors playing the role of transmitters. In the simplest case, we assume

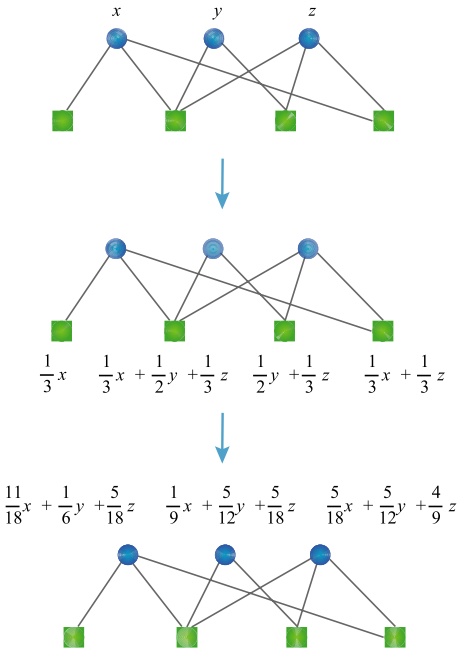

Fig. 10. Illustration of the ProbS's resource-allocation process in a simple bipartite network. The assigned resources first flow from object-nodes (circles) to user-nodes (squares) and then return back to object-nodes.

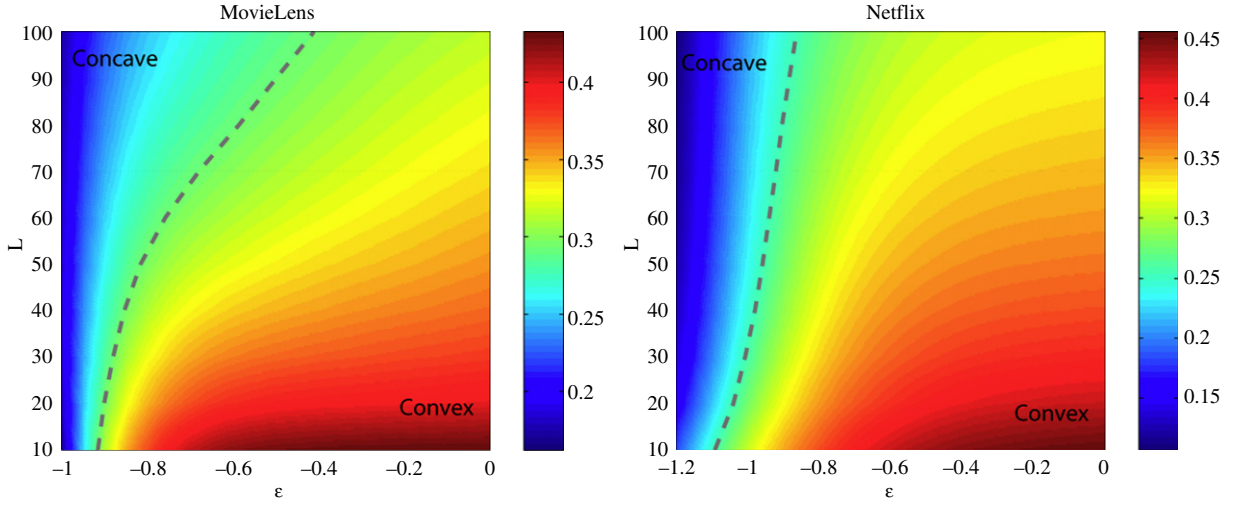

Fig. 11. (Color online) Intra-similarity $I(L)$ (see Eq. (21)) for MovieLens and Netflix. With the parameter combination (ε, L) along this line, the intra-similarity equals the value of the system.

that each transmitter has a unit of resource, and will equally distribute it to all neighbors. This defines a similarity index (called *resource allocation index* [102]) between nodes:

$$s_{ij} = \sum_{l \in \Gamma_i \cap \Gamma_j} \frac{1}{k_l}, \tag{95}$$

where Γ_i denotes the set of i's neighboring nodes. Recent works showed that despite its simplicity, this index performs better than many known local similarity indices in link prediction [102], community detection [209], and the characterization of weighted transportation networks [210].

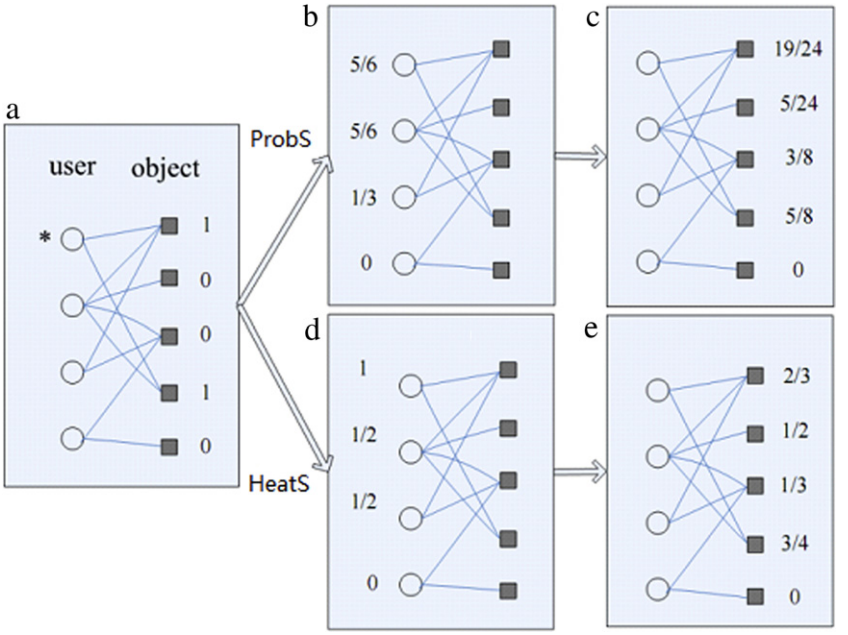

Fig. 12. (Color online) Comparison of ProbS and HeatS, where the target user is marked by a star and the collected objects are of initial resource 1. The final scores after ProbS and HeatS are listed in the right sides of plots (c) and (e).

6.4. Hybrid spreading-relevant algorithms

To answer the need of diversity in algorithm-based recommendation (see Section 2.2 for a discussion of this problem and possible solutions), a hybrid algorithm was proposed in [39] which combines accuracy-focused ProbS with diversity-favoring heat spreading. As in probabilistic spreading, heat spreading works by assigning objects an initial level of "resource" denoted by the vector $\boldsymbol{h}$, and then redistributing it via the transformation $\tilde{\boldsymbol{h}} = \mathrm{W}^H \boldsymbol{h}$. The transition matrix of heat spreading reads

$$W^H_{\alpha\beta} = \frac{1}{k_\alpha} \sum_{i=1}^{M} \frac{r_{i\alpha} r_{i\beta}}{k_i} \tag{96}$$

which, in contrast to W^P obtained with Eq. (94), is row-normalized and corresponds to a heat diffusion process (thus named HeatS) on the given user–object network.

Fig. 12 illustrates the procedures of ProbS and HeatS. According to the final scores, ProbS will recommend the third object to the target user, while HeatS will recommend the second. Generally speaking, HeatS is able to find out unpopular (i.e., of low degree) objects, yet ProbS tends to recommend popular objects and thus lacks diversity and novelty. On the other hand, recommendations obtained by HeatS are too peculiar to be useful.[9] To integrate the advantages from both two algorithms, in [39] they proposed an elegant hybrid of W^H and W^P (named HybridS) in the form

$$W^{H+P}_{\alpha\beta} = \frac{1}{k_\alpha^{1-\lambda} k_\beta^{\lambda}} \sum_{i=1}^{M} \frac{r_{i\alpha} r_{i\beta}}{k_i}, \tag{97}$$

where $\lambda = 0$ gives pure heat spreading and $\lambda = 1$ gives pure probabilistic spreading. As before, the resulting recommendation scores for user i are computed as $\tilde{h}^i_\alpha = \sum_{\beta=1}^{N} W^{H+P}_{\alpha\beta} h^i_\beta$ where the initial resource values are set as $h^i_\beta = r_{i\beta}$. Results shown in [39] show that this combination of two different algorithms allows us not merely to compromise between diversity and accuracy but to simultaneously improve both aspects. By tuning the degree of hybridization, represented by the parameter λ, the algorithms can be tailored to many custom situations and requirements. Note that, the parameter λ is not necessarily to be the same for different users and objects, namely each user i can have her own parameter λ_i or each object α can have its own parameter λ_α, in this way, the algorithmic performance can be further improved [213]. And the initial resource distribution is not necessarily to be homogeneous, and by introducing the heterogeneity, the algorithmic performance can be improved [214].

[9] Compared with the similarity-based methods and ProbS, the AUC value and precision of HeatS are considerably lower. Therefore, using HeatS alone seems not proper. Recent works [211,212] indicate that some weighted version of HeatS could also give highly accurate recommendation.

Similar to the Hybrid spreading algorithm, *B-Rank* combines a random walk process with heat diffusion but it does so for data containing explicit ratings [215]. The transition matrix is introduced in the form

$$P_{\alpha\beta} = \frac{1-\delta_{\alpha\beta}}{n_\alpha} \sum_{i=1}^{M} w_i x_{i\alpha} x_{i\beta} \tag{98}$$

where $x_{i\alpha} = 1$ if user i has rated object α and $x_{i\alpha} = 0$ otherwise, w_i is the weight of user i and n_α is a term which makes the matrix P row-normalized. User weights w_i are set all identical but can be potentially set heterogeneous, for example, to give more weight to reliable users or suppress spammers (these possibilities have not been studied yet). Due to the term $1-\delta_{\alpha\beta}$, $P_{\alpha\alpha} = 0$ and the corresponding random walk on the weighted object–object network is thus non-lazy (there is no possibility to return to the initial node after one step). Using the vector with ratings of user i, $h^i_\alpha = r_{i\alpha}$, object scores corresponding to the forward and backward propagation of $\boldsymbol{h}^i$ are computed as $\boldsymbol{F}^i = \mathsf{P}^T\boldsymbol{h}^i$ and $\boldsymbol{B}^i = \mathsf{P}\boldsymbol{h}^i$ (forward and backward propagation correspond to random walk and heat diffusion, respectively—for details see [202]). The final score of object α is obtained as $f^i_\alpha = F^i_\alpha B^i_\alpha$ (the higher, the better). Note that this algorithm does not aim at predicting missing scores (hence the traditional measures of recommender systems, MAE and RMSE cannot be applied to evaluate it). Instead, it provides a personalized ranking (hence the name, 'B-Rank') of objects for each user.

The above-mentioned diffusion processes can be applied in computing the similarity between users or items, and then integrated into the similarity-based methods. Liu et al. [206] defined the similarity between users according to ProbS,[10] and showed that based on the routine collaborative filtering algorithm, the proposed similarity index improved the algorithmic performance compared with the Pearson similarity index. Pan et al. [216] applied the HybridS process to define similarity, which outperforms the cosine similarity under the framework of collaborative filtering.

7. Social filtering

Recommendations made by a recommender system are often less appreciated than those coming from our friends [217] and social influences may play a more important role than similarity of past activities [218,219]. In addition, accuracy of recommendation can be improved by analyzing social relationships, such as coauthorships in academic literature recommendation [220] and friendships and memberships in product recommendation [221]. Many real systems, such as *Delicious.com* and *Facebook.com*, allow users to recommend objects to their friends. Similarly, users can subscribe to articles from selected bloggers in blogging sites (*Twitter.com* and others) as well as to news alerts from information dissemination systems (*Elesvier.com* and others). In this chapter, we will first present empirical evidences that demonstrate the presence of social influences on information filtering. Then we will introduce two basic ways social filtering are employed in recommendation: by quantifying and utilizing trust relationships between users and by using the opinion from "taste mates" to select the content to be recommended.

7.1. Social influences on recommendations

Social influences, also called the word-of-mouth effects in the literature, are known to be crucial to many sociometric processes, such as decisions making, opinions spreading and the propagation of innovation and fashion [222–225]. Scientists have been aware of the commercial values of social influences for a long time [226], yet large-scale applications for commercial purpose only emerged when the Internet era began. Besides, the availability and the great variety of data provide us good opportunities to quantitatively understand social influences [227]. This section focuses on the social recommendations, whose effects can be roughly divided into two classes: one is on users' prior expectations, leading to the increase of sales; another is on users' posterior evaluations, resulting in the enhancement of the user loyalty.

Positive effects of social recommendations on prior expectation have already been demonstrated in a number of real examples. They are found in a wide range of systems, including product reviews [228,229], e-mails [230], blogs [231] and microblogs [232]. In [233], the authors studied the effects of social influences on purchase preference: users of an e-commerce system were given the option to recommend an item to their friends through e-mails after purchase. The first person to purchase the same item through a referral link from e-mails got a 10% discount, and when this happens, the recommender will receive a 10% credit. As shown in Fig. 13, the purchase probability for a DVD grows remarkably with the increasing number of received recommendations from friends on this DVD. There is a saturation at about 10 recommendations, after which the purchase probability does not increase any more. In other examples, social influences can be much more complicated. In [233], they reported a similar experiment with book sales where in contrast to the common sense, recommendations had little or even negative effect on the purchase probability. Social influences may also vary across topics and items: [234] showed that different tags and topics spread on Twitter differently and [235] found that strength and direction of social influence are topic-dependent. A recent work shows that the mutual interaction between opinions of past viewers and potential future viewers leads to a complex dynamics that agrees qualitatively with movie popularity behavior seen in real systems [236].

[10] Similar to Eq. (94), but the spreading process starts from user side and ends at user sides.

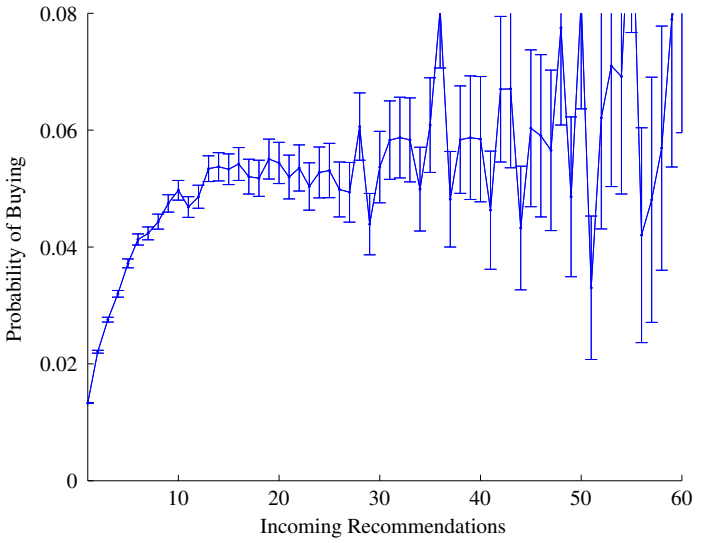

Fig. 13. Probability of buying a DVD given a number of incoming recommendations.
Source: Taken from [233].

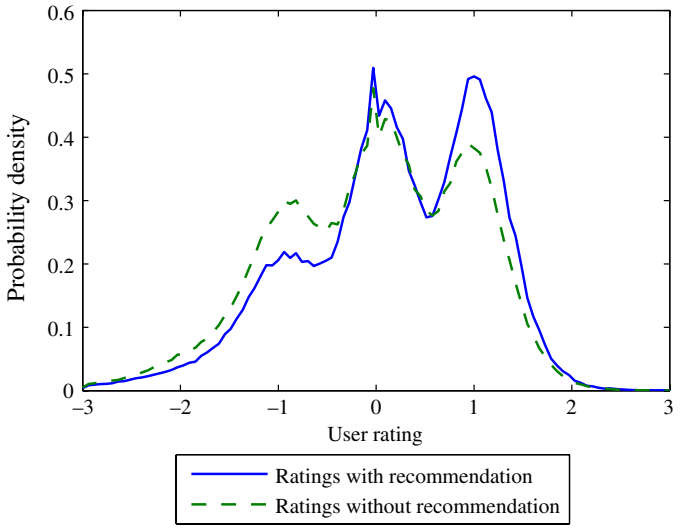

Fig. 14. The probability distributions of ratings, posted with (solid line) or without (dashed line) a word-of-mouth recommendation in Douban.
Source: Taken from [237].

In comparison, the issue about how social recommendations affect users' posterior evaluation received less attention. [237] empirically analyzed two web sites, *Douban.com* and *Goodreads.com*, where millions of users rate books, music and movies, and share their ratings and reviews with friends and followers. On these social network sites, the phenomenon that users recommend favorites to friends and followers plays an important role in shaping users' behaviors and collections. Fig. 14 compares the probability distribution of ratings on items in Douban with and without recommendations (the result is very similar in Goodreads). This demonstrates that an individual is more likely to give a high rating to an item with word-of-mouth recommendations, compared with items without recommendations. Zhu et al. [329] reported an experiment showing that people's choices could be reversed with high probability (20%) under social influence from others' choices.

There are also indirect evidences about the positive social influences, for example, in Twitter, statistically a tweet will spread to about 10^3 users if it gets retweeted [238], and in Taobao,[11] the communication between buyers is a fundamental driving force for purchasing activity [239].

[11] Taobao.com is a Chinese consumer marketplace that is the world's largest e-commerce website.

Many ingredients could result in positive social influences in online recommendation. Firstly, the word-of-mouth influences and role-model effects from social mates are very strong in offline society, for example, an experiment in Nepal [240] shows that on average the probability of a woman to use a novel menstrual cup *Take-Up* will increase by 18.6% if one more of her friends has used Take-Up. Secondly, the friendship network and interest-based network are strongly correlated with each other [241], and friends tend to visit the same items and vote them with similar ratings [242].

7.2. Trust-aware recommender algorithms

It is the basic paradigm of recommender systems that when computing recommendations for an individual user, evaluations of the others are not weighed equally and preference is given to those who have similar rating patterns as the given user. This approach neglects an important facet of the evaluation process: it is not only personal tastes but also social relationships and the quality of evaluations that differs from one user to another. To make a better use of social relationships, various recommendation algorithms relying explicitly on trust or user reputation [243,244] have been developed and applied by many commercial web sites such as eBay.com [245]. Trust can be used instead of user similarity [246], in combination with collaborative filtering to help deal with data sparsity and the cold start problem [247–249], or it can help to further filter recommendations by prioritizing those approved by trusted sources (see [250] for a review). The use of reputation in recommendation is further supported by the evidence that trust correlates with user similarity [251], meaning that by introducing trust we are unlikely to conflict with users' interests and preferences. As noted in [252], even an imperfect reputation system may be beneficial as (i) it provides an incentive for good behaviors, (ii) imposes costs on participants to get established, and (iii) swiftly reacts to bad behavior. The use of trust and reputation has also its drawbacks, which includes: (i) time consuming computation, (ii) low incentives for users to provide the required feedback, (iii) privacy concern for data of trust relationship, and (iv) low availability of trust data sets for tests of algorithms. However, without algorithms for trust and reputation, online transactions would be dramatically affected, if not halted.

The difference between reputation and trust is that while the former characterizes general beliefs about a user's trustworthiness, the latter concept is personal and relates to a user's willingness to rely on the actions of a given user. The terms local and global trust metric are sometimes used instead of trust and reputation, respectively. To establish trust or reputation, one may let the involved participants to rate each other and use these evaluations to derive trust or reputation scores [244]. When mutual user evaluations are given, an initial seed of trusted users can be used to find all the trustworthy users as in the classical Advogato trust metric (see http://www.advogato.org/trust-metric.html). Other popular trust metrics, such as PageRank [154] and Eigentrust [253], use the spreading activation approach by which nodes are initially loaded and then propagate their load within the network [254] (diffusion-based recommendation methods are presented in Section 6 also use this approach). When trust relations between the users are weighted (with trust values 1 and 0 representing total trust and distrust, respectively), the plain trust propagation can be shown to be insufficient—the Appleseed algorithm solves this problem by creating virtual trust edges for backward propagation [255].

It is a great disadvantage of reputation-aware recommender systems that they usually require substantial input on the user side to evaluate trust and reputation. Some trust-aware reputation algorithms hence tried not to rely on explicit evaluations of other users. In [256], they propose to detect noisy ratings by comparing the actual ratings with the predicted ratings obtained by a recommendation algorithm, with data only from a set of implicitly trusted users. This "reputation of ratings" can in turn be used to build reputation of users [250]. A different approach was proposed by [257] where the authors use the information contained in social relationships between the users (which may stem from users' family or friendship relations), the transitivity of trust (as in [258]), and the propagation of users' queries in social networks. When computing recommendations for a specific user i, the greatest weight is hence given to the users who can be connected with user i in the social network by a short path with high trust values along its edges. The proposed system is shown to assign correctly trust values and self-organizes to a state producing highly accurate recommendations (when compared to a simple benchmark strategy when one of the recommendations from peers is chosen at random).

7.3. Adaptive social recommendation models

While the above described trust-aware systems make use of existing social relationships, adaptive social recommendation models build a network of users based on their evaluations. In [259], the authors proposed a model where the recommended items spread over the network similarly as an epidemics [64,260] or rumor [10,261] spreads in a society. Simultaneously with this spreading, the network of users evolves and adapts to best capture users' similarities. This epidemic-like spreading of a successful item is of particular importance in the case when individual items swiftly loose their relevance [262] – as it is the case for news stories, for example – because it combines personalization with the speed of access. It is very different from the currently popular services such as *digg.com* and *reddit.com* which still rely on centralized distribution of items where only those of very general interest can become popular and be accessed by many. For a recent review of approaches to news recommendation see [263].

Here we describe briefly the model introduced in [259]. In this model, users either "approve" or "disapprove" the consumed items. Each user i has S sources (*i.e.*, S other users from whom i receives news) and thus the system can be described by a directed network with a constrained node in-degree S. When a news is approved by user i, it is added to the

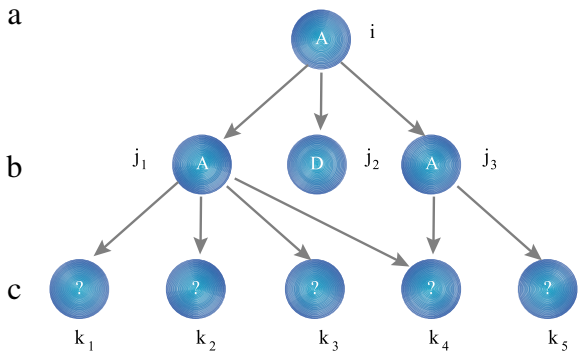

Fig. 15. Illustration of the news propagation in an adaptive network model [259]. User i added a new news, which is automatically considered as approved (A) and sent to users j_1, j_2, j_3 who are i's followers. While user j_2 dislikes (D) the news, users j_1 and j_3 approve it and pass it further to their followers $k_1, \ldots, k_5$ who have not evaluated the news yet (which is denoted with question marks). User k_4 receives the news from the authorities j_1 and j_3, yielding the news's recommendation score $s_{j_1k_4} + s_{j_3k_4}$. At the same time, user k_5 receives the news only from the authority j_3 and hence for this user, the recommendation score is only $s_{j_3k_5}$.

recommendation lists to all i's followers (*i.e.*, all users who have i as one of their sources). This spreading process is illustrated in Fig. 15. Similarity between users i and j is defined according to the agreement of their past evaluations,

$$s_{ij} = \frac{n_A}{n_A + n_D}\left(1 - \frac{1}{\sqrt{n_A + n_D}}\right) \tag{99}$$

where n_A and n_D denote the number of news for which evaluations of i and j agree and disagree evaluations, respectively. The term $1/\sqrt{n_A + n_D}$ aims at penalizing user pairs with little overlap of evaluated news as they may seem to be a great match for each other simply because agreeing in evaluations of a few news. To summarize, items at the top of a user's recommendation list are probably recommended by multiple sources of this user or, at least, by a source whose similarity with this user is high.

Apart from using user similarity in the recommendation process (the recommendation score of item α for user i is given by the similarity between i and the sources of i who approved this news), it is also crucial for updating the source–follower network. This updating aims at maximizing the similarity of each user with their sources. As shown in [259] by agent-based simulations, the system can evolve from a random initial state to a highly organized state where taste mates are connected and news spread effectively. The original network can be improved by rewiring, through ineffective random replacements or computationally demanding global optimization. One can also combine simple greedy optimization with random assignment [264], repeated trials [265] or exploration of the directed user network both in the direction of followers and sources [266]. Robustness of this recommendation approach can be enhanced by introducing the user reputation [264]. This adaptive evolution of the network of user–user interactions can be used to explain the widely observed scale-free leadership structure in social recommender systems [267], and recent analysis suggested that users could get better information by selecting proper leaders in social sharing websites [268].

8. Meta approaches

Input data for recommendation can be extended far behind the traditional user–item-rating scheme. In this section, we briefly review the so-called meta approaches where additional information of various kind (tags assigned to items by users or time stamps of past evaluations) enters the recommendation process or simply several recommendation methods are combined together (either in an iterative self-evaluating way or by forming a hybrid algorithm). As possibilities for extensions are relatively easy to find, this direction has seen high activity over the past years.

8.1. Tag-aware methods

In the past decade, the advent of *Web 2.0* and its affiliated applications brought us a new paradigm to facilitate the user-generated creation on the Internet. One such example are user-driven platforms that allow users to store resources (bookmarks, images, documents, and others) and to associate them with personalized words, so-called *tags*. The resulting ternary user–object–tags structure, so-called *folksonomy*, represents a rich data structure that is of interest for computer scientists, sociologists, linguists, and also physicists [94]. When viewed from the perspective of an individual user, these tags constitute a personalized folksonomy [269] where tags, although only simple words, contain highly abstracted yet personalized information. Different from other kinds of meta-data (such as profile, attributes, and content), tags are not predefined by domain experts or administrators. This approach has the advantage of being scalable and requiring no specific skills, hence allowing every individual to participate and contribute. Despite the lack of imposed organization, shared

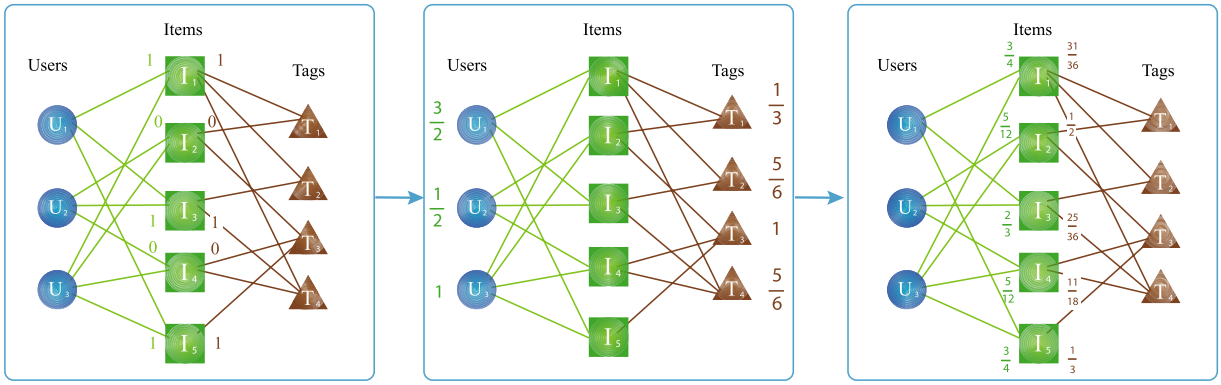

Fig. 16. (Color online) Illustration of a user–item–tag tripartite graph consists of three users, five items and four tags, as well as the recommendation process described in [286]. The tripartite graph is decomposed to user–item (black links) and item–tag (red links) bipartite graphs connected by items. This is how the scoring process works for a given target user U_1. (a) First, highlight the items, I_1, I_3, I_5, collected by the target user U_1 and mark them with unit resource. In the depicted case we have: $f_{I_1} = f_{I_3} = f_{I_5} = 1$, and $f_{I_2} = f_{I_4} = 0$. (b) Second, distribute the resources from items to their corresponding users and tags, respectively: $f_{U_3} = f_{I_1} \times \frac{1}{2} + f_{I_2} \times \frac{1}{2} + f_{I_5} \times \frac{1}{2} = 1 \times \frac{1}{2} + 0 + 1 \times \frac{1}{2} = 1$ and $f_{T_4} = f_{I_1} \times \frac{1}{3} + f_{I_3} \times \frac{1}{2} + f_{I_4} \times \frac{1}{2} = 1 \times \frac{1}{3} + 1 \times \frac{1}{2} + 0 = \frac{5}{6}$; (c) Finally, redistribute the resources from users and tags to their neighboring items: $f^{p}_{I_4} = f_{U_2} \times \frac{1}{3} + f_{U_3} \times \frac{1}{4} = \frac{1}{2} \times \frac{1}{3} + 1 \times \frac{1}{4} = \frac{5}{12}$ and $f^{pt}_{I_4} = f_{T_3} \times \frac{1}{3} + f_{T_4} \times \frac{1}{3} = 1 \times \frac{1}{3} + \frac{5}{6} \times \frac{1}{3} = \frac{11}{18}$.

vocabularies were shown to emerge in folksonomies [270], making them increasingly accessible to advanced information filtering techniques. Tags therefore represent a simple yet promising tool to provide reasonable recommendations and solve some outstanding problems in recommender systems, *e.g.* the cold-start problem [271]. The social impact [272] and dynamical properties of folksonomies [273,274] are expected to be applied to obtain trustworthy and real-time recommendations in tagging systems. In addition, the hypergraph [94] theory is considered to fully utilize the complete network structure of tagging platforms without using any hybrid methods and losing any information, which gives promises for generally more reliable recommendations. At the same time, there are also drawbacks caused by the freedom of tags: *e.g.* polysemy, synonymy, and ambiguity [275]. To alleviate these problems, advanced methods such as tag hierarchical clustering [276], introduction of ontologies [277] and recommendation of tags [278] were proposed.

Unlike being used as a traditional filter, researches tend to apply more sophisticated theories and methods (*e.g.*, social impact) in designing tag-aware recommendation algorithms. The *FolkRank* [279], a modified PageRank algorithm, was proposed to rank tags in folksonomies by assuming that important tags are given by important users. Due to the success of collaborative filtering, many works were devoted to using tags to measure similarity among users or objects, and then fuse with the standard memory-based collaborative filtering framework [280–282]. In [84], the present tag-aware algorithms are classified as follows:

1. Topic-based models. They implement probability-based methods such as pLSA and LDA (see Sections 5.3 and 5.4) to extract latent topics from the available tags in the user or object space and then produce recommendations using classical probability-based models [283–285].
2. Network-based models. They implement graph theory-based methods such as ProbS (see Section 6.3) to represent tags as nodes in a tripartite user–object–tag network and apply a diffusion process to generate recommendations [286] (see Fig. 16).
3. Tensor-based models. They implement tensor factorization [287] to reduce the ternary relation into low-rank feature matrices, alleviate the sparsity problem in large-scale data sets and ultimately provide personalized recommendations [288,289].

8.2. Time-aware methods

Nowadays, huge quantities of information emerge every second. We receive news from various media, such as newspapers, TV programs, websites, etc. Due to its timeliness and in particular its convenience, more and more people prefer to read news online (*e.g.*, using RSS feeds) instead of from traditional media like newspapers. However, given the enormous amount of online news, one challenging issue is the novelty of news recommendation, *i.e.*, how to appropriately and efficiently recommend news to readers, matching their reading preferences as much as possible. The analysis of data from a popular platform for sharing news stories, Digg.com, by Wu and Huberman [290] shows that the novelty of news half there decays in a very short period. Another typical instance is the communication system of e-commerce websites, which require real-time feedback among various agents [291]. Such information updates so frequently that it is impossible for individuals to evaluate each, or even read them in time. Consequently, an urgent question emerges: how to automatically filter out the irrelevant information, receive timely news and perform immediate yet appropriate response? One promising solution lies in time-aware recommender systems, which can hopefully help to address aforementioned issues. Collaborative filtering (see Section 4), as the most widely adopted method in recommender systems, is the first one to be considered. Following

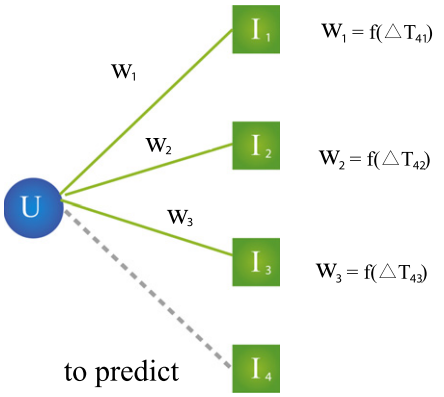

Fig. 17. (Color online) Illustration of time based recommendation where the target user has collected three items, I_1, I_2 and I_3, which are then used to predict the newest item I_4. f is the decay function to weight the time difference between I_4 and other items that were collected before.

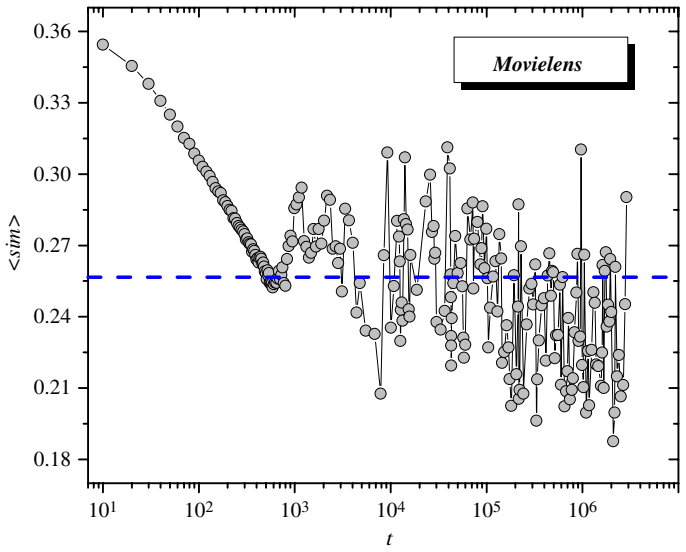

Fig. 18. (Color online) Illustration of users' interests changing with time in the *MovieLens* data. Shifts of user interests are represented by the average similarity among item pairs of the target user within an observed time t (shown with gray circles). The dashed line represents the average similarity of all users.

the classical collaborative filtering framework, most of the related work focuses on designing time factors to suppress old evaluations or objects. Generally, such kind of weight is expressed by various decay functions (see Fig. 17), which suggests that user interests in a single topic would decay with time (see Fig. 18). Ding and Li [292] weighed different items by putting smaller weights on older ones. Similar methods used the time factor to adaptively choose temporal neighborhoods and then obtain recommendations via the refined neighbors [293–295]. Another kind of attempts consider decaying time to weigh user–item binary edges in bipartite networks. Liu and Deng [296] hypothesized the time effect decayed in a exponential manner, which could also been found in other empirical studies [290] and models [297]. A broad picture of collaborative filtering with time can be found in a recent Ph.D. thesis [298].

Another important issue in recommender systems is that of *novelty*. Although an item with the highest recommendation score is the most possible candidate for a target user, it may fail to be picked due to the diversity of human tastes. In such cases, these items should not always occupy the recommendation list and be recommended over and over again. Therefore, temporal diversity [299] becomes crucial in designing time-based algorithms. Xiang et al. divided user interests into two general categories: longer-term and short-term [44]. Longer-term interests govern the essential preferences of users and would not easily change over time. By contrast, short-term interests are more likely to be effected by social environment (Fig. 18 shows such difference). By identifying and making use of the differences between them using a time factor, in [44] they successfully provided more reliable yet interesting recommendations, which can also be found in relative works [300].

Besides recommendation, time also plays an important role in various fields, such as network growth [262,301], identification of original scientific contributions [302,303], selecting the backbone structure of citation networks [304], and

aging effects in synchronization [305]. However, how to appropriately use the time information to discover the underlying dynamics of user preferences and help us master the current information era still remains an important research challenge.

8.3. Iterative refinement

Iteratively solved self-consistent equations are widely applied in characterizing structural and functional features of nodes and/or edges in networked systems. Their solution is usually used to describe a stable distribution of a certain quantity in a system consisting of many interacting individuals, where the amount of this quantity assigned to individual i is affected by both the interacting rules and the amounts of this quantity assigned to other individuals interacting with i. In a directed network, the significance of a node is not only determined by its attributes (if applicable), but also contributed by the significance of its downstream nodes. For example, the classical set of self-consistent equations for PageRank value $G(i)$ of the web page i has the form [154]

$$G(i) = c + (1 - c) \sum_{j \in \Gamma_i} \frac{G(j)}{k_j^{out}}, \tag{100}$$

where j runs over all the web pages that point to i, c is the return probability accounting for the random browsing, and k_j^{out} is j's out-degree. This set of equations represents a particular Markovian process on a network and can be successfully solved using an iterative approach thanks to the fast convergence to a stationary solution observed in most cases [306]. Besides web pages, similar self-consistent equations are also successfully applied in ranking people [307], genes [308], and so on. If instead of referring to nodes, the individuals refer to pairs of nodes, this approach can be used to quantify node similarity [144,309]. In more complicated situations, the individuals can be users and items, or scientists and publications, and similar iterative equations embodied in bipartite networks can be applied in building quantitative reputation systems, namely simultaneously estimate people's reputation and objects' quality [310–315].

Besides the above-mentioned iterative equations, a closely related framework is that of the so-called *self-consistent refinement* [316,317]. In the link prediction and personalized recommendation, the known information is the adjacency matrix representing a unipartite or a bipartite network and the task is to estimate the likelihoods of link existence for currently zero elements of the adjacency matrix. For recommender systems with ratings, the algorithms need to predict unknown ratings according to the rating matrix. Denoting R the known matrix and $\tilde{\mathsf{R}}$ the predicted matrix (*i.e.*, output), the procedure of many algorithms can be written in a generic form

$$\tilde{\mathsf{R}} = \mathfrak{D}(\mathsf{R}), \tag{101}$$

where $\mathfrak{D}$ is a matrix operator.[12] Denoting the initial configuration as $\mathsf{R}^{(0)}$ and the initial time step $k = 0$, a generic framework of self-consistent refinement reads [317]: (i) Implement the operation $\mathfrak{D}(\mathsf{R}^{(k)})$; (ii) Set the elements of $\mathsf{R}^{(k+1)}$ as

$$R_{i\alpha}^{(k+1)} = \begin{cases} \mathfrak{D}(\mathsf{R}^{(k)})_{i\alpha} & \text{when } R_{i\alpha}^{(0)} = 0, \\ R_{i\alpha} & \text{when } R_{i\alpha}^{(0)} \neq 0. \end{cases} \tag{102}$$

Then, set $k = k + 1$. (iii) Repeat (i) and (ii) until the difference between $\mathsf{R}^{(k)}$ and $\mathsf{R}^{(k-1)}$ is smaller than a given terminating threshold.

Considering the matrix series $\mathsf{R}^{(0)}, \mathsf{R}^{(1)}, \ldots, \mathsf{R}^{(T)}$ (T denotes the last time step) as a certain dynamics driven by the operator $\mathfrak{D}$, all the elements corresponding to the known items (*i.e.*, $R_{i\alpha}^{(0)} \neq 0$) can be treated as the *boundary conditions* expressing to the known and fixed information.[13] If $\tilde{\mathsf{R}}$ is an ideal prediction, it should satisfy the self-consistent condition $\tilde{\mathsf{R}} = \mathfrak{D}(\tilde{\mathsf{R}})$. However, this equation is not hold for most known algorithms. In contrast, the convergent matrix $\mathsf{R}^{(T)}$ is self-consistent.

As shown in [317], applying the self-consistent framework can lead to great improvements compared with non-iterative methods employing Eq. (101). We next show a simple example for similarity-based recommendation. Taking into account the different evaluation scales of different users [318], we subtract the corresponding user average from each evaluated entry in the rating matrix R and get a new matrix R'. The predicted rating is calculated by using a weighted average, as:

$$\tilde{R}_{i\alpha} = \frac{\sum_{\beta} \Omega_{\alpha\beta} \cdot R'_{i\beta}}{\sum_{\beta} |\Omega_{\alpha\beta}|}, \tag{103}$$

where the similarity between items α and β is defined by Eq. (34). As shown in Fig. 19, experiment on MovieLens verifies the advantages of the iterative refinement, where the original similarity-based algorithm corresponds to the first iteration step.

---

12 See [317] on how to use this generic form to represent the well-known similarity-based and spectrum-based algorithms for rating prediction.

13 This is the essential difference between the self-consistent refinement and the above-mentioned iterative equations like PageRank, since in the latter case, every matrix element is free to be changed.

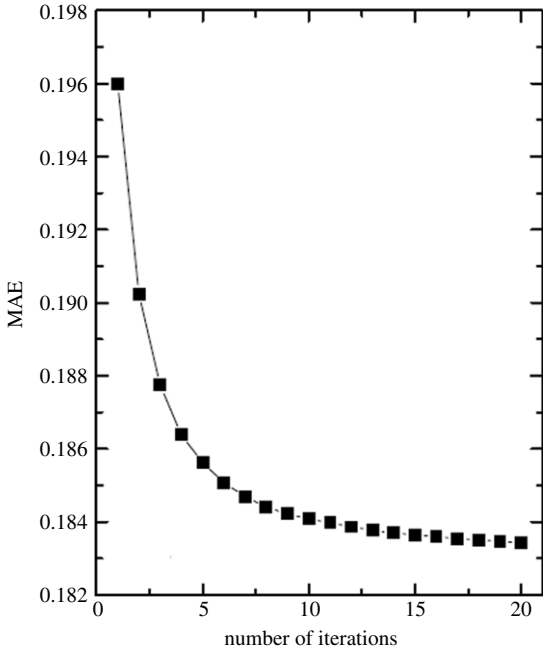

Fig. 19. Prediction error (MAE) versus iterative step for MovieLens data. We use the *MovieLens* data that consists of $N = 3020$ users, $M = 1809$ movies, and 2.24×10^5 discrete ratings from 1 to 5. All the ratings are sorted according to their time stamps with 90% earlier ratings as the training set, and the remaining ratings (with later time stamps) as the testing set.

8.4. Hybrid algorithms

Even for a good recommender algorithms it is difficult to address diverse needs of its heterogeneous users. Hybrid methods overcome this problem by aptly combining recommendations obtained by different methods [319,320]. Hybridization is hence often used in practical implementations of recommender systems, even in very early ones [164]. One of the most important applications of hybrid recommendation algorithms is to solve the cold-start problem, by combining collaborative and content data in such a way that even a new object that has never been rated before can be recommended [321] (similarly, a new user who has never rated anything can receive some recommendations). Since hybrid algorithms combine different approaches to recommendation, they also have the potential to improve the diversity of recommendations [39]. The following classification of hybridization methods is taken from [15]:

1. implement collaborative and content-based methods separately and combine their predictions,
2. incorporate some content-based characteristics into a collaborative approach,
3. incorporate some collaborative characteristics into a content-based approach, and
4. construct a general unifying model that incorporates both content-based and collaborative characteristics.

Now we can extend class 4 to include also unifying models that incorporate two or more collaborative methods (see Section 6.4).

Hybrid methods range from simple, such as using a linear combination of ratings obtained by different methods [162], to very complex, such as employing Markov chain Monte Carlo methods [322] to model combined collaborative and content data. Recent Netflix prize (see Section 2.1) has provoked interest in sophisticated methods for combined predictions (also called *ensemble learning* or *blending*) which were shown to be very successful in lowering the error of predictions [323]. The main idea of blending is that the prediction vectors of F distinct recommendation methods, denoted as $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_F$, are combined by a function $\Omega : \mathbb{R}^F \to \mathbb{R}$ so that the prediction error on a test set (evaluated by RMSE, for example) is minimized. The optimal weighting function Ω is obtained by linear regression, neural networks, or bagging predictors [324,325]. For details and evaluation of different blending schemes on the extensive data set provided by Netflix for the competition see [323]. Some challenges can also be partially solved by hybridization, for example, the link prediction algorithm can be used to generate artificial links that could eventually improve the recommendation in very sparse data [125,326].

Table 5
Data sets used for evaluation of recommendation methods. Data density is defined as the ratio of the available ratings to the maximum possible number of ratings; k_U and k_O denote the average user and object degree, respectively.

Data set	Users	Objects	Ratings	Density (%)	k_U	k_O
Movielens	6040	3,706	1000,209	4.5	166	270
Netflix	8000	17,148	1632,172	1.2	204	95

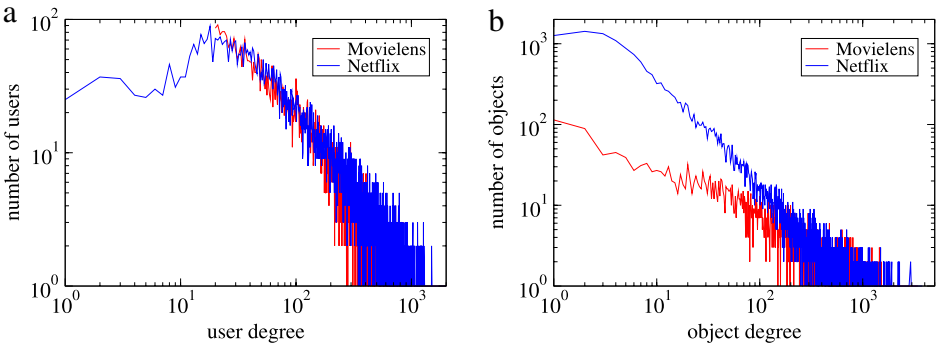

Fig. 20. User and object degree distribution for the two data sets used to evaluate recommendation methods.

9. Performance evaluation

In this section we briefly compare performance of individual algorithms presented in this review. To test the algorithms, we use two standard data sets: Movielens 1M[14] and Netflix.[15] The Movielens data set, which contains ratings from 6040 users to 3706 movies (corresponding to the rating density of $4.5 \cdot 10^{-2}$), is used in its original form. Our subset of the original Netflix data set was created by randomly selecting 8000 users from the original data set released by Netflix for the Netflix Prize and keeping all their evaluations (see 2.1 for details on the competition). In this way, a data set with 17,148 objects (DVDs rented by the company to its users) and 1632,172 ratings in the integer scale from one to five (corresponding to the rating density of $1.2 \cdot 10^{-2}$) was created. Both data sets use the integer rating scale from one to five. For methods requiring no explicit ratings (binary data), we assume that all ratings greater or equal than three represent objects *liked* by the users and hence constitute a corresponding user–object link (if the rating is less than three, no link is formed). As a result, there are 1387,039 and 836,478 links in the Netflix and Movielens data set, respectively. Table 5 summarizes basic properties of the data sets.

Our two data sets differ not only by their density and user/object ratio—histograms of user and object degrees in Fig. 20 reveal further differences. Firstly, Movielens data were originally prepared in such a way that all users rated at least twenty movies. This constraint has not been applied to the Netflix data set, resulting in a considerable number of users with only little data on their past preferences available. Secondly, Netflix data also contains a large portion of movies that have been rated only a few times (this probably reflects the fact that the company rents a wide variety of DVDs, many of which are of interest for only a small part of the customers). Unsurprisingly, all degree distributions are broad and right-skewed, as similar to many other social systems [26].

To test a recommendation method, we employ the standard approach. First, a randomly selected small part of the input data is moved into a so-called probe. In our case, the probe contains 10% of ratings present in the input data set. The remaining 90% of the data is then given to the recommendation method and are used to estimate the missing ratings. The estimated missing ratings are then compared with the true ratings present in the probe set. This comparison is done by means of root mean square error (RMSE) and mean absolute error (MAE) in the case of data with explicit ratings and by means of precision, recall and the average relative rank in the case of data without explicit ratings (see Section 3.4 for a detailed description of these performance metrics). For precision and recall, we take into account top 100 places of each user's recommendation list. To eliminate possible effects of the probe selection, we repeat the procedure for ten independent randomly selected probe sets and present the averaged results.

Method *overall average* is used only as a benchmark; it uses the average rating in the input data as estimate for every user–object pair. For similarity-based methods, we employ the Pearson correlation coefficient which slightly outperforms cosine similarity in terms of RMSE and MAE. For *SVD*, we used parameter values $D = 20$, $\eta = 0.001$ and $\lambda = 0.1$ that result in favorable performance (see Section sec:SVD for the meaning of these parameters). Note that a better founded approach would be to learn "optimal" values of these parameters from the data itself. This can be achieved by choosing $\{D, \eta, \lambda\}$

14 This data set can be obtained at http://www.grouplens.org/node/73.
15 This data set can be obtained at www.netflixprize.com.

Table 6
Performance of algorithms for data with explicit ratings (averaged over 10 realizations).

Method	Movielens		Netflix	
	RMSE	MAE	RMSE	MAE
Overall average	1.12	0.93	1.09	0.92
Object average	0.98	0.78	1.02	0.82
User average	1.04	0.83	1.00	0.80
Multilevel spreading	0.94	0.75	0.97	0.77
User similarity	0.91	0.72	0.93	0.72
Object similarity	0.89	0.70	0.96	0.74
SVD	0.85	0.67	0.87	0.68
Slope one	0.91	0.71	0.93	0.73
Slope one weighted	0.90	0.71	0.93	0.73

Table 7
Performance of algorithms for data without explicit ratings (averaged over 10 realizations).

Method	Movielens			Netflix		
	P_{100}	R_{100}	Rank	P_{100}	R_{100}	Rank
Global rank	0.039	0.311	0.143	0.041	0.272	0.066
Bayesian clustering	0.028	0.276	0.137	0.045	0.262	0.069
pLSA	0.071	0.575	0.090	0.071	0.456	0.050
LDA	0.081	0.543	0.093	0.081	0.439	0.048
ProbS	0.052	0.422	0.112	0.053	0.359	0.051
HeatS	0.039	0.336	0.116	0.001	0.020	0.099
ProbS + HeatS, $\lambda = 0.2$	0.067	0.548	0.080	0.062	0.421	0.046

based on RMSE or MAE computed for a small test set of predictions (which could again be created by taking 10% of the input data) and only then reporting the resulting method's performance computed for the probe. However, with only three free parameters to tune, the results are likely to differ little from the results obtained by our naive approach where $\{D, \eta, \lambda\}$ are chosen directly from performance observed for the probe.

While numerical performance values as shown in Table 6 may seem very close to each other across all the methods (perhaps with the exception of *overall average*), differences between the methods from the user's point of view are much greater than one would expect from RMSE varying at the second decimal place. For example, *user average* outperforms *object average* for the Netflix data set, yet in fact it has zero filtering ability. For a given user, estimated rating of all unrated objects is the same (equal to this user's average rating), this user is thus provided no useful information as to which object to select in the future. Further, the performance of *object average* may seem appealing with respect to the method's simplicity, yet one can easily check that objects with the highest estimated score are likely those that received only a few ratings. This is because while a rarely viewed object may easily receive the highest possible average rating of five, a popular object inevitable receives some worse marks which result in a lower average rating. For example, top-rated movies in both tested data sets have all received less than five ratings and scored 5.0 on average. This analysis shows that RMSE and MAE, while useful and easy to understand, give only a very limited information about a method's performance.

Table 7 summarizes performance of methods requiring data without explicit ratings (binary data). As performance metrics we use precision P, recall R and the relative rank (marked as rank in the table) of probe objects (if probe object α belonging to user i appears on place x of this user's recommendation list and this user has collected k_i objects, the relative rank of α is $x/(M-k_i)$). Method *global rank* corresponds to recommendation of the most popular items that have not yet been collected by a given user. The results of pLSA and LDA are obtained from $K = 50$, while a slight increase in performance is observed when K increases further. Results of Bayesian clustering are obtained with $(K_{\text{user}}, K_{\text{item}}) = (70, 35)$ for Movielens and $(K_{\text{user}}, K_{\text{item}}) = (70, 140)$ for Netflix, where $K_{\text{user}}/K_{\text{items}}$ is in a rough proportion to the corresponding ratio of users to items. Note that *global rank* and Bayesian clustering are able to yield low relative rank but they fail to score in precision and recall.

10. Outlook

After reviewing extensively the past work in the field of recommender systems, we describe here a few challenges that the field of recommendation needs to tackle in the future.

To begin with, let us take the conceptual question of the possibility of effective recommendation. For a long time, we all thought that we know ourselves better than anyone else does. Without much fanfare and fuss about philosophical implications, IT experts and online businesses continue in their exploration of the part of our knowledge that resides not on in our minds, but at crossroads of communities. This in effect has violated both our self-knowledge belief and an implicit, long cherished notion: individuals are masters of themselves. Conceptually, this admitting is nothing short of a revolution. The

potential of the new approach is huge: the "extra-body" knowledge about our preferences manifests itself in communication data and is thus much easier to analyze and decipher than the part hiding among our neurons and synapses. What Amazon or Netflix does is just scratching on the surface of the huge potential, as they only take into account a tiny fraction of information about us. Google's Gmail gains more insight about what its users do in the hope that emails reveal more than what can be obtained from data about searching, book-buying, and movie-rating, thereby matching ads more closely to our preferences and hence increasing their efficiency. Some recent online communities go even a step further than Gmail. For example, Facebook.com lets its members to create trusted relationships and keeps track of members' activities and conversations, obtaining the opportunity to infer intimate details about users' preferences. Though most of this information is implicit and not yet ready for recommendation, the huge data basis in principle can yield much more insight than hitherto seen. By letting the users to reveal more and more, the potential for inferring their future wants grows and we are still to know what the consequences will be. The induced danger of privacy violation calls for new, privacy preserving recommender systems where no sensitive data leave users' computers, yet users can enjoy the benefit of collaborative filtering.

IMDB (Internet Movie Database) and similar web sites aggregate millions of votes for a wide range of movies, and online sellers of movies use some degree of collaborative filtering to make recommendations given one's past purchasing history. However, an open reputation-sharing mechanism remains to become widespread. One can project forward to imagine innovative applications, such as "Movies Wanted": a system where plot descriptions are collaboratively developed and voted on, to highlight movies desired by a constituency. The net effect of reputation filtering will be to bring more old, foreign, and niche movies to light, with similar effects for music and other culture. Cultural opportunities that languish for want of attention due to high search costs will reach audiences that did not know what they were missing. Many recommender systems provide suggestions based on expressed or observed preferences. But reputations could also encode other properties of media, such as "ethicalness" of lyrics (and indeed of the performers' lives and aims if one desires), or specific legal or reproduction rights. Licensing schemes like Creative Commons certify an artistic work as having particular legal properties; it is then feasible to provide both recommendations and direct access just within the set of freely available music.

Beyond music and movies, numerous cultural areas and experience goods are ripe for recommendation services provided by reputations. Book ratings and suggestions provide a navigation tool through humanity's ever-growing literary output—most notably from Amazon, but also from a variety of small-scale services and personal lists. Travel guidebooks aid in getting the insider view of an unfamiliar locale, but interpreted experiences of natives and previous travelers could be even better. Whether for festivals, museums, opera, or the thousands of other shared activities which enrich our social landscape, the cultural sector is fertile ground for development.

In the age of Internet and World Wide Web, ICT tools allows information to spread faster and wider than ever before, and dominates the way we form our opinions and knowledge. While there has been an undeniable progress in the information availability, a fundamental question remains elusive: do we get more diverse information than in the past? Although the ICT Revolution was expected to allow people to access ever more diversified information sources and products, one often sees that this is not the case. Popular viral videos copy the strategy of blockbuster films and target the tastes of the general audience, giving rise to global hits. On the Internet, a few sites attire a huge part of web traffic. A similar process is in action in science where disproportional attention is given to a small fraction of all new and exciting works [302]. The problem is that search engines and recommender systems fall prey to a self-reinforcing rich-get-richer phenomenon: items that were popular in the past tend to be served to even more users in the future. The natural outcomes of such defective dynamics are the narrowing of people tastes and opinions together with a general cultural flattening. To address this issue, we need to consider the long-term impacts of information filtering systems on the information ecology and study information filtering tools that favor diversity without sacrificing their overall performance [39].

Another interesting facet of the mentioned diversity challenge is related to a concept of "crowd-avoidance". There are situations where the generated data naturally fits the paradigm of recommender systems, yet using a standard recommender system may result in poor outcomes. For example, when given data of user preferences for restaurants, it is natural to recommend a user a new place to eat. However, if too many users are recommended the same place, it gets crowded and nobody enjoys their meal. Similarly, when given data of industrial sectors active in various countries (which can be effectively represented by a bipartite network [327], so much discussed and utilized in this review), one may recommend a country a new sector on the basis of its similarity with already active sectors. However, if the country faces a strong competition from its neighbors, it may do better by choosing a less similar sector where the competition is weaker. The same happens on a smaller scale where companies routinely compete with products of other companies, yet avoiding a direct clash may be very beneficial. The concept of crowd avoidance in recommender systems could yield benefits in situations similar to those described above, where resources cannot be shared by an arbitrary number of parties due to constraints of geographic space or limited interest of customers.

The crowd avoidance phenomenon ranges from practically non-existing (in the case of e-book distribution, for example) to very strong (no two customers can share an item) where it approaches the classical assignment problem [328]. The most challenging seems to be the moderate case where recommending an item to a small number of consumers does not create a problem yet (which fits well the above-described product and restaurant recommendation, for example). Note that this whole problem is in principle similar to quantum physics systems where occupation numbers are confined by constraints (such as the Pauli exclusion principle which says that two identical fermions cannot simultaneously occupy the same quantum state) or where mutual repulsion among particles sharing the same site exists. Analogies with physics can thus prove useful in studying this kind of systems.

The science of recommendation is just starting—despite impressive progresses, much remains to be understood. For further advances intuition alone is no longer enough and a multidisciplinary approach will surely bring powerful tools that may help innovative matchmakers to turn the immense potential of recommendations into real life applications.

Acknowledgments

This work was partially supported by the EU FET-Open Grant 231200 (project QLectives) and National Natural Science Foundation of China (Grant Nos. 11075031, 11105024, 61103109 and 60973069). CHY is partially supported by EU FET FP7 project STAMINA (FP7-265496). TZ is supported by the Research Funds for the Central Universities (UESTC).

References

[1] D.J. Watts, A twenty-first century science, Nature 445 (2007) 489.
[2] A. Vespignani, Predicting the behavior of techno-social systems, Science 325 (2009) 425–428.
[3] R.N. Mantegna, H.E. Stanley, An Introduction to Econophysics: Correlations and Complexity in Finance, Cambridge University Press, Cambridge, 2000.
[4] J.-P. Bouchaud, M. Potters, Theory of Financial Risk and Derivative Pricing: From Statistical Physics to Risk Management, Cambridge University Press, Cambridge, 2009.
[5] R. Albert, A.-L. Barabási, Statistical mechanics of complex networks, Reviews of Modern Physics 74 (2002) 47–97.
[6] S.N. Dorogovtsev, J.F.F. Mendes, Evolution of networks, Advances in Physics 51 (2002) 1079–1187.
[7] M.E.J. Newman, The structure and function of complex networks, SIAM Review 45 (2003) 167–256.
[8] S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, D.-U. Huang, Complex networks: structure and dynamics, Physics Report 424 (2006) 175–308.
[9] M.E.J. Newman, A.-L. Barabási, D.J. Watts, The Structure and Dynamics of Networks, Princeton University Press, Princeton, 2006.
[10] C. Castellano, S. Fortunato, V. Loreto, Statistical physics of social dynamics, Review of Modern Physics 81 (2009) 591–646.
[11] J. Ellenberg, This psychologist might outsmart the math brains competing for the netflix prize, Wired Magazine, 2008, pp. 114–122.
[12] J. Hagel III, M. Singer, Net Worth: Shaping Markets When Customers Make the Rules, Harvard Business School Press, Boston, 1999.
[13] J.L. Herlocker, J.A. Konstan, K. Terveen, J.T. Riedl, Evaluating collaborative filtering recommender systems, ACM Transactions on Information Systems 22 (2004) 5–53.
[14] F. Ricci, L. Rokach, B. Shapira, P.B. Kantor (Eds.), Recommender Systems Handbook, Springer, New York, NY, USA, 2011.
[15] G. Adomavicius, A. Tuzhilin, Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions, IEEE Transactions on Knowledge and Data Engineering 17 (2005) 734–749.
[16] C. Anderson, The Long Tail: Why the Future of Business is Selling Less of More, Hyperion, 2006.
[17] E. Brynjolfsson, Y. Hu, M.D. Smith, Consumer surplus in the digital economy: estimating the value of increased product variety at online booksellers, Management Science 49 (2003) 1580–1596.
[18] J.B. Schafer, J.A. Konstan, J. Riedl, E-commerce recommendation applications, Data Mining and Knowledge Discovery 5 (2001) 115–153.
[19] P.-Y. Chen, S.-Y. Wu, J. Yoon, The impact of online recommendations and consumer feedback on sales, in: Proceedings of the 25th International Conference on Information Systems, 2004, pp. 711–724.
[20] J. Bennett, S. Lanning, The Netflix prize, in: Proceedings of KDD Cup and Workshop, 2007, pp. 3–6.
[21] R.M. Bell, Y. Koren, Lessons from the Netflix prize challenge, ACM SIGKDD Explorations Newsletter 9 (2007) 75–79.
[22] Y. Koren, The bellkor solution to the netflix grand prize, Report from the Netflix Prize Winners, 2009.
[23] M. Piotte, M. Chabbert, The pragmatic theory solution to the netflix grand prize, Report from the Netflix Prize Winners, 2009.
[24] A. Töscher, M. Jahrer, The bigchaos solution to the netflix grand prize, Report from the Netflix Prize Winners, 2009.
[25] A. Narayanan, V. Shmatikov, Robust de-anonymization of large sparse datasets, in: IEEE Symposium on Security and Privacy, 2008, pp. 111–125.
[26] M.E.J. Newman, Power laws Pareto distributions and Zipf's law, Contemporary Physics 46 (2005) 323–351.
[27] W. Weibull, A statistical distribution function of wide applicability, J. Appl. Mech.-Trans. ASME 18 (3) (1951) 293–297.
[28] Z. Huang, H. Chen, D. Zeng, Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering, ACM Transactions on Information Systems 22 (2004) 116–142.
[29] B. Sarwar, J. Konstan, J. Riedl, Incremental singular value decomposition algorithms for highly scalable recommender systems, in: International Conference on Computer and Information Science, 2002, pp. 27–28.
[30] C.-H. Jin, J.-G. Liu, Y.-C. Zhang, T. Zhou, Adaptive information filtering for dynamics recommender systems, arXiv:0911.4910, 2009.
[31] M.H. Holmes, Introduction to Perturbation Methods, Springer, 2005.
[32] A.I. Schein, A. Popescul, L.H. Ungar, D.M. Pennock, Methods and metrics for cold-start recommendations, in: Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM Press, New York, 2002, pp. 253–260.
[33] X.N. Lam, T. Vu, T.D. Le, A.D. Duong, Addressing cold-start problem in recommendation systems, in: Proceedings of the 2nd International Conference on Ubiquitous Information Management and Communication, 2008, pp. 208–211.
[34] L. Zhang, L.-S. Bai, T. Zhou, Crossing recommendation based on multi-B2C behavior, J. Univ. Elect. Sci. Technol. China (to be published).
[35] S.M. Mcnee, J. Riedl, J.A. Konstan, Being accurate is not enough: how accuracy metrics have hurt recommender systems, in: Proceedings of the CHI 06 Conference on Human Factors in Computing Systems, ACM, New York, 2006, pp. 1097–1101.
[36] B. Smyth, P. Mcclave, Similarity vs. diversity, in: D.W. Aha, I. Watson (Eds.), Case-Based Reasoning Research and Development, Springer, 2001, pp. 347–361.
[37] C.-N. Ziegler, S.M. Mcnee, J.A. Konstan, G. Lausen, Improving recommendation lists through topic diversification, in: Proceedings of the 14th International Conference on World Wide Web, ACM, New York, 2005, pp. 22–32.
[38] N. Hurley, M. Zhang, Novelty and diversity in top-N recommendation—analysis and evaluation, ACM Transactions on Internet Technology 10 (2011) 14.
[39] T. Zhou, Z. Kuscsik, J.-G. Liu, M. Medo, J.R. Wakeling, Y.-C. Zhang, Solving the apparent diversity–accuracy dilemma of recommender systems, Proceedings of the National Academy of Sciences of the United States of America 107 (2010) 4511–4515.
[40] B. Mobasher, R. Burke, R. Bhaumik, C. Williams, Towards trustworthy recommender systems: an analysis of attack models and algorithm robustness, ACM Transactions on Internet Technology 7 (2007) 23.
[41] S.K. Lam, D. Frankowski, J. Riedl, Do You Trust Your Recommendations? An Exploration of Security and Privacy Issues in Recommender Systems, in: Lecture Notes in Computer Science, vol. 3995, Springer, Heidelberg, Germany, 2006, pp. 14–29.
[42] R. Burke, M.P. O'mahony, N.J. Hurley, Robust Collaborative Recommendation, in: F. Ricci, L. Rokach, B. Shapira, P.B. Kantor (Eds.), Recommender Systems Handbook, Part 5, Springer, 2011, pp. 805–835. (Chapter 25).
[43] S.-H. Min, I. Han, Detection of the customer time-variant pattern for improving recommender systems, Expert Systems with Applications 28 (2005) 189–199.
[44] L. Xiang, Q. Yuan, S. Zhao, L. Chen, X. Zhang, Q. Yang, J. Sun, Temporal recommendation on graphs via long-and short-term preference fusion, in: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM Press, New York, 2010, pp. 723–732.

[45] R. Sinha, K. Swearingen, The role of transparency in recommender systems, in: Proceedings of the CHI'06 Conference on Human Factors in Computing Systems, 2002, pp. 830–831.
[46] A.D.J. Cooke, H. Sujan, M. Sujan, B.A. Weitz, Marketing the unfamiliar: the role of context and item-specific information in electronic agent recommendations, Journal of Marketing Research 39 (2002) 488–497.
[47] Z. Huang, D. Zeng, H. Chen, Analyzing consumer–product graphs: empirical findings and applications in recommender systems, Management Science 53 (2007) 1146–1164.
[48] S. Sahebi, W.W. Cohen, Community-based recommendations: a solution to the cold start problem (unpublished).
[49] C. Song, Z. Qu, N. Blumm, A.-L. Barabási, Limits of predictability in human mobility, Science 327 (2010) 1018–1021.
[50] E. Cho, S.A. Myers, J. Leskovec, Friendship and mobility: user movement in location-based social networks, in: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM Press, New York, 2011, pp. 1082–1090.
[51] V.W. Zheng, Y. Zheng, X. Xie, Q. Yang, Collaborative location and activity recommendations with GPS history data, in: Proceedings of the 19th International Conference on World Wide Web, ACM, New York, 2010, pp. 1029–1038.
[52] M. Clements, P. Serdyukov, A.P.De. Vries, M.J.T. Reinders, Personalised travel recommendation based on location co-occurrence, arXiv:1106.5213, 2011.
[53] M.-S. Shang, L. Lü, Y.-C. Zhang, T. Zhou, Empirical analysis of web-based user–object bipartite networks, EPL 90 (2010) 48006.
[54] C.-J. Zhang, A. Zeng, Behavior patterns of online users and the effect on information filtering, Physica A 391 (2012) 1822–1830.
[55] J. Vig, S. Sen, J. Riedl, Navigation the tag genome, in: Proceedings of the 16th International Conference on Intelligent User Interfaces, ACM Press, New York, 2011, pp. 93–102.
[56] L. Chen, P. Pu, Critiquing-based recommenders: survey and emerging trends, User Modeling and User-Adapted Interaction 22 (2012) 125–150.
[57] L. Euler, Solutio problematis ad geometriam situs pertinentis, Commentarii academiae scientiarum Petropolitanae 8 (1741) 128–140.
[58] B. Bollobás, Modern Graph Theory, Springer-Verlag, New York, 1998.
[59] P. Erdös, A. Rényi, On random graphs I, Publicationes Mathematicae (Debrecen) 6 (1959) 290–297.
[60] B. Bollobás, Random Graphs, Academic Press, London, 1985.
[61] G. Caldarelli, Scale-Free Networks, Oxford University Press, Oxford, 2007.
[62] A. Clauset, C.R. Shalizi, M.E.J. Newman, Power-law distributions in empirical data, SIAM Review 51 (2009) 661–703.
[63] M.L. Goldstein, S.A. Morris, G.G. Yen, Problems with fitting to the power-law distribution, European Physical Journal B 41 (2004) 255–258.
[64] R. Pastor-Satorras, A. Vespignani, Epidemic spreading in scale-free networks, Physical Review Letters 86 (2001) 3200–3203.
[65] A. Vázquez, R. Pastor-Satorras, A. Vespignani, Large-scale topological and dynamical properties of the internet, Physical Review E 65 (2002) 066130.
[66] M.E.J. Newman, Assortative mixing in networks, Physical Review Letters 89 (2002) 208701.
[67] M.E.J. Newman, Mixing patterns in networks, Physical Review E 67 (2003) 026126.
[68] S. Zhou, R.J. Mondragón, Structural constraints in complex networks, New Journal of Physics 9 (2007) 173.
[69] V. Latora, M. Marchiori, Efficient behavior of small-world networks, Physical Review Letters 87 (2001) 198701.
[70] S. Milgram, The small world problem, Psychology Today 2 (1967) 60–67.
[71] D.J. Watts, S.H. Strogatz, Collective dynamics of 'small-world' networks, Nature 393 (1998) 440–442.
[72] G. Simmel, Soziogie: Untersuchungen Uber Die Formen Der Vergesellschaftung, Duncket and Humbolt, Berlin, 1908.
[73] S. Wasserman, K. Faust, Social Network Analysis, Cambridge University Press, Cambridge, 1994.
[74] L. da F. Costa, F.A. Rodrigues, G. Travieso, P.R. Villas Boas, Characterization of complex networks: a survey of measurements, Advances in Physics 56 (2007) 167–242.
[75] H. Jeong, B. Tombor, R. Albert, Z.N. Oltvai, A.-L. Barabási, The large-scale organization of metabolic networks, Nature 407 (2000) 651–654.
[76] M.E.J. Newman, The structure of scientific collaboration networks, Proceedings of the National Academy of Sciences of the United States of America 98 (2001) 404–409.
[77] Q. Xuan, F. Du, T.-J. Wu, Empirical analysis of Internet telephone network: from user ID to phone, Chaos 19 (2009) 023101.
[78] J. Grujić, Movies recommendation networks as bipartite graphs, Lecture Notes in Computer Sciecne 5102 (2008) 576–583.
[79] R. Lambiotte, M. Ausloos, Uncovering collective listening habits and music genres in bipartite networks, Physical Review E 72 (2005) 066107.
[80] J. Laherrère, D. Sornette, Stretched exponential distributions in nature and economy: "fat tails" with characteristic scales, The European Physical Journal B 2 (1998) 525–539.
[81] R. Lambiotte, M. Ausloos, Collaborative tagging as a tripartite network, Lecture Notes in Computer Science 3993 (2006) 1114–1117.
[82] C. Cattuto, C. Schmitz, A. Baldassarri, V.D.P. Servedio, V. Loreto, A. Hotho, M. Grahl, G. Stumme, Network properties of folksonomies, AI Communications 20 (2007) 245–262.
[83] G. Palla, I.J. Farkas, P. Pollner, I. Derényi, T. Vicsek, Fundamental statistical features and self-similar properties of tagged networks, New Journal of Physics 10 (2008) 123026.
[84] Z.-K. Zhang, T. Zhou, Y.-C. Zhang, Tag-aware recommender systems: a state-of-the-art survery, Journal of Computer Science and Technology 26 (2011) 767–777.
[85] C. Berge, Graphs and Hypergraphs, North-Holland Publishing Company, London, 1973.
[86] V. Zlatić, G. Ghoshal, G. Caldarelli, Hypergraph topological quantities for tagged social networks, Physical Review E 80 (2009) 036118.
[87] A. Vazquez, Finding hypergraph communities: a Bayesian approach and variational solution, Journal of Statistical Mechanics: Theory and Experiment (2009) P07006.
[88] D. Bollé, R. Heylen, N.S. Skantzos, Thermodynamics of spin systems on small-world hypergraphs, Physical Review E 74 (2006) 056111.
[89] D. Bollé, R. Heylen, Small-world hypergraphs on a bond-disordered Bethe lattice, Physical Review E 77 (2008) 046104.
[90] A. Vázquez, Population stratification using a statistical model on hypergraphs, Physical Review E 77 (2008) 066106.
[91] S. Klamt, U.-U. Haus, F. Theis, Hypergraphs and cellular networks, PLoS Computational Biology 5 (2009) e1000385.
[92] C. Taramasco, J.-P. Cointet, C. Roth, Academic team formation as evolving hypergraphs, Scientometrics 85 (2010) 721–740.
[93] G. Ghoshal, V. Zlatić, G. Caldarelli, M.E.J. Newman, Random hypergraphs and their applications, Physical Review E 79 (2009) 066118.
[94] Z.-K. Zhang, C. Liu, A hypergraph model of social tagging networks, Journal of Statistical Mechanics: Theory and Experiment (2010) P10005.
[95] Y. Hu, Y. Koren, C. Volinsky, Collaborative filtering for implicit feedback datasets, in: Proceedings of the 8th IEEE International Conference on Data Mining, IEEE Press, 2008, pp. 263–272.
[96] Y. Koren, J. Sill, OrdRec: an ordinal model for predicting personalized item rating distributions, in: Proceedings of the 5th ACM Conference on Recommender Systems, ACM Press, New York, 2011, pp. 117–124.
[97] J.L. Rodgers, W.A. Nicewander, Thirteen ways to look at the correlation coefficient, The American Statistician 42 (1988) 59–66.
[98] C. Spearman, The proof and measurement of association between two things, The American Journal of Psychology 15 (1904) 72–101.
[99] M. Kendall, A new measure of rank correlation, Biometrika 30 (1938) 81–89.
[100] Y.Y. Yao, Measuring retrieval effectiveness based on user preference of documents, Journal of the American Society for Information Science 46 (1995) 133–145.
[101] J.A. Hanely, B.J. McNeil, The meaning and use of the area under a receiver operating characteristic (ROC) curve, Radiology 143 (1982) 29–36.
[102] T. Zhou, L. Lü, Y.-C. Zhang, Predicting missing links via local information, The European Physical Journal B 71 (2009) 623–630.
[103] T. Zhou, J. Ren, M. Medo, Y.-C. Zhang, Bipartite network projection and personal recommendation, Physical Review E 76 (2007) 046115.
[104] C.J. van Rijsbergen, Information Retrieval, second ed., Butterworth-Heinemann Newton, MA, USA, 1979.
[105] M. Pazzani, D. Billsus, Learning and revising user profiles: the identification of interesting web sites, Machine Learning 27 (1997) 313–331.
[106] C. Buckley, E.M. Voorhees, Retrieval system evaluation, in: E.M. Voorhees, D.K. Harman (Eds.), TREC: Experiment and Evaluation in Information Retrieval, MIT Press, Cambridge, MA, 2005, pp. 53–75.

[107] C. Buckley, E.M. Voorhees, Retrieval evaluation with incomplete information, in: Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM Press, New York, 2004, pp. 25–32.
[108] A. Moffat, J. Zobel, Rank-biased precision for measurement of retrieval effectiveness, ACM Transactions on Information Systems 27 (2008) 2–27.
[109] J.S. Breese, D. Heckerman, C. Kadie, Empirical analysis of predictive algorithms for collaborative filtering, in: Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence, 1998, pp. 43–52.
[110] K. Järvelin, J. Kekäläinen, Cumulated gain-based evaluation of IR techniques, ACM Transactions on Information Systems 20 (2002) 422–446.
[111] P. Castells, S. Vargas, J. Wang, Novelty and diversity metrics for recommender systems: choice, discovery and relevance, in: Proceedings of International Workshop on Diversity in Document Retrieval, DDR, ACM Press, New York, 2011, pp. 29–37.
[112] T. Zhou, L.-L. Jinag, R.-Q. Su, Y.-C. Zhang, Effect of initial configuration on network-based recommendation, EPL 81 (2008) 58004.
[113] T. Zhou, R.-Q. Su, R.-R. Liu, L.-L. Jiang, B.-H. Wang, Y.-C. Zhang, Accurate and diverse recommendations via eliminating redundant correlations, New Journal of Physics 11 (2009) 123008.
[114] M. Tribus, Thermostatics and Thermodynamics, Van Nostrand, Princeton, NJ, 1961.
[115] L. Lü, W. Liu, Information filtering via preferential diffusion, Physical Review E 83 (2011) 066119.
[116] F. Cacheda, V. Carneiro, D. Fernández, V. Formoso, Comparison of collaborative filtering algorithms: limitations of current techniques and proposals for scalable, high-performance recommender systems, ACM Transactions on Web 5 (2011) 2.
[117] G. Linden, B. Smith, J. York, Amazon.com recommendations:item-to-item collaborative filtering, IEEE Internet Computing 7 (2003) 76–80.
[118] B. Sarwar, G. Karypis, J. Konstan, J. Riedl, Analysis of recommendation algorithms for e-commerce, in: Proceedings of the 2nd ACM Conference on Electronic Commerce, 2000, pp. 158–167.
[119] W. Zeng, M.-S. Shang, Q.-M. Zhang, L. Lü, T. Zhou, Can dissimilar users contribute to accuracy and diversity of personalized recommendation? International Journal of Modern Physics C 21 (2010) 1217–1227.
[120] W. Zeng, Y.-X. Zhu, L. Lü, T. Zhou, Negative ratings play a positive role in information filtering, Phyisca A 390 (2011) 4486–4493.
[121] J.S. Kong, K. Teague, J. Kessler, Just count the love-hate squares: a rating network based method for recommender system, in: Proceedings of KDD Cup Workshop at SIGKDD'11, the 13th ACM International Conference on Knowledge Discovery and Data Mining, ACM Press, New York, 2011.
[122] M.-S. Shang, L. Lü, T. Zhou, Y.-C. Zhang, Relevance is more significant than correlation: information filtering on sparse data, EPL 88 (2009) 68008.
[123] X. Su, T.M. Khoshgoftaar, A survey of collaborative filtering techniques, Advances in Artificial Intelligence 2009 (2009) 4.
[124] D. Almazro, G. Shahatah, L. Albdulkarim, M. Kherees, R. Martinez, W. Nzoukou, A survey paper on recommender systems, arXiv:1006.5278, 2010.
[125] L. Lü, T. Zhou, Link prediction in complex networks: a survey, Physica A 390 (2011) 1150–1170.
[126] D. Goldberg, D. Nichols, B. Oki, D. Terry, Using collaborative filtering to weave an information tapestry, Communications of the ACM 35 (1992) 61–70.
[127] U. Shardanand, P. Maes, Social information filtering: algorithms for automating "word of mouth", in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, ACM Press, New York, 1995, pp. 210–217.
[128] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, J. Riedl, GroupLens: an open architecture for collaborative filtering of netnews, in: Proceedings of the 1994 ACM conference on Computer Supported Cooperative Work, ACM Press, New York, 1994, pp. 175–186.
[129] K. Goldberg, T. Roeder, D. Gupta, C. Perkins, Eigentaste: a constant time collaborative filtering algorithm, Information Retrieval 4 (2001) 133–151.
[130] B. Sarwar, G. Karypis, J. Konstan, J. Riedl, Item-based collaborative filtering recommendation algorithms, in: Proceedings of the 10th International Conference on World Wide Web, 2001, pp. 285–295.
[131] J. Wang, A.P. de Vries, 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 Rretrieval, ACM Press, New York, 2006, pp. 501–508.
[132] Z.B. Liu, W.Y. Qu, H.T. Li, C.S. Xie, A hybrid collaborative filtering recommendation mechanism for P2P networks, Future Generation Computer Systems 26 (2010) 1409–1417.
[133] D. Lemire, A. Maclachlan, Slope One predictors for online rating-based collaborative filtering, in: Proceedings of SIAM Data Mining, SDM'05, 2005.
[134] P. Wang, H.W. Ye, A personalized recommendation algorithm combining slope one scheme and user based collaborative filtering, in: Proceedings of IIS '09, IEEE Computer Society, Washington, DC, 2009, pp. 152–154.
[135] D. Zhang, An item-based collaborative filtering recommendation algorithm using slope one scheme smoothing, in: Proceedings of ISECS '09, IEEE Computer Society, 2009, pp. 215–217.
[136] M. Gao, Z. Wu, Personalized context-aware collaborative filtering based on neural network and slope one, Lecture Notes in Computer Science 5738 (2009) 109–116.
[137] J.L. Herlocker, J.A. Konstan, L.G. Terveen, J.T. Riedl, Evaluating collaborative filtering recommender systems, ACM Transactions on Information Systems 22 (2004) 5–53.
[138] J.L. Herlocker, J.A. Konstan, A. Borchers, J. Riedl, An algorithmic framework for performing collaborative filtering, in: Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM Press, New York, 1999, pp. 230–237.
[139] D. Liben-Nowell, J. Kleinberg, The link-prediction problem for social networks, Journal of the American Society for Information Science and Technology 58 (2007) 1019–1031.
[140] G. Salton, M.J. Mcgill, Introduction to Modern Information Retrieval, MuGraw-Hill, Auckland, 1983.
[141] P. Jaccard, Étude comparative de la distribution florale dans une portion des Alpes et des Jura, Bulletin de la Societe Vaudoise des Science Naturelles 37 (1901) 547–579.
[142] T. Sørensen, A method of establishing groups of equal amplitude in plant sociology based on similarity of species content and its application to analyses of the vegetation on Danish commons, Biologiske Skrifter 5 (1948) 1–34.
[143] E. Ravasz, A.L. Somera, D.A. Mongru, Z.N. Oltvai, A.-L. Barabási, Hierarchical organization of modularity in metabolic networks, Science 297 (2002) 1551–1555.
[144] E.A. Leicht, P. Holme, M.E.J. Newman, Vertex similarity in networks, Physical Review E 73 (2006) 026120.
[145] L.A. Adamic, E. Adar, Friends and neighbors on the web, Social Networks 25 (2003) 211–230.
[146] A.-L. Barabási, R. Albert, Emergence of scaling in random networks, Science 286 (1999) 509–512.
[147] P. Holme, B.J. Kim, C.N. Yoon, S.K. Han, Attack vulnerability of complex networks, Physical Review E 65 (2002) 056109.
[148] C.-Y. Yin, W.-X. Wang, G.-R. Chen, B.-H. Wang, Decoupling process for better synchronizability on scale-free networks, Physical Review E 74 (2006) 047102.
[149] G.-Q. Zhang, D. Wang, G.-J. Li, Enhancing the transmission efficiency by edge deletion in scale-free networks, Physical Review E 76 (2007) 017101.
[150] L. Lü, C.-H. Jin, T. Zhou, Similarity index based on local paths for link prediction of complex networks, Physical Review E 80 (2009) 046122.
[151] L. Katz, A new status index derived from sociometric analysis, Psychmetrika 18 (1953) 39–43.
[152] D.J. Klein, M. Randic, Resistance distance, Journal of Mathematical Chemistr 12 (1993) 81–95.
[153] F. Fouss, A. Pirotte, J.-M. Renders, M. Saerens, Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation, IEEE Transactions on Knowledge and Data Engineering 19 (2007) 355–369.
[154] S. Brin, L. Page, The anatomy of a large-scale hypertextual Web search engine, Computer Networks and ISDN Systems 30 (1998) 107–117.
[155] H. Tong, C. Faloutsos, J.-Y. Pan, Fast random walk with restart and its applications, in: Proceedings of the 6th IEEE International Conference on Data Mining, IEEE Press, Hong Kong, China, 2006, pp. 613–622.
[156] G. Jeh, J. Widom, SimRank: a measure of structural-context similarity, in: Proceedings of the 8h ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM Press, New York, 2002, pp. 538–543.
[157] P. Chebotarev, E.V. Shamis, The matrix-forest theorem and measuring relations in small social groups, Automation and Remote Control 58 (1997) 1505–1514.
[158] W.-P. Liu, L. Lü, Link prediction based on local random walk, EPL 89 (2010) 58007.

[159] K.H.L. Tso, L. Schmidt-Thieme, Empirical analysis of attribute-aware recommendation algorithms with variable synthetic data, Data Science and Classification (2006) 271–278.
[160] A. Kobsa, Privacy-enhanced web personalization, in: P. Brusilovsky, A. Kobsa, W. Nejdl (Eds.), The Adaptive Web: Methods and Strategies of Web Personalization, Springer-Verlag, Berlin, Heidelberg, 2007, pp. 628–670.
[161] M.J. Pazzani, D. Billsus, Content-based recommendation systems, The Adaptive Web (2007) 325–341.
[162] M. Claypool, A. Gokhale, T. Miranda, P. Murnikov, D. Netes, M. Sartin, Combining content-based and collaborative filters in an online newspaper, in: Proceedings of ACM SIGIR Workshop on Recommender Systems, ACM Press, New York, 1999.
[163] M. Pazzani, A framework for collaborative, content-based, and demographic filtering, Artificial Intelligence Review 13 (1999) 393–408.
[164] M. Balabanović, Y. Shoham, Fab: content-based, collaborative recommendation, Communications of the ACM 40 (1997) 66–72.
[165] P. Melville, R.J. Mooney, R. Nagarajan, Content-boosted collaborative filtering for improved recommendations, in: Proceedings of the 2002 National Conference on Artificial Intelligence, AAAI Press, Edmonton, Canada, 2002, pp. 187–192.
[166] J. Salter, N. Antonopoulos, CinemaScreen recommender agent: combining collaborative and content-based filtering, IEEE Intelligent Systems 21 (2006) 35–41.
[167] I. Soboroff, C. Nicholas, Combining content and collaboration in text filtering, in: Proceedings of the IJCAI Workshop on Machine Learning in Information Filtering, 1999, pp. 86–91.
[168] C. Basu, H. Hirsh, W. Cohen, Recommendation as classification: using social and content-based information in recommendation, in: Proceedings of the 1998 National Conference on Artificial Intelligence, AAAI Press, 1998, pp. 714–720.
[169] A. Popescul, L.H. Ungar, D.M. Pennock, S. Lawrence, Probabilistic models for unified collaborative and content-based recommendation in sparse-data environments, in: Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence, Morgan Kaufmann Publishers Inc., San Francisco, 2001, pp. 437–444.
[170] K. Yu, A. Schwaighofer, V. Tresp, W.-Y. Ma, H. Zhang, Collaborative ensemble learning: combining collaborative and content-based information filtering, in: Proceedings of the 19th Conference on Uncertainty in Artificial Intelligence, 2003, pp. 616–623.
[171] X. Jin, Y. Zhou, B. Mobasher, A maximum entropy web recommendation system: combining collaborative and content features, in: Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining, New York, NY, USA, 2005, pp. 612–617.
[172] G. Takács, I. Pilászy, B. Németh, D. Tikk, On the gravity recommendation system, In: Proceedings of KDD Cup Workshop at SIGKDD'07, 13th ACM International Conference on Knowledge Discovery and Data Mining, San Jose, CA, USA, 2007, pp. 22–30.
[173] L. Ungar, D. Foster, A formal statistical approach to collaborative filtering, In: Proceedings of the Conference of Automated Learning and Discovery, Pittsburg, PA, USA, 1998.
[174] T. Hofmann, Latent semantic models for collaborative filtering, ACM Transactions on Information Systems 22 (2004) 89–115.
[175] D.L. Blei, A.Y. Ng, M.I. Jordan, Latent Dirichlet allocation, The Journal of Machine Learning Research 3 (2003) 993–1022.
[176] E.J. Candés, B. Recht, Exact matrix completion via convex optimization, Foundations of Computational Mathematics 9 (2009) 717–772.
[177] E.J. Candés, Y. Plan, Matrix completion with noise, Proceedings of the IEEE 98 (2010) 926–936.
[178] R.H. Keshavan, A. Montanari, S. Oh, Matrix completion from noisy entries, Journal of Machine Learning Research 11 (2010) 2057–2078.
[179] R.H. Keshavan, A. Montanari, S. Oh, Matrix completion from a few entries, IEEE Trans. Inform. Theory 56 (2010) 2980–2998.
[180] R.H. Keshavan, S. Oh, A gradient descent algorithm on the grassman manifold for matrix completion, arXiv.org:0910.5260 (unpublished).
[181] R. Gemulla, P.J. Hass, E. Nijkamp, Y. Sismanis, Large-scale matrix factorization with stochastic gradient descent, in: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM Press, New York, 2011, pp. 69–77.
[182] Y. Koren, R. Bell, C. Volinsky, Matrix factorization techniques for recommender systems, Computer 42 (2009) 30–37.
[183] H. Ma, D. Zhou, C. Liu, M.R. Lyu, I. King, Recommender systems with social regularization, in: Proceedings of the 4th ACM International Conference on Web Search and Data Mining, ACM Press, New York, 2011, pp. 287–296.
[184] L. Getoor, M. Sahami, Using probabilistic relational models for collaborative filtering, In: Proceedings of Workshop on Web Usage Analysis and User Profiling, San Diego, CA, USA, 1999.
[185] J. Pearl, Reverend Bayes on inferenced engines: a distributed hierarchical approach, In: Proceedings of the 2nd AAAI National Conference on Artificial Intelligence, Pittsburgh, USA, 1982, pp. 133–136.
[186] J.S. Yedidia, W.T. Freeman, Y. Weiss, Constructing free-energy approximations and generalized belief propagation algorithms, IEEE Transactions on Information Theory 51 (2005) 2282–2312.
[187] M.R. Gupta, Y. Chen, Theory and use of the EM algorithm, Foundations and Trends in Signal Processing 4 (2010) 223–296.
[188] S. Geman, D. Geman, Stochastic relaxation, gibbs distributions, and the bayesian restoration of images, IEEE Transactions on Pattern Analysis and Machine Intelligence 6 (1984) 721–741.
[189] T.L. Griffiths, M. Steyvers, Finding scientific topics, Proceedings of the National Academy of Sciences of the United States of America 101 (2004) 5228–5235.
[190] M.E.J. Newman, G.T. Barkema, Monte Carlo Methods in Statistical Physics, Oxford University Press, Oxford, 1999.
[191] Y. Zhang, J. Koren, Efficient Bayesian hierarchical user modeling for recommendation systems, in: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM Press, New York, 2007, pp. 47–54.
[192] H. Shan, A. Banerjee, Bayesian co-clustering, in: Proceedings of ICDM'08 (IEEE International Conference on Data Mining), 2008, pp. 530–539.
[193] T. Hofmann, Collaborative filtering via Gaussian probabilistic latent semantic analysis, in: Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval, ACM Press, New York, 2003, pp. 259–266.
[194] D.M. Blei, J.D. Mcauliffe, Supervised topic models, Advances in Neural Information Processing Systems 20 (2008) 121–128.
[195] W.-Y. Chen, J.-C. Chu, J. Luan, H. Bai, Y. Wang, E.Y. Chang, Collaborative filtering for orkut communities: discovery of user latent behavior, in: Proceedings of the 18th International Conference on World Wide Web, 2009, pp. 681–690.
[196] D. Agarwal, B.-C. Chen, fLDA: matrix factorization through latent dirichlet allocation, in: Proceedings of the 3rd ACM International Conference on Web Search and Data Mining, 2010, pp. 91–100.
[197] X. Wei, W.B. Croft, LDA-based document models for ad-hoc retrieval, in: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 2006, pp. 178–185.
[198] T. Griffiths, Gibbs sampling in the generative model of Latent Dirichlet Allocation, Technical Report, Stanford University, 2002.
[199] D. Newman, A. Asuncion, P. Smyth, M. Welling, Distributed inference for latent dirichlet allocation, Advances in Neural Information Processing Systems 20 (2008) 1081–1088.
[200] Y.-C. Zhang, M. Blattner, Y.-K. Yu, Heat conduction process on community networks as a recommendation model, Physical Review Letters 99 (2007) 154301.
[201] Y.-C. Zhang, M. Medo, J. Ren, T. Zhou, T. Li, F. Yang, Recommendation model based on opinion diffusion, EPL 80 (2007) 68003.
[202] A. Stojmirović, Y.-K. Yu, Information flow in interaction networks, Journal of Computational Biology 14 (2007) 1115–1143.
[203] M.-S. Shang, C.-H. Jin, T. Zhou, Y.-C. Zhang, Collaborative filtering based on multi-channel diffusion, Physica A 388 (2009) 4867–4871.
[204] C.-X. Jia, R.-R. Liu, D. Sun, B.-H. Wang, A new weighting method in network-based recommendation, Physica A 387 (2008) 5887–5891.
[205] J.-G. Liu, T. Zhou, H.-A. Che, B.-H. Wang, Y.-C. Zhang, Effects of high-order correlations on personalized recommendations for bipartite networks, Physica A 389 (2010) 881–886.
[206] J.-G. Liu, B.-H. Wang, Q. Guo, Improved collaborative filtering algorithm via information transformation, International Journal of Modern Physics C 20 (2009) 285–293.
[207] J.-G. Liu, T. Zhou, B.-H. Wang, Y.-C. Zhang, Q. Guo, Degree correlation of bipartite network on personalized recommendation, International Journal of Modern Physics C 20 (2009) 137–147.

[208] J.-G. Liu, T. Zhou, B.-H. Wang, Y.-C. Zhang, Q. Guo, Effects of user's tastes on personalized recommendation, International Journal of Modern Physics C 20 (2009) 1925–1932.
[209] Y. Pan, D.-H. Li, J.-G. Liu, J.-Z. Liang, Detecting community structure in complex networks via node similarity, Physica A 389 (2010) 2849–2857.
[210] Y.-L. Wang, T. Zhou, J.-J. Shi, J. Wang, D.-R. He, Empirical analysis of dependence between stations in Chinese railway network, Physica A 388 (2009) 2949–2955.
[211] J.-G. Liu, T. Zhou, Q. Guo, Information filtering via biased heat conduction, Physical Review E 84 (2011) 037101.
[212] J.-G. Liu, Q. Guo, Y.-C. Zhang, Information filtering via weighted heat conduction algorithm, Physica A 390 (2011) 2414–2420.
[213] T. Qiu, G. Chen, Z.-K. Zhang, T. Zhou, An item-oriented recommendation algorithm on cold-start problem, EPL 95 (2011) 58003.
[214] C. Liu, W.-X. Zhou, An improved HeatS+ProbS hybrid recommendation algorithm based on heterogeneous initial resource configurations, arXiv:1005.3124, 2010.
[215] M. Blattner, B-Rank: a top N recommendation algorithm, In: Proceedings of the 1st International Multi-Conference on Complexity, Informatics and Cybernetics, 2010, pp. 336–341.
[216] X. Pan, G.-S. Deng, J.-G. Liu, Information filtering via improved similarity definition, Chinese Physics Letters 27 (2010) 068903.
[217] R. Sinha, K. Swearingen, Comparing recommendations made by online systems and friends, in: Proceedings of the DELOS-NSF Workshop on Personalization and Recommender Systems in Digital Libraries, 2001.
[218] M.J. Salganik, P.S. Dodds, D.J. Watts, Experimental study of inequality and unpredictability in an artificial cultural market, Science 311 (2006) 854–856.
[219] P. Bonhard, M.A. Sasse, ''Knowing me knowing you''—using profiles and social networking to improve recommender systems, BT Technology Journal 24 (2006) 84–98.
[220] S.-Y. Hwang, C.-P. Wei, Y.-F. Liao, Coauthorship networks and academic literature recommendation, Electronic Commerce Research and Applications 9 (2010) 323–334.
[221] P. Symeonidis, E. Tiakas, Y. Manolopoulos, Product recommendation and rating prediction based on multi-modal social networks, in: Proceedings of the 5th ACM Conference on Recommender Systems, ACM Press, New York, 2011, pp. 61–68.
[222] P.M. Herr, F.R. Kardes, J. Kim, Effects of word-of-mouth and product-attribute information on persuasion: an accessibility-diagnosticity perspective, Journal of Consumer Research 17 (1991) 454–462.
[223] P.F. Bone, Word-of-mouth effects on short-term and long-term product judgments, Journal of Business Research 32 (1995) 213–223.
[224] S. Fortunato, C. Castellano, Scaling and universality in proportional elections, Physical Review Letters 99 (2007) 138701.
[225] A. Ellero, G. Fasano, A. Sorato, A modified Galam's model for word-of-mouth information exchange, Physica A 388 (2009) 3901–3910.
[226] J. Arndt, Word-of-mouth Advertising: a review of the literature, Advertising Research Foundation, New York, 1967.
[227] C. Dellarocas, The digitization of word of mouth: promise and challenges of online feedback mechanisms, Management Science 49 (2003) 1407–1424.
[228] J.A. Chevalier, D. Mayzlin, The effect of word of mouth on sales: online book reviews, Journal of Marketing Research 43 (2006) 345–354.
[229] C.A. Yeung, T. Iwata, Strength of social influence in trust networks in product review sites, in: Proceedings of the 4th ACM International Conference on Web Search and Data Mining, ACM Press, New York, 2011, pp. 495–504.
[230] J.E. Phelps, R. Lewis, L. Mobilio, D. Perry, N. Raman, Viral marketing or electronic word-of-mouth advertising: examining consumer responses and motivations to pass along email, Journal of Advertising Research 44 (2004) 333–348.
[231] N. Agarwal, H. Liu, L. Tang, P.S. Yu, Identifying the influential bloggers in a community, in: Proceedings of the International Conference on Web Search and Web Data Mining, ACM Press, New York, 2008, pp. 207–218.
[232] B.J. Jansen, M. Zhang, K. Sobel, A. Chowdury, Twitter power: tweets as electronic word of mouth, Journal of the American Society for Information Science and Technology 60 (2009) 2169–2188.
[233] J. Leskovec, L.A. Adamic, B.A. Huberman, The dynamics of viral marketing, ACM Transactions on Web 1 (2007) 5.
[234] D.M. Romero, B. Meeder, J. Kleinberg, Differences in the mechanics of information diffusion across topics: idioms, political hashtags, and complex contagion on twitter, in: Proceedings of the 20th International Conference on World Wide Web, ACM Press, New York, 2011, pp. 695–704.
[235] J. Tang, J. Sun, C. Wang, Z. Yang, Social influence analysis in large-scale networks, in: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM Press, New York, 2009, pp. 807–816.
[236] C.H. Yeung, G. Cimini, C.-H. Jin, Dynamics of movie competition and popularity spreading in recommender systems, Physical Review E 83 (2011) 016105.
[237] J. Huang, X.-Q. Cheng, H.-W. Shen, T. Zhou, X. Jin, Exploring social influence via posterior effect of word-of-mouth recommendations, in: Proceedings of the 5th ACM International Conference on Web Search and Data Mining, ACM Press, New York, 2012, pp. 573–582.
[238] H. Kwak, C. Lee, H. Park, S. Moon, What is twitter a social network or a news media?, in: Proceedings of the 19th International Conference on World Wide Web, ACM Press, New York, 2010.
[239] S. Guo, M. Wang, J. Leskovec, The role of social networks in online shopping: information passing, price of trust, and consumer choice, in: Proceedings of the 12th ACM Conference on Electronic Commerce, ACM Press, New York, 2011, pp. 157–166.
[240] E. Oster, R. Thornton, Determinants of technology adoption: private value and peer effects in menstrual cup take-up, Working Paper No. 14828, National Bereau of Economic Research, 2009.
[241] S.-H. Yang, B. Long, A. Smola, N. Sadagopan, Z.-H. Zhen, H.-Y. Zha, Like like alike—joint friendship and interest propagation in social networks, in: Proceedings of the 20th International Conference on World Wide Web, ACM Press, New York, 2011.
[242] J. He, W.W. Chu, A social network-based recommender system (SNRS), Annals of Information Systems 12 (2010) 47–74.
[243] J. Sabater, C. Sierra, Review on computational trust and reputation models, Artificial Intelligence Review 24 (2005) 33–60.
[244] A. Jøsang, R. Ismail, C. Boyd, A survey of trust and reputation systems for online service provision, Decision Support Systems 43 (2007) 618–644.
[245] C. Dellarocas, The digitization of word of mouth: promise and challenges of online feedback mechanisms, Management Science 49 (2003) 1407–1424.
[246] J. Golbeck, Generating predictive movie recommendations from trust in social networks, Lecture Notes in Computer Science 3986 (2006) 93–104.
[247] P. Massa, B. Bhattacharjee, Using trust in recommender systems: an experimental analysis, Lecture Notes in Computer Science 2995 (2004) 221–235.
[248] P. Massa, P. Avesani, Trust-aware recommender systems, in: Proceedings of the 2007 ACM conference on Recommender systems, ACM Press, New York, 2007, pp. 17–24.
[249] M. Jamali, M. Ester, TrustWalker: a random walk model for combining trust-based and item-based recommendation, in: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM Press, New York, 2009, pp. 397–406.
[250] J. O'Donovan, B. Smyth, Trust in recommender systems, in: Proceedings of the 10th International Conference on Intelligent User Interfaces, 2005, pp. 167–174.
[251] C.-N. Ziegler, G. Lausen, Analyzing correlation between trust and user similarity in online communities, Lecture Notes in Computer Science 2995 (2004) 251–265.
[252] P. Resnick, R. Zeckhauser, Trust among strangers in internet transactions: empirical analysis of eBay's reputation system, Advances in Applied Microeconomics 11 (2002) 127–157.
[253] S.D. Kamvar, M.T. Schlosser, H. Garcia-Molina, The Eigentrust algorithm for reputation management in P2P networks, in: Proceedings of the 12th International Conference on World Wide Web, 2003, pp. 640–651.
[254] R. Quillian, Semantic memory, in: M. Minsky (Ed.), Semantic Information Processing, MIT Press, 1968, pp. 227–270.
[255] C.-N. Ziegler, G. Lausen, Spreading activation models for trust propagation, In: Proceedings of the IEEE International Conference on e-Technology, e-Commerce and e-Service, EEE'04, 2004, pp. 83–97.
[256] M.P. O'Mahony, N.J. Hurley, G.C.M. Silvestre, Detecting noise in recommender system databases, in: Proceedings of the 11th International Conference on Intelligent User Interfaces, 2006, pp. 109–115.
[257] F.E. Walter, S. Battiston, F. Schweitzer, A model of a trust-based recommendation system on a social network, Autonomous Agents and Multi-Agent Systems 16 (2008) 57–74.

[258] R. Guha, R. Kumar, P. Raghavan, A. Tomkins, Propagation of trust and distrust, in: Proceedings of the 13th International Conference on World Wide Web, 2004, pp. 403–412.
[259] M. Medo, Y.-C. Zhang, T. Zhou, Adaptive model for recommendation of news, EPL 88 (2009) 38005.
[260] T. Zhou, Z.-Q. Fu, B.-H. Wang, Epidemic dynamics on complex networks, Progress in Natural Science 16 (2006) 452–457.
[261] Y. Moreno, M. Nekovee, A.F. Pacheco, Dynamics of rumor spreading in complex networks, Physical Review E 69 (2004) 066130.
[262] M. Medo, G. Cimini, S. Gualdi, Temporal effects in the growth of networks, Physical Review Letters 107 (2011) 238701.
[263] D. Billsus, M.J. Pazzani, Adaptive news access, Lecture Notes in Computer Science 4321 (2007) 550–570.
[264] G. Cimini, M. Medo, T. Zhou, D. Wei, Y.-C. Zhang, Heterogeneity, quality, and reputation in an adaptive recommendation model, The European Physical Journal B 80 (2011) 201–208.
[265] D. Wei, T. Zhou, G. Cimini, P. Wu, W. Liu, Y.-C. Zhang, Effective mechanism for social recommendation of news, Physica A 390 (2011) 2117–2126.
[266] D.-B. Chen, G. Cimini, L. Lü, M. Medo, Y.-C. Zhang, T. Zhou, Adaptive topology evolution in information-sharing scoial networks, arXiv:1107.4491 (2011).
[267] T. Zhou, M. Medo, G. Cimini, Z.-K. Zhang, Y.-C. Zhang, Emergence of scale-free leadership strcuture in social recommender systems, PLoS ONE 6 (2011) e20648.
[268] C.A. Yeung, Analysis of strategies for item discovery in social sharing on the web, in: Proceedings of Web Science Conference 2010.
[269] M. Lipczak, Tag recommendation for folksonomies oriented towards individual users, in: Proceedings of the ECML/PKDD 2008 Discovery Challenge Workshop, Part of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases, 2008, pp. 84–95.
[270] H. Halpin, V. Robu, H. Shepherd, The complex dynamics of collaborative tagging, in: Proceedings of the 16th International Conference on World Wide Web, 2007, pp. 211–220.
[271] Z.-K. Zhang, C. Liu, Y.-C. Zhang, T. Zhou, Solving the cold-start problem in recommender systems with social tags, EPL 92 (2010) 28002.
[272] J.C. Turner, Social influence, Open University Press, Buckingham, UK, 1991.
[273] C. Cattuto, V. Loreto, L. Pietronero, Semiotic dynamics and collaborative tagging, Proceedings of the National Academy of Sciences of the United States of America 104 (2007) 1461–1464.
[274] C. Liu, C.H. Yueng, Z.-K. Zhang, Self-organization in social tagging systems, Physical Review E 83 (2011) 066104.
[275] H. Wu, M. Zubair, K. Maly, Harvesting social knowledge from folksonomies, in: Proceedings of the 7th Conference on Hypertext and Hypermedia, 2006, pp. 111–114.
[276] A. Shepitsen, J. Gemmell, B. Mobasher, R. Burke, Personalized recommendation in social tagging systems using hierarchical clustering, in: Proceedings of the 2008 ACM Conference on Recommender Systems, pp. 259–266.
[277] P. Mika, Ontologies are us: a unified model of social networks and semantics, Lecture Notes in Computer Science 3729 (2007) 5–15.
[278] Y. Song, L. Zhang, C.L. Giles, Automatic tag recommendation algorithms for social recommender systems, ACM Transactions on the Web 5 (2011) 4.
[279] A. Hotho, R. Jäschke, C. Schmitz, G. Stumme, Information retrieval in folksonomies: search and ranking, Lecture Notes in Computer Science 4011 (2006) 84–95.
[280] Z. Xu, Y. Fu, J. Mao, D. Su, Towards the semantic web: collaborative tag suggestions, in: Proccedings of Collaborative Web Tagging Workshop at the 15th International Conference on World Wide Web, 2006.
[281] K.H.L. Tso-Sutter, L.B. Marinho, L. Schmidt-Thieme, Tag-aware recommender systems by fusion of collaborative filtering algorithms, in: Proceedings of the 2008 ACM Symposium on Applied Computing, 2008, pp. 1995–1999.
[282] M.-S. Shang, Z.-K. Zhang, T. Zhou, Y.-C. Zhang, Collaborative filtering with diffusion-based similarity on tripartite graphs, Physica A 389 (2010) 1259–1264.
[283] A. Said, R. Wetzker, W. Umbrath, R. Umbrath, L. Hennig, A hybrid PLSA approach for warmer cold start in folksonomy recommendation, in: Proceedings of Recommender Systems and the Social Web, 2009, pp. 87–90.
[284] X. Si, M. Sun, Tag-LDA for scalable real-time tag recommendation, Journal of Computational Information Systems 6 (2009) 23–31.
[285] R. Krestel, P. Fankhauser, W. Nejdl, Latent Dirichlet allocation for tag recommendation, in: Proceedings of the 3rd ACM Conference on Recommender Systems, 2009, pp. 61–68.
[286] Z.-K. Zhang, T. Zhou, Y.-C. Zhang, Personalized recommendation via integrated diffusion on user-item-tag tripartite graphs, Physica A 389 (2010) 179–186.
[287] T.G. Kolda, B.W. Bader, Tensor decompositions and applications, SIAM Review 51 (2009) 455–500.
[288] S. Rendle, L.B. Marinho, A. Nanopoulos, L.S. Thieme, Learning optimal ranking with tensor factorization for tag recommendation, in: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2009, pp. 727–736.
[289] P. Symeonidis, User recommendations based on tensor dimensionality reduction, Artificial Intelligence Applications and Innovations III (2009) 331–340.
[290] F. Wu, B.A. Huberman, Novelty and collective attention, Proceedings of the National Academy of Sciences of the United States of America 104 (2007) 17599–17601.
[291] P.H. Chou, P.H. Li, K.K. Chen, M.J. Wu, Integrating web mining and neural network for personalized e-commerce automatic service, Expert Systems with Applications 37 (2010) 2898–2910.
[292] Y. Ding, X. Li, Time weight collaborative filtering, in: Proceedings of the 14th ACM International Conference on Information and Knowledge management, ACM Press, New York, 2005, pp. 485–492.
[293] N. Lathia, S. Hailes, L. Capra, Temporal collaborative filtering with adaptive neighbourhoods, in: Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM Press, New York, 2009, pp. 796–797.
[294] P.G. Campos, A. Bellogín, F. Díez, J.E. Chavarriaga, Simple time-biased KNN-based recommendations, in: Proceedings of the Workshop on Context-Aware Movie Recommendation, ACM Press, New York, 2010, pp. 20–23.
[295] P. Wu, C.H. Yeung, W. Liu, C. Jin, Y.-C. Zhang, Time-aware collaborative filtering with the piecewise decay function, 2010, arXiv:1010.3988.
[296] J. Liu, G. Deng, Link prediction in a user–object network based on time-weighted resource allocation, Physica A 39 (2009) 3643–3650.
[297] Y. Koren, Collaborative filtering with temporal dynamics, in: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, SIGKDD'09, ACM Press, New York, 2009, pp. 447–456.
[298] N.K. Lathia, Evaluating collaborative filtering over time, in: Proceedings of the SIGIR 2009 Workshop on the Future of Infomration Retrieval Evaluations, 2009, pp. 41–42.
[299] N. Lathia, S. Hailes, L. Capra, X. Amatriain, Temporal diversity in recommender systems, in: Proceedings of the 33rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR'10, ACM Press, New York, 2010, pp. 210–217.
[300] N.N. Liu, M. Zhao, E. Xiang, Q. Yang, Online evolutionary collaborative filtering, in: Proceedings of the 4th ACM Conference on Recommender Systems, ACM Press, New York, 2010, pp. 95–102.
[301] R. Pastor-Satorras, A. Vázquez, A. Vespignani, Dynamical and correlation properties of the Internet, Physical Review Letters 87 (2001) 258701.
[302] M.E.J. Newman, The first-mover advantage in scientific publication, EPL 86 (2009) 68001.
[303] S. Gualdi, M. Medo, Y.-C. Zhang, Influence, originality and similarity in directed acyclic graphs, EPL 96 (2011) 18004.
[304] S. Gualdi, C.H. Yeung, Y.-C. Zhang, Tracing the evolution of physics on the backbone of citation networks, Physical Review E 84 (2011) 046104.
[305] D.-U. Hwang, M. Chavez, A. Amann, S. Boccaletti, Synchronization in complex networks with age ordering, Physical Review Letters 94 (2005) 138701.
[306] M. Franceschet, PageRank: standing on the shoulders of giants, Communications of the ACM 54 (2011) 92–101.
[307] L. Lü, Y.-C. Zhang, C.H. Yeung, T. Zhou, Leaders in social networks, the delicious case, PLoS ONE 6 (2011) e21202.
[308] J. Zhao, T.-H. Yang, Y. Huang, P. Holme, Ranking candidate disease genes from gene expression and protein interaction: a Katz-centrality based approach, PLoS ONE (2011) e24306.
[309] D. Sun, T. Zhou, J.-G. Liu, R.-R. Liu, C.-X. Jia, B.-H. Wang, Information filtering based on transferring similarity, Physical Review E 80 (2009) 017101.

[310] Y.-K. Yu, Y.-C. Zhang, P. Laureti, L. Moret, Decoding information from noisy, redundant, and intentionally distorted sources, Physica A 371 (2006) 732–744.
[311] P. Laureti, L. Moret, Y.-C. Zhang, Y.-K. Yu, Information filtering via iterative refinement, EPL 75 (2006) 1006.
[312] L.-L. Jiang, M. Medo, J.R. Wakeling, Y.-C. Zhang, T. Zhou, Building reputation systems for better ranking, 2010, arXiv:1001.2186.
[313] Y.-B. Zhou, T. Lei, T. Zhou, A robust ranking algorithm to spamming, EPL 94 (2011) 48002.
[314] Y.-B. Zhou, L. Lü, M.-H. Li, Quantifying the influence of scientists and their publications: distinguish between prestige and popularity, New Journal of Physics 14 (2012) 033033.
[315] M. Medo, J.R. Wakeling, The effect of discrete vs. continuous-valued ratings on reputation and ranking systems, EPL 91 (2010) 48004.
[316] S. Maslov, Y.-C. Zhang, Extracting hidden information from knowledge networks, Physical Review Letters 87 (2001) 248701.
[317] J. Ren, T. Zhou, Y.-C. Zhang, Information filtering via self-consistent refinement, EPL 82 (2008) 58007.
[318] M. Blattner, Y.-C. Zhang, S. Maslov, Exploring an opinion network for taste prediction: an empirical study, Physica A 373 (2007) 753–758.
[319] R. Burke, Hybrid recommender systems: survey and experiments, user model, User Modeling and User-Adapted Interaction 12 (2002) 331–370.
[320] R. Burke, Hybrid web recommender systems, in: P. Brusilovsky, A. Kobsa, W. Nejdl (Eds.), The adaptive web: methods and strategies of web personalization, Springer-Verlag, 2007, pp. 377–408.
[321] A.I. Schein, A. Popescul, L.H. Ungar, D.M. Pennock, Methods and metrics for cold-start recommendations, in: Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 2002, pp. 253–260.
[322] B.P. Carlin, T.A. Louis, Bayes and Empirical Bayes Methods for Data Analysis, Chapman & Hall, London, 1996.
[323] M. Jahrer, A. Töscher, R. Legenstein, Combining predictions for accurate recommender systems, in: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2010, pp. 693–702.
[324] L. Breiman, Bagging predictors, Machine Learning 24 (1996) 123–140.
[325] I.H. Witten, E. Frank, M.A. Hall, Data Mining: Practical Machine Learning Tools and Techniques, 3rd Edition, Morgan Kaufmann/Elsevier, Burlington, 2011.
[326] I. Esslimani, A. Brun, A. Boyer, Densifying a behavioral recommender system by social networks link prediction methods, Social Network Analysis and Mining 1 (2011) 159–172.
[327] G. Caldarelli, M. Cristelli, A. Gabrielli, L. Pietronero, A. Scala, A. Tacchella, Ranking and clustering countries and their products; a network analysis, 2011, arXiv:1108.2590.
[328] R. Burkard, M. Dell'amico, S. Martello, Assignment Problems, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, 2009.
[329] H. Zhu, B.A. Huberman, Y. Luon, To switch or not to switch: Understanding social influence in recommender systems, arXiv: 1108.5147.