CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
Concurrency Computat.: Pract. Exper. 2015; **27**:3915–3942
Published online 9 September 2014 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/cpe.3370

SPECIAL ISSUE PAPER

A survey of recommendation techniques based on offline data processing

Yongli Ren*,†, Gang Li and Wanlei Zhou

School of Information Technology, Deakin University, 221 Burwood Highway, Vic 3125, Australia

SUMMARY

Recommendations based on offline data processing has attracted increasing attention from both research communities and IT industries. The recommendation techniques could be used to explore huge volumes of data, identify the items that users probably like, translate the research results into real-world applications and so on. This paper surveys the recent progress in the research of recommendations based on offline data processing, with emphasis on new techniques (such as *temporal recommendation*, *graph-based recommendation* and *trust-based recommendation*), new features (such as *serendipitous recommendation*) and new research issues (such as *tag recommendation* and *group recommendation*). We also provide an extensive review of evaluation measurements, benchmark data sets and available open source tools. Finally, we outline some existing challenges for future research. Copyright © 2014 John Wiley & Sons, Ltd.

Received 7 January 2014; Revised 27 May 2014; Accepted 20 August 2014

KEY WORDS: recommendation; collaborative filtering; recommender systems

1. INTRODUCTION

With the rapid expansion of e-commerce and Web 2.0 techniques, more products are sold on the Internet, and more customers are willing to provide their ratings on their consumed products. This increasing amount of rating information has stirred much excitement and created abundant opportunities for firms to provide more customized service and reach their profit objectives. *Recommendations based on offline data processing*, a subdiscipline within data mining, refers to the computational techniques for either generating personalized product recommendations for a particular customer or identifying the potential customers for a particular product by offline processing of available customer data, such as *ratings*, *browsing* and *consuming history*.

Since the middle 1990s, the problem of *generating recommendations from data* has been investigated from many perspectives [1, 2]. To address this problem, techniques from *information retrieval* (especially the content-based method) and *user modelling* have been widely utilized [2]. Moreover, the related techniques have been successfully deployed in many practical applications, for example, book recommendations in *Amazon*, movie recommendations in *Netflix*, music recommendations in *Lastfm* and video recommendations in *Youtube*. Some well-known recommender systems are summarized in Table I.

In the past decade, recommendation generation has become an active research topic, and many algorithms have been developed for it. Some attempts to provide surveys on related techniques have been made, for example:

*Correspondence to: Yongli Ren, School of Information Technology, Deakin University, 221 Burwood Highway, Vic 3125, Australia.
†E-mail: yonglitom@gmail.com

Table I. Typical real-world recommender systems.

Application	Representatives	Website
Articles	Google News	http://news.google.com/
	Genieo	http://www.genieo.com/
	CiteULike	http://www.citeulike.org/
	Google Reader	http://reader.google.com/
	StumbleUpon	http://www.stumbleupon.com/
Books	Amazon	http://www.amazon.com/
	Dou Ban Reading	http://www.douban.com/
	Dang Dang	http://www.dangdang.com/
Movies	Netflix	http://www.netflix.com/
	Movielens	http://www.movielens.org/
	Jinni	http://www.jinni.com/
	What To Rent	http://whattorent.com
	Rotten Tomatoes	http://www.rottentomatoes.com/
	Flixter	http://www.flixster.com/
	MTime	http://www.mtime.com/
Music	Dou Ban Radio	http://douban.fm/
	Lastfm	http://www.last.fm/
	Pandora	http://www.pandora.com/
	Mufin	http://www.mufin.com/
	Lala	http://lala.com/
	Emusic	http://www.emusic.com/
	Ping	http://www.apple.com/itunes/ping/
Social network	Facebook	http://www.facebook.com/
	Twitter	http://twitter.com/
Tourism	Wanderfly	http://www.wanderfly.com/
	TripAdvisor	http://www.tripadvisor.com.au/
Videos	Youtube	http://www.youtube.com/
	Hulu	http://www.hulu.com/
	Clicker	http://www.clicker.com/
Others	GetGlue	http://getglue.com/
	Strands	http://strands.com/
	ChoiceStream	http://www.choicestream.com/
	Baifendian	http://www.baifendian.com/

(1) The influential survey by Adomavicius and Tuzhilin, focused on '*The Direction of the Next Generation of Recommender Systems*' up to 2005 [2].
(2) Su and Khoshgoftaar's work reviewed the *collaborative filtering* (CF) techniques [3].
(3) Burke gave an extensive review on hybrid recommender systems [4].
(4) The *recommender systems* handbook provided an overview of concepts, methodologies, challenges and applications in this field [5].

Adomavicius and Tuzhilin's work envisioned the prospective directions of future recommender systems based on research before 2005. Other published surveys mainly focused on one specific recommendation techniques, such as CF [3] and hybrid techniques [4]. The recommender systems handbook includes various topics in the research of recommender systems, such as UIs for recommender systems, conversational systems, cross-domain recommender systems and constraint-based recommender systems [6]. However, no existing survey focused on the techniques that generate recommendations by offline processing of available data, such as users' feedback or expert opinions. Moreover, since the launch of the *Netflix Prize* in 2006, which was working offline, and sought for recommendation techniques that could outperform *Netflix* company's recommender, *Cinematch*, by 10%, and the announcement of the *ACM Conference on Recommender Systems* (ACM RecSys), many research papers that address different aspects of recommendation in various applications have

DOI: 10.1002/cpe

been published. In spite of the aforementioned excellent surveys, a large body of recent important research issues in offline working mode have not been covered in any existing survey. The current paper attempts to fill this gap.

In this paper, we attempt to provide an extensive literature review of related techniques, research issues, measurements, benchmark data sets and open source tools, using the following guidelines:

- We will focus on surveying techniques that generate recommendations based on offline data processing and new research topics in this field that have emerged but not adequately covered in any existing survey, such as *temporal recommendation*, *graph-based recommendation* and *serendipitous recommendation*.
- We will refrain from providing a tutorial overview of any specific topic but will focus on tracing the trends and directions that the research has *undertaken*. Specifically, we start reviewing contributions to each surveyed topic from the very beginning up to now and structure them according to their appearing time so as to provide the development trend. For example, for temporal recommendation, we start with contributions that dates back to 2001. In addition to getting the foundational knowledge on recommender systems, readers can see how each topic was developed so as to understand its undertaken trends and directions.
- We will include more recent references in ACM RecSys, SIGKDD, SIGIR, ICDM, ICML, TKDE and so on, whilst attempting to avoid references from extant surveys.

This section aims to establish a structural organization for the large body of existing literature on recommendations based on offline data processing. The remaining part of this paper is organized as follows:

- Section 2 will provide the basic notations and summarize various assumptions that are commonly used in related research.
- From the perspective of utilizing various information, Section 3 reviews the recommendation generating techniques together with their latest development.
- Because of its practical value, recommendation based on offline data processing has incurred a surge of interest. Section 4 will analyse promising emerging research issues.
- Section 5 will summarize existing recommendation evaluation measurements, together with benchmark data sets. Available open source tools will also be reviewed to help researchers develop new recommendation methods or help practitioners to create real applications.

2. DEFINITIONS AND PROBLEM FORMULATION

A large fraction of applications of recommender system have two basic objects involved: *users* and *items*. Let $\mathcal{U} = \{u_1, u_2, \cdots, u_x, \cdots, u_m\}$ denote a set of m users [‡], and $\mathcal{T} = \{t_1, t_2, \cdots, t_i, \cdots, t_n\}$ denote a set of n product items, such as *movies*, *books*, *CDs* or *hotels*. In the consuming process, users usually unconsciously form opinions about product items they like, do not like and do not even care about. This kind of opinions can be formalized as a *utility function* $f_{utility}$ that measures the possible rating for an item t_i by the active user u_a [2]:

$$f_{utility} : \mathcal{U} \times \mathcal{T} \rightarrow \mathcal{R}^+ \tag{1}$$

where $\mathcal{R}^+$ is the set of positive real numbers. Collectively, all users' utility functions can be summarized into a user–item *rating matrix* $\mathcal{R}$:

$$\begin{bmatrix} r_{11} & \cdots & r_{1i} & \cdots & r_{1n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ r_{x1} & \cdots & r_{xi} & \cdots & r_{xn} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ r_{m1} & \cdots & r_{mi} & \cdots & r_{mn} \end{bmatrix}$$

[‡]These kind of customers are called *users* in the context of recommender systems

DOI: 10.1002/cpe

where r_{xi} denotes the rating on item t_i by user u_x. As it is impractical for a user to rate all items, or for an item to be rated by all users, the rating matrix $\mathcal{R}$ is sparse in almost all cases.

Although users' preferences vary, they do follow some patterns. For example, users tend to like product items that are similar to other items they like; similarly, users tend to like product items that similar users like. These patterns can be utilized to predict such likes and dislikes. Recommendation is all about predicting these patterns of preferences and using them to discover new and desirable items that users did not already know about.

The specific problem addressed by recommendation techniques can now be formally defined. Given a set of users $\mathcal{U}$, a set of product items $\mathcal{T}$ and a sparse rating matrix $\mathcal{R}$, we want to identify the item t'_{u_x} that maximizes the utility value for user u_x [2]:

$$\forall u_x \in \mathcal{U}, \quad t'_{u_x} = \arg\max_{t_i \in \mathcal{T}} f_{utility}(u_x, t_i). \tag{2}$$

This aforementioned function defines a system that suggests only one item. However, there are some other recommender systems that recommend more than one items, for example, *find good items*, *find all items* and *recommend a sequence of items* [6].

2.1. Reference information for recommendation

There are three kinds of reference information involved in the 'recommending' process: users, items and the rating matrix. Recommendations could be generated by utilizing one or several types of these references information.

Each of these objects can be described by some features: For instance, a customer could have demographic information, such as *age*, *gender* and *occupation*; a product could also have attributes, such as a movie's *production time*, *actor*, *director*, *genre* and so on. Accordingly, for any particular item t_i, its content can be represented as a vector $\overrightarrow{v_i}$. Similarly, for a particular user u_x, his or her profile can be represented as a vector $\overrightarrow{p_x}$, which can be based on the demographic information, or typically in many recommender system, just as the aggregation of the content vectors of all those items rated by user u_x. On the basis of this representation, it is possible to evaluate the similarity between users or between items. In this paper, we use $N_k(u_x)$ to represent the set of user u_x's k nearest neighbours and use $L(u_x)$ to represent the list of items recommended to user u_x by some recommendation algorithms.

As the most commonly used information for recommendations based on offline data processing, the sparse rating matrix $\mathcal{R}$ can have different forms:

Numeric rating matrix : When users' preferences are expressed numerically, the rating matrix $\mathcal{R}$ is represented as a $m \times n$ matrix. We can use μ, r_{max} and r_{min} for the overall average rating, the maximum rating and the minimum rating, respectively. Similarly, we use $\bar{r}_x$ and $\bar{r}_y$ to denote the average rating given by users u_x and u_y, respectively. For a particular user u_x, we use $\hat{r}_{xi}$ to denote the predicted rating by user u_x on item t_i.

Binary rating matrix : When users' preferences are expressed as binary, such as users '*installed*' or '*bookmarked*' an item, or when we only consider whether a user rated an item or not, the rating matrix $\mathcal{R}$ could be treated as a binary one, where r_{xi} indicates whether u_x collected t_i. Hence, $r_{xi} = 1$ means that user u_x did collect item t_i; otherwise, $r_{xi} = 0$.

Vector decomposition for rating matrix : The rating matrix $\mathcal{R}$ can be decomposed into a number of row vectors: $\mathcal{R} = [\mathbf{u_1}, \mathbf{u_2}, \cdots, \mathbf{u_x}, \cdots, \mathbf{u_m}]^T$ where $\mathbf{u_x} = [r_{x1}, r_{x2}, \cdots, r_{xi}, \cdots, r_{xn}]$. Similarly, $\mathcal{R}$ can also be decomposed into a number of column vectors: $\mathcal{R} = [\mathbf{t_1}, \mathbf{t_2}, \cdots, \mathbf{t_i}, \cdots, \mathbf{t_n}]$ and $\mathbf{t_i} = [r_{1i}, r_{2i}, \cdots, r_{xi}, \cdots, r_{mi}]^T$, where each column vector $\mathbf{t_i}$ corresponds to the item t_i's rating information by all users. Additionally, in the context of multicriteria rating, the rating matrix on different criteria could be denoted as $\mathcal{R}_i, \mathcal{R}_j, \mathcal{R}_k$ and so on.

List decomposition for rating matrix : When users' preferences are expressed categorically such as *positive*, *neutral* and *negative*, the rating matrix $\mathcal{R}$ can also be decomposed into a set of lists. For example, the user u_x's rating can be represented as a set $R(u_x) = R_{pos}(u_x) \cup R_{neu}(u_x) \cup R_{neg}(u_x)$, where $R_{pos}(u_x)$, $R_{neu}(u_x)$ and $R_{neg}(u_x)$ denote the lists of items on which the user showed a *positive*, *neutral* and *negative* attitude, respectively. In addition, the *common items set*

DOI: 10.1002/cpe

between two users can be defined as $T_{xy} = R(u_x) \cap R(u_y) = \{t_i \in T | r_{xi} \neq \oslash, r_{yi} \neq \oslash\}$, which includes all those items corated by user u_x and user u_y. Similarly, the rating matrix $\mathcal{R}$ can be also decomposed into a set of lists for items: $S(t_i)$ can be used to represent the set of users who rated a particular item t_i.

2.2. Common assumptions

As the most immediately recognizable data mining technique in use, recommendation system has been widely used by companies to recommend books, movies or even research papers based on users' consuming or rating history. They try to infer tastes and preferences and identify unknown items that are of interest for a user. Although users' tastes vary, there are some patterns which can be exploited. One popular assumption is the following:

Assumption 1
If two users u_x and u_y rated a set of items similarly, their rating or consuming behaviour on other items will tend to be similar too.

On the basis of this assumption, recommendations may be generated by analysing a user's rating or consuming history on items. This is the strategy in CF methods. By using this assumption, items consumed by different users could be systematically evaluated for recommendation.

Likewise, as users tend to like things that are similar to other things they like, many recommendation techniques also assume the following:

Assumption 2
Users prefer items similar to those they have liked.

This assumption could simplify the generation of recommendation by tailoring the solution to the relatively small scope of *similar* items, and the content-based similarity can be adopted to identify the most promising items. In fact, this is the foundational strategy in most content-based recommendation methods.

Even though the *temporal dynamics* is well recognized as an important issue, it has not received much attention until recently when it was explicitly addressed in the winning algorithms of the Netflix grand prize. Many recommendation techniques have the following static assumption:

Assumption 3
User's product preference and an item's favouriteness do not change over time.

Although it simplifies the problem, this assumption, when combined with Assumption 2, could limit the recommendations in a narrower field and eventually bore the users so that all recommendations may be discarded. In order to alleviate these problems, two emerging research issues, temporal recommendation and serendipitous recommendation, have been recently investigated. We will review this related work in Sections 3.4 and 4.2, respectively.

2.3. Scope

In the field of recommender systems, there are various research topics, such as evaluation of recommender systems, UIs for recommender systems, conversational systems, cross-domain recommender systems, security issues in recommender systems and constraint-based recommender systems. This survey focuses on techniques that are offline processing rating data bases to generate recommendations. One closely related research topic is recommendations based on real-time (online) data processing. Real-time recommendation techniques basically move the model-building stage to the time when a user is actually consuming an information item. For example, real-time news recommender systems generate recommendations, whilst users are reading news [7].

Recommendation techniques based on offline data processing are very practical and are deployed in most commercial recommender systems, because of its high quality recommendations. However, it is still a challenge to cope with situations with fast changing trends, such as *Twitter* [8]. In this case, real-time recommendation techniques will help. However, the gain in processing time pays the cost

DOI: 10.1002/cpe

of low quality recommendations. To build the recommendation model in real time, it is essential and necessary to take the most recent data into account. This is the basics of real-time recommendation techniques, which also comes with a main issue: These real-time approaches tend to disregard past observations quickly and show a short-term memory. Therefore, high quality recommendations are not guaranteed. The detailed discussion of the advantages and disadvantages of real-time (online) recommendation techniques can be found in literatures [7–9].

3. RECOMMENDATION TECHNIQUES

The aim of recommendation generation is to identify the right items that are likely to be accepted by the users. Conventionally, CF methods produce recommendations based on the rating matrix; whilst *content-based methods* generate recommendations by exploiting regularities in the items' *content*, with or without referring to the rating matrix. More recent recommendation techniques are exploiting extra information such as the *context*, *time* and *social trust*: *Context-based recommendation* methods make recommendations by exploiting the context information in the process of recommendation generation. Temporal recommendation methods treat recommendation generations as a dynamic process over time. Graph-based recommendation methods repose recommendations as a link prediction problem on a bipartite *graph*. *Trust-based methods* provide recommendations by utilizing the social trust information in social networks.

In this section, we will review these recommendation techniques together with hybrid methods that incorporate benefits from some of them.

3.1. Collaborative filtering

When trying to recommend a product item to the active user u_a, CF methods look for users with similar rating pattern and recommend the item that is highly rated by similar users. Under the Assumption 1, CF methods produce recommendations based on, and sometimes only based on, the rating matrix $\mathcal{R}$. Existing CF methods can be categorized into *memory-based* methods and *model-based* methods [10].

3.1.1. Memory-based collaborative filtering. One pioneering work in memory-based CF was proposed by Resnick *et al*. in 1994 [1]. Their proposed system, *GroupLens*, utilized the rating matrix $\mathcal{R}$ to identify users who are similar to the active user and to predict the active user's rating on a particular item.

Two stages are involved in CF: The *neighbour selection* will determine the neighbourhood $N_k(u_a)$ of the active user u_a by evaluating the similarity between users, and the *rating aggregation* will aggregate the ratings from the neighbourhood into a predicted rating on item t_i by the active user u_a.

In the stage of neighbour selection, the neighbourhood $N_k(u_a)$ of user u_a is selected by evaluating the similarity between users. Two popular similarity measurements are the *Pearson correlation coefficient* (PCC) [1] and *cosine similarity* (COS) [10]:

- PCC:

$$sim(u_x, u_y) = \frac{\sum_{t_i \in T_{xy}} (r_{xi} - \bar{r}_x)\left(r_{yi} - \bar{r}_y\right)}{\sqrt{\sum_{t_i \in T_{xy}} (r_{xi} - \bar{r}_x)^2 \sum_{t_i \in T_{xy}} \left(r_{yi} - \bar{r}_y\right)^2}} \tag{3}$$

- COS

$$sim(u_x, u_y) = cos(\mathbf{u_x}, \mathbf{u_y}) = \frac{\sum_{t_i \in T_{xy}} r_{xi} r_{yi}}{\sqrt{\sum_{t_i \in T_{xy}} r_{xi}^2 \sum_{t_i \in T_{xy}} r_{yi}^2}} \tag{4}$$

where $T_{xy} = \{t_i \in T | r_{xi} \neq \oslash, r_{yi} \neq \oslash\}$ denotes the set of items corated by both u_x and u_y, and $\bar{r}_x$ and $\bar{r}_y$ are the average ratings by user u_x and u_y, respectively.

DOI: 10.1002/cpe

In the stage of rating aggregation, for any item t_i, all the ratings on t_i by users in the $N_k(u_a)$ will be aggregated into the predicted rating value $\hat{r}_{ai}$ by user u_a [2]:

$$\hat{r}_{ai} = \begin{cases} \dfrac{1}{k} \displaystyle\sum_{u_x \in N_k(u_a)} r_{xi} & (a) \\ \eta \displaystyle\sum_{u_x \in N_k(u_a)} sim(u_a, u_x) \times r_{xi} & (b) \\ \bar{r}_a + \eta \displaystyle\sum_{u_x \in N_k(u_a)} sim(u_a, u_x) \times (r_{xi} - \bar{r}_x) & (c) \end{cases} \tag{5}$$

where $\eta = \frac{1}{\sum_{u_x \in N_k(u_a)} sim(u_a, u_x)}$ serves as a normalizing factor; $\bar{r}_a$ is the average rating given by user u_a.

As shown in Equation 5(a), the aggregation could be as simple as the *average* function, although more common aggregation can be achieved through the *weighted majority prediction* [11] as in Equation 5(b), where the similarity serves as heuristics to indicate the importance of neighbours to the active user. Equation 5(c) further extended this formula by replacing the absolute rating values with the difference to the average rating by the corresponding users.

3.1.2. Model-based collaborative filtering. Model-based CF algorithms construct a model from the given rating matrix and utilize the model to predict ratings on new items. A wide range of machine learning techniques have been adopted: *supervised learning*, *unsupervised learning* and *matrix decomposition*.

- *Supervised learning-based CF*: For binary rating matrix, Billsus and Pazzani proposed to treat the recommendation as a binary classification problem that can be solved by methods such as *neural networks* [12]. For general rating matrix, Su and Khoshgoftaar applied the *belief nets* to treat it as a multiclass classification problem [13]. In addition, Bayesian classifiers have been used to combine user-based and item-based CF methods, and their results indicated that the combined method can outperform the single method [14].
- *Unsupervised learning-based CF*: By grouping similar users together and using the cluster as the neighbourhood, clustering can improve the scalability of CF methods, although not necessarily produce more accurate results [15]. Biclustering can also be applied to group user and items simultaneously, and this further reduces the computational cost in CF [16]. In addition, Lemire and Maclachlan proposed the *slope one* algorithm, on the basis of the *popularity differential* principle between items, which means user rating behaviour is influenced by both the user's rating style (e.g. the user's average rating) and items popularity (e.g. the average rating for an item) [17].
- *Matrix factorization-based CF*: Billsus and Pazzani proposed to use the SVD to exploit the *latent structure* in user ratings [12]. This method could utilize information from users whose ratings were not correlated with the active user. Since then, many other SVD-based methods have been proposed, such as the *SVD++* method that combines SVD and the neighbourhood-based method, together with the user's explicit and implicit feedbacks [18]. Moreover, Ren *et al*. tackle the data sparsity issue by iteratively refining the projection of each user on a low-rank subspace with the global rating patterns in the $user \times item$ rating matrix [19, 20].

Conventionally, CF methods only make recommendations based on explicit numerical ratings and face several important challenges. One natural issue is the *new user* problem. This refers to the difficulty of making recommendations to a new user who has no rating history. It also suffers from the *new item* problem, which refers to the difficulty of making recommendations when items without any rating are involved. These two problems are collectively referred to as the *cold start* problem in the literature. Moreover, when the size of users and items are huge, the size of rating matrix also increases tremendously, which will further bring the *scalability* issue as another challenge for CF methods.

DOI: 10.1002/cpe

3.2. Content-based recommendation

When trying to recommend a product item to the active user u_a, content-based methods will build a classifier for each user by using various kinds of features of the items then make recommendations for a user without taking models for the other users into consideration. Specifically, it compares the content of each candidate item t_i with the active user's preference profile and recommends the items with a highest degree of commonalities to the active user.

A natural issue in content-based methods is *how to represent the content of items*. Let $\overrightarrow{v_i}$ represent the content of an item t_i, and $\overrightarrow{p_a}$ represent the preference profile of the active user u_a. For text-based item content, $\overrightarrow{v_i}$ could be characterized using methods such as the *term frequency–inverse document frequency*. On the basis of this vector representation for item content, $\overrightarrow{p_a}$ could be further represented by aggregating content vectors of all items rated by user u_a.

Another important issue is *how to measure the commonality with the active user*, which could be expressed as the measurement of similarity between an item and items rated by the active user. Scoring heuristics such as the *cosine* distance have been adopted [21].

$$sim(t_i, u_a) = cos\left(\overrightarrow{v_i}, \overrightarrow{p_a}\right) = \frac{\sum_{k=1}^{N} v_{ik} p_{ak}}{\sqrt{\sum_{k=1}^{N} v_{ik}^2 \sum_{k=1}^{N} p_{ak}^2}} \tag{6}$$

where N is the dimensionality of the item content vector and v_{ik} and p_{ak} are the kth element of $\overrightarrow{v_i}$ and $\overrightarrow{p_a}$, respectively. In contrast to this heuristics, machine learning algorithms have also been used to capture the commonality between items [22].

Traditionally, *content-based recommendation* methods only utilize the item content information and face several vital limitations. Similar to CF methods, content-based recommendation methods suffer from the new user problem, where no rated item content is available for constructing the user's preference profile. Another problem is the limitation in item content analysis. Whilst it is straight forward to extract content information from text documents, it is much more difficult to extract content information from multimedia resources, such as *songs*, *videos* or *pictures*. Finally, one more associated problem is the *diversity* issue, where those recommended items may be limited only to items similar to the active users' rated ones.

3.3. Context-based recommendation

In addition to item content, user profiles and rating history, context information also significantly affect the consumption. For example, for movie recommendation, a user's behaviour may be affected by environment factors such as *when*, *where* and *with whom* to watch the movie. It is well recognized by researchers that context information could further be used to improve the recommendation quality. Consequently, context-based recommendation methods attempt to utilize the context information in the process of recommendation generations.

With the available context information, the recommendation generation could be reposed as the identification of a function $f_{utility}$ over a n-dimensional space, rather than the two-dimensional space [23]:

$$f_{utility} : \mathcal{D}_1 \times \mathcal{D}_2 \times \cdots \times \mathcal{D}_n \rightarrow \mathcal{R}^+ \tag{7}$$

Baltrunas and Ricci's work takes a special form of context information – time – into consideration by splitting the item ratings into subsets according to time-related factors [24], for example, the season when the ratings were produced. However, this method can also be applied to other contextual factors. Recently, Adomavicius and Tuzhilin termed this kind of recommendation as *Context-Aware Recommender Systems* and proposed two algorithmic paradigms to incorporate the contextual information for recommendation generations [25]:

- *Prefiltering (PreF)*: The context information is used to preprocess the data *before* the conventional (2D) recommender techniques are launched. For example, *exact prefiltering* selects recommendations matching the exactly specified context.

DOI: 10.1002/cpe

- *Postfiltering (PoF)*: The context information is used to postprocess the recommendations *after* the conventional (2D) recommender techniques are applied [26]. For example, recommendations can be adjusted according to their relevance to the specified context, and recommendations showing little relevance to the specified context can be eliminated.

Recently, Karatzoglou *et al*. introduced the contextual information into a *tensor factorization*-based collective filtering method for context-aware recommendations [27]. When no explicitly context is available, *how to infer context information* is an interesting issue. One of the early attempt is the Bayesian-based 'contextual' model proposed by Palmisano *et al*. to infer context from existing uncontextual data [28].

In addition to context, other side information such as *tags* is also useful. As *tagging* has been a basic infrastructure in the *World Wide Web*, Zhen *et al*. explored the utility of tagging information in the context of model-based collective filtering, and demonstrated that tags did contain useful information for item recommendation [29].

3.4. Temporal recommendation

Time is an important context information. There are two types of temporal information that are related to users and items. For users, we have *user age* that refers to how long a user has been in a recommender system, *user purchasing time* that is the time when a user rated the item and *user preference pattern* that refers to the user's rating pattern over time. For items, we have *item age* that refers to how long an item has been rated by the users, *item launching time* that is the production time and *item popularity* that could change dramatically in a relative short period [30]. However, temporal information did not attract the research interest that it should deserve until only recently when the *timeSVD++* algorithm played an important role in the winning of the Netflix progress award. Temporal recommendation have emerged as a promising method that further considers the temporal dynamics of data into the recommendation generation process.

Assumption 4
User preferences change over time, and the temporal patterns of the same user's preferences are similar on similar items.

In 2001, Zimdars *et al*. argued that CF should be considered as an univariate time series problem [31]. Tang *et al*. considered the movie's production year in the context of movie recommendations [32] and applied it to reduce the volume of candidate recommendations, targeting the scalability problem [33]. Lee *et al*. investigated the effectiveness of several kinds of temporal information on the prediction accuracy of CF [34]. Moreover, De Pessemier and Dooms investigated the temporal influence on data quality and further argued that old data would be beneficial to a provider-generated content system but would not benefit a user-generated content system [35].

Lathia *et al*. formalized recommendations as a time-dependent iterative prediction problem and found that some algorithms could improve prediction accuracy on the Netflix data set but could not perform similarly on dynamic data sets [36]. Ding and Li proposed to use a personalized decay factor for each user according to his or her rating history and allocate less weights to old data to reflect the importance of time in recommendations [37]. For applications where *the latest products are of importance while old products are trivial*, Ding *et al*. proposed a recency-based collective filtering algorithm by defining a new similarity metric in the context of *concept drift* [38].

Within a recommender system involving multiple product items and users, there is more than one localized characteristic shifting simultaneously, whilst there is only one single concept tracked in concept drift. Hence, Koren emphasized that the temporal models in recommender systems could be different from conventional concept drift [39]. In their proposed *timeSVD++* algorithm, the rating prediction could be formulated as follows:

$$\hat{r}_{xi}(t) = \mu + b_i(t) + b_x(t) + q_i^T \left(p_x(t) + |R(u_x)|^{\frac{1}{2}} \sum_{j \in R(u_x)} y_j \right), \tag{8}$$

DOI: 10.1002/cpe

Table II. Overview of temporal recommendation algorithms.

Time	Representatives	Temporal Type	Context	Application	Goal
2001	Zimdars *et al*.	UPT	CF	Webpage	Improve accuracy
2003	Tang *et al*.	ILT	Hybrid	Movie	Improve accuracy
2005	Tang *et al*.	ILT	Hybrid	Movie	Reduce the size of candidate set
	Ding and Li	UPT	Item-based CF	Movie	Improve accuracy
2006	Ding *et al*.	UPT	Item-based CF	Movie	Improve accuracy
2008	Lee *et al*.	UPT + ILT	CF	Mobile e-commerce	Improve accuracy
2009	Koren	LTP	Model-based CF	Movie	Improve accuracy
	Lu *et al*.	LTP	Model-based CF	Movie and Webpage	Improve accuracy
	Lathia *et al*.	UPT	Memory-based CF	Movie	Showing temporal effect
2010	Lathia *et al*.	UPT	CF	Movie	Measure temporal diversity
	Xiang *et al*.	STP + LTP	Graphs	Research papers and Webpage	Capturing temporal factors
	Xiong *et al*.	UPT	Model-based CF	Movie	Learning global temporal effects
	De Pessemier *et al*.	UPT	CF	Movie	Investigate the time dependency of data quality

UPT, user purchasing time; ILT, item launching time; CF, collaborative filtering.

where μ denotes the overall average rating, $b_i(t)$ is time-changing item bias, $b_x(t)$ is the user-changing bias, q_i and y_j are item vectors in a joint latent factor space f, $R(u_x)$ is the set of items rated by user u_x and $p_x(t)$ is a user vector in a joint latent factor space f, which could capture the changes of user preference over time.

Lu *et al*. investigated the temporal information from the perspective of *matrix factorization*, together with the spatial structure in users' rating history [30]. Moreover, Xiong *et al*. applied tensor factorization to capture the temporal patterns by introducing an additional factor for time [40]. Together with the Bayesian parameter estimation, they proposed a temporal CF method, *Bayesian probabilistic tensor factorization*.

All these temporal recommendation work previously discussed focuses on explicit ratings. However, little work have been carried out on binary data. Amongst them, Xiang *et al*. investigated the temporal effects on implicit feedback (binary) data by using a graph-based method to capture user's time-specific tastes [41]. They proposed the *session-based temporal graph*, in which a user's long-term preference was captured by user–item connections, and short-term preference was captured by session–item connections. Furthermore, Lathia *et al*. argued that conventional accuracy measurements failed to reflect the temporal influence and further proposed an evaluation method for temporal diversity in recommendations [42]. Recently, Ren *et al*. proposed the notion of preference patterns, which characterize the temporal dynamics in users' preferences [43]. An overview of recent research on temporal recommendation is summarized in Table II.

3.5. Graph-based recommendation

From the perspective of network analysis, recommendation can be reposed as a *link prediction* problem on a *bipartite graph*, where the edges are only allowed to connect nodes from different sets: the user set $\mathcal{U}$ and the item set $\mathcal{T}$. When a user u_x collected an item t_i, there is an edge which connects the node u_x and the node t_i. One underlying assumption on graph-based recommendation techniques is the following:

Assumption 5
The fact that a user u_x rated an item t_i indicates that some attribute of item t_i is favoured by user u_x, and this *favourableness* could be distributed through the edges on the bipartite graph.

DOI: 10.1002/cpe

Inspired by the network-based resource allocation dynamics, Zhou *et al*. proposed the *network-based inference* method, which generates recommendations based on the resource-allocation process [44]. For the active user u_a, the favourableness allocated to items collected by u_a is initialized as the following:

$$AC(t_i) = \begin{cases} 1 & \text{if } t_i \in R(u_a) \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

where $R(u_a)$ denotes the set of items rated by user u_a. If considering the favourableness of an item as its allocated resource (e.g. $AC(t_i)$ in Equation 9), this kind of resource could be redistributed amongst all items by the following two steps, and items with highest resources will be recommended to the user u_a.

- *Spreading step*: In this step, all resources will flow from $\mathcal{T}$ to $\mathcal{U}$, and the resource allocated to a user node u_x could be calculated as follows:

$$AC(u_x) = \sum_{i=1}^{n} \frac{c_{xi} AC(t_i)}{|S(i)|}, \tag{10}$$

  where $S(i)$ denotes the set of users who rated item t_i, $AC(u_x)$ is the resources on user node u_x, $AC(t_i)$ is the initial resource allocated to item t_i by user u_x, $c_{xi} = 1$ when user u_x is connected with item t_i and $c_{xi} = 0$ otherwise.
- *Redistribution step*: In this step, the resources will flow back to $\mathcal{T}$, and the final resource allocated to item t_i, $AC'(t_i)$, could be calculated as follows:

$$AC'(t_i) = \sum_{x=1}^{m} \frac{c_{xi} AC(u_x)}{|R(u_x)|}. \tag{11}$$

  Combined with Equation 10, Equation 11 could be rewritten as the following:

$$AC'(t_i) = \sum_{j=1}^{n} w_{ij} AC(t_j) \text{ with } w_{ij} = \frac{1}{|S(j)|} \sum_{x=1}^{m} \frac{c_{xi} c_{xj}}{|R(u_x)|} \tag{12}$$

One limitation in this process is that the initial resource allocated to an item is proportional to its popularity, and this initialization amplifies the effects of popular items [45].

Moreover, the favourableness on the same item rated by different users may be due to different attributes. But in the spreading stage, the same attribute of the item will be counted, and this leads to the problem of *redundant correlation*. A second-order weight matrix could be used to alleviate this problem [46].

In addition to the bipartite graph, when the tag information is considered, the bipartite graph can be extended into a *tripartite* graph, which is a combination of the *Users* × *Items* graph and the *Items* × *Tags* graph. Zhang *et al*. proposed to use this tripartite graph to alleviate the *cold-start* problem when tag information is available [47]. Furthermore, Zhou *et al*. proposed to use *heat spreading* for resource spreading that hybridized network-based inference to address the problem of diversity in the context of recommendations [48].

3.6. Trust-based recommendation

In CF, the neighbourhood selection is based on the similarity between users' profiles, before the neighbour's ratings are aggregated into a predicted value. In real situation, the neighbourhood could also be formed by selecting people whom the active user knew, rather than those users whose profiles are similar. Guy's empirical results indicated that the performance of *familiarity*-based methods was superior for recommendation purposes to the *similarity*-based methods [49]. As the flourishing of social network websites, the social trust relationships reflected in these social networks provides another suitable reference to generate quality recommendations. This is the motivation of *trust-based recommendation* [50], which provides a promising direction to overcome the cold-start problem for both new users and new items.

DOI: 10.1002/cpe

Two kinds of trust measurements exist for a user in the social network: The *global* measurement estimates the credibility of a user in the community, and the *local* measurement estimates the active user's personal trust in a user. For personalized recommendation, it is generally believed that local trust measurements were more suitable, as evidenced by Massa and Avesani [51].

When a social network is available, the trust measurements can be estimated from the social network structure. *MoleTrust*, as used in a real application *Moleskiing.it*, is a local trust on user u_z by the active user u_a [52]:

$$tr_{az} = \frac{\sum_{y \in adj(a)} tr_{ay} tr_{yz}}{\sum_{y \in adj(a)} tr_{ay}} \tag{13}$$

where $\forall y \in adj(a)$ is the set of social network neighbours of user u_a. More specifically, if the active user u_a does not know the user u_z, he or she will ask their social network friends on how much they trust u_z and aggregate their trust by a weighted average; if their friends do not know u_z either, they will continue to ask their friends until there is a link to u_z.

TidalTrust, as used in the real application *FileTrust*, is another local trust measurement based on social network [53]. Golbeck and Manness investigated the effects of path length on trust estimation and found that the shorter propagation paths were, the more accurate the estimated trust would be; whilst the higher the trust involved in the paths, the better the result would be [54]. Hence, *TidalTrust* is estimated in a similar way to *MoleTrust*, as in Equation 13, although it works in a breadth-first fashion. In *TidalTrust*, $adj(a)$ only contains users u_y on the shortest path from u_a to u_y, but $adj(a)$ includes all reachable users u_y through a direct or indirect relation.

In addition, several other trust measurements are also based on social networks, such as *SUNNY* [55] and *Trustwalker* [56]. *Trustwalker*, a recommendation model that combines trust-based and item-based techniques with a random walk model, has been used to improve top-k recommendations [56].

When no social network information is available, how to infer trust from a user's ratings is an essential issue. Chua and Lim proposed the *trust antecedent factor* model, which predicts ratings by using both the rater's and the contributor's ratings [57]. Trust antecedent factor considers the dual property of social network trust and its dependency on ratings, and therefore, it captures the correlation between both trust information and user ratings. Ma *et al*. proposed the *social trust ensemble*, a probabilistic matrix factorization framework that naturally fused a user's preference with their trusted friends' tastes [58].

Besides the trust measurements, Ma *et al*. investigated the usefulness of *distrust* relationships and argued that distrust relations amongst users tended to be as helpful as trust relations [59]. Walter proposed a dynamic trust measurement that took time into consideration [60]. Moghaddam introduced a two-dimensional trust model that can be updated dynamically based on a user's feedback [61]. In addition, Tavakolifard showed that trust information could be transferred within similar situations [62]. In addition, Andersen *et al*. developed a group of five natural axioms that were expected to be satisfied by every trust-based recommendation method; however, they argued that no method could satisfy all five axioms simultaneously, even though it was possible to satisfy any four of them [63].

3.7. Hybrid method

To cope with the limitations in different recommendation methods, efforts have also been devoted to hybrid various methods at different stages of recommendation generation:

- *Information extracting stage*: This kind of methods attempt to integrate various information as used in different recommendation technique. For instance, when the rating matrix is sparse, or no corated item exists between two users, the item content provides a valuable information for the CF to alleviate the cold start problems, such as the *Fab* method [64]. Similarly, the time information could also be utilized to model user's long-term and short-term preferences precisely [41]. Moreover, the social trust information have been successfully incorporated into traditional CF method, whilst the tags information can be integrated into graph-based recommendation method [47].

DOI: 10.1002/cpe

- *Recommendation stage*: Methods in this category attempt to directly aggregate different methods into an unified model. One common strategy is to implement various recommendation techniques separately and produce the final recommendation through voting [65]. Another way is to combine different recommendation techniques directly by applying some mathematical models. For example, Gunawardana and Meek combine *content-based* and CF methods based on a *Boltzmann machine*, in which both the content and collaborative information are represented as features [66]. A recent empirical research on different aggregation methods was carried out by Jahrer *et al*. [67], and their results indicated that linear regression is suboptimal, when compared with methods such as neural network, *bagged gradient boosted decision trees* and *kernel ridge regression*.

In addition, Burke classified hybrid methods according to how different recommendation techniques are combined [4].

4. RESEARCH ISSUES IN RECOMMENDATION GENERATION

In spite of recent advances in recommendation techniques, there are several promising new directions for developing and advancing new recommendation generation research. In this section, we discuss those recently emerged research issues.

4.1. Multicriteria recommendation

Although most recommendation applications are based on single criterion, multicriteria ratings are getting increasing attention recently. For example, *Yahoo! Movies* facilitates users to provide an overall rating together with four detailed ratings on the following criteria: *story*, *acting*, *direction* and *visuals*, while *Zagat's Guide*, a restaurant guide, offers ratings on three criteria: *food*, *decor* and *service*.

Multicriteria recommendation methods treat the generation of recommendation as a *multicriteria decision-making* problem [68–71]:

$$f_{utility} : \mathcal{U} \times \mathcal{T} \rightarrow \mathcal{R}_0^+ \times \mathcal{R}_1^+ \cdots \mathcal{R}_k^+, \tag{14}$$

where $\mathcal{R}_0$ denotes the overall rating and $\mathcal{R}_i, 1 \geqslant i \leqslant k$ denotes the rating for each criteria. Apparently, conventional CF methods are not applicable when more than one ratings are available for each user–item pair.

One approach is to extend the memory-based CF method by aggregating individual criteria into an overall similarity. In general, the similarity between two users on each criteria is calculated separately, and then, the overall similarity between them is produced by aggregating those $k + 1$ individual similarities. Following this, neighbours are identified, and the overall rating estimation could be produced by conventional memory-based CF methods. For example, Adomavicius and Kwon proposed to aggregate using the average and the *worst-case* (minimum) function [72]:

$$sim_{avg}(u_x, u_y) = \frac{1}{k+1} \sum_{i=0}^{k} sim_i(u_x, u_y), \tag{15}$$

$$sim_{min}(u_x, u_y) = min_{i=0:k} sim_i(u_x, u_y), \tag{16}$$

where $sim_i(u_x, u_y)$ denotes the PCC or COS similarity between two users on the ith criteria. Other distance metrics such as *Euclidean* and *Chebyshev* could also be adopted to evaluate user similarities based on multicriteria [72].

On the other hand, multicriteria ratings can also be utilized in model-based methods. For example, Sahoo *et al*. investigated the dependency structure amongst the rating criteria, and they further incorporated this dependency into a *flexible mixture model* [73]. Li *et al*. took the multicriteria ratings into consideration by utilizing *multilinear SVD* to explore the hidden relations amongst users, items and criteria [74].

DOI: 10.1002/cpe

Table III. Overview of serendipitous recommendation algorithms.

Time	Representatives	Method-based	Context	Application
2001	Sarwar *et al*.	Item-based similarity	CF	Movie
2004	Herlocker *et al*.	—	—	Defining the 'serendipitous recommendations'.
2005	Ziegler *et al*.	Topic diversification	CF	Books
	Yang and Li	Degree of interest	Content-based method	Digital library
	Kamahara *et al*.	Clustering	Hybrid method	TV programme
2008	Iaquinta *et al*.	Serendipitous IR	Content-based method	Paintings
2009	Onuma *et al*.	Graph mining	Graph-based method	Movies and articles
	Kawamae *et al*.	Personal innovator degree	CF	Music, video and query
2010	Ge *et al*.	—	—	How to measure 'serendipity'
	Kawamae	User flow probability	CF	Music, video and query

CF, collaborative filtering.

4.2. Serendipitous recommendation

Assumption 2 is a common heuristic in many recommendation methods. However, this assumption is arguable, as McNee *et al*. indicate that improving the accuracy was not the ultimate goal of recommendations [75]. Recently, much research have been devoted to the *serendipity* in recommender systems. Originally proposed by Herlocker *et al*., serendipitous recommendation methods attempt to help the user find a surprisingly interesting item he might not have otherwise discovered [76].

Sarwar *et al*. proposed to capture serendipity by emphasizing those items that were favoured by the active user rather than the whole population [77]. Ziegler *et al*. dealt with the serendipity problem by increasing the diversity of the top-k recommendation list [78]. More specifically, they proposed an *intralist similarity* (ILS) measurement for the diversity:

$$ILS(L_k(u_x)) = \frac{\sum_{t'_i \in L_k(u_x)} \sum_{t'_j \in L_k(u_x), i \neq j} s\left(t'_i, t'_j\right)}{2}, \tag{17}$$

where $L_k(u_x)$ includes the list of k items recommended to user u_x; $s\left(t'_i, t'_j\right)$ is a function: $\mathcal{T} \times \mathcal{T} \rightarrow [-1, +1]$, measuring the similarity between two recommended items t'_i and t'_j. A higher $ILS(L_k(u_x))$ value implies a lower diversity of $L_k(u_x)$.

Responding to top-k recommendation's limitations in exploring the entire product space, Seyerlehner *et al*. proposed to recommend *niche products* [79], which were those hard-to-find items residing in the *long tail* [80]. Kamahara proposed the community-based partial similarity to discover unexpected items for users in the context of TV programme recommendation [81]. Yang and Li defined the concept of *degree of interest* by integrating the amount of information and the novelty in the candidate document, together with its similarity with previously accessed documents in the context of document recommendations [82]. Iaquinta *et al*. introduced a serendipitous information retrieval technique into the content-based recommendation in the context of digital library [83]. By transforming the serendipity recommendation as the node selection on a graph, Onuma *et al*. proposed the TANGENT, a *surprise me* recommendation algorithm [84]. More specifically, TANGENT assigns high weights to nodes that are not only well connected to a user's old choices but also well connected to other alternatives.

Recently, Kawamae proposed the *personal innovator degree* to distinguish *personal innovator* (earlier adopters) from the other users and assigned higher weight to personal innovators so that novel items can be ranked higher in the recommendation list [85]. Furthermore, they extended the personal innovator degree into the *personal innovator probability* that evaluated the *user flow probability* to measure how likely users would purchase an item after they had purchased a previous item [86]. An overview of recent research on serendipitous recommendation is summarized in Table III.

DOI: 10.1002/cpe

Table IV. Overview of aggregation methods.

ⓐ Plurality voting	Applying the 'first past the post', choosing the items that earns the most votes
ⓑ Average	The average of ratings of individuals
ⓒ Multiplicative	The product of individual ratings
ⓓ Borda count	Counting each item's position in an individual's preference list: the item at the bottom earns zero points; the next one up adds one point
ⓔ Copeland rule	The difference of how often one item beats others and how often it loses
ⓕ Approval voting	The number of individuals who rated one item over the threshold
ⓖ Least misery	Taking the minimum of individuals' ratings
ⓗ Most pleasure	Taking the maximum of individuals' ratings
ⓘ Without misery	Avoiding items whose ratings are below a threshold for any individual
ⓙ Fairness	The top items for each individual are selected, treated equally
ⓚ Most respected person	Using the ratings of the most respected member

4.3. Group recommendation

In real-world applications, there are many scenarios where group recommendation is needed, such as *tourist promotion* [87], *holiday packages* [88] and *family food recipe* [89]. *Group recommendation* focuses on producing recommendations that suit a group of users rather than an individual, and several important issues have been identified in this specific context [90]:

- *Group preference identification*: Group recommendation also needs users' specific preferences *explicitly* or *implicitly*. For example in *PolyLens* [91], CF is used to capture users' preferences based on their explicit ratings on previously watched movies. Jameson proposed the *collaborative preference specification* for group members to assimilate, share or consider attitudes so that a successful recommendation can be made [92]. Chao *et al*. argued that identification of negative preferences made more sense than identification of detailed ratings and used *negative preference* in their *AdaptiveRadio* system [93].
- *Group recommendation generation*: Although methods exist for eliciting group preferences, most group recommendation methods generate recommendations by aggregating individual preferences to evaluate the utility of a recommended item for the whole group. Existing methods can be roughly categorized into two schemes:
  - *Aggregating ratings for individuals*: For each candidate item, the ratings by each individual user is predicted, then aggregated into a final group rating and, finally, items with the highest rating are recommended. A set of aggregation methods is summarized in Table IV [94]. Recently, Baltrunas *et al*. proposed to use rank aggregation techniques to produce group recommendations from the recommendations for individuals [95].
  - *Constructing group model*: First, a group preference model is constructed to represent the group preferences as a whole, then the rating for each candidate item by the group preference model is predicted and , finally, items with the highest rating are recommended. For instance, *e-Tourism* also applied intersection and incremental intersection methods to construct a group preference model [87].

  Amongst them, the constructing group model scheme is slightly more popular for its advantage in alleviating the privacy issues.
- *Achieving consensus*: For group recommendation, extensive negotiation amongst group members is normally required. One way is to explain the rational of group recommendation so that individual members can achieve consensus [90]: For instance, *PolyLens* gives the predicted ratings to each member and to the group as a whole. A similar strategy can be found in *Intrigue*, a group recommender system in tourism [96]. *Flytrap* [97] gives the explanation by a visualization that presents who stay in the room when specific music is played. A few group recommender system further provides support for group members to settle on a final decision, for example, the *travel decision forum* [92].

Table V provides a summary of the best known group recommender systems and the major techniques involved. Recently, Masthoff investigated the aggregation of individual user models and

DOI: 10.1002/cpe

Table V. Overview of group recommender systems.

Systems	Feature elicitation	Generating recommendation	Explanation	Achieving consensus
PolyLens	EX	ARI + ⓖ	√	
CATS	IM + CPS	CGM + ⓘ		
Masthoff's recommender	IM	CGM + ⓒ etc.		
Travel decision forum	EX + CPS	CGM + ⓑ	√	√
Intrigue	IM	ARI + ⓑ	√	
Adaptive radio	NP	CGM + ⓘ		
Flytrap	IM	ARI + ⓐ	√	
e-Tourism	EX	CGM + ⓑ		

EX, explicitly; ARI, aggregating ratings for individuals; IM, implicitly; CPS, collaborative preference specification; CGM, constructing group model; NP, negative preference.

various aspects such as *the impact of the recommendation sequence* and *the affective state of individuals* [98]. Furthermore, she claimed that group recommendation techniques could be beneficial for individual recommendations especially on *how to aggregate rating in multiple criteria*, *how to deal with the cold-start problem* and *how to incorporate other users' opinions into the recommendation process*.

4.4. Tag recommendation

Social tagging system has become a basis of infrastructure for modern websites (e.g. *Delicious*, *Flickr*, *Bibsonomy* and *Last.fm*). Collaborative tagging is defined as the process in which users add tags to annotate resources items (e.g. *Web links*, songs, videos and pictures) to speed up the search or to benefit other applications [99]. As a special kind of context information, tag information has been utilized to improve the capacity of recommender systems [100]. Here in this section, we will present the work on recommending tags for resource items. On the basis of the aim of the tagging process, there are two different tag recommendation tasks [101]:

- *Personalized tag recommendation*: This task aims to assist individuals to annotate their own resources for better management and easy retrieval. By projecting the ternary relationship amongst users, items and tags into a lower dimensional space, CF could be applied to generate recommended tags [102]. More content or context information, such as item content, item title, item profile, user profile and user *personomy*[§], can be utilized to generate tags [103, 104]. Hart *et al*. proposed *iTag*, a personalized tag recommendation algorithm for blogs, which made recommendations based on the content of tags previously used by a blogger [105].

  A more popular strategy is based on graphs. Earlier works include *FolkRank* [102], which is adapted from the well-known *PageRank* algorithm. One strong assumption for these tag recommendation is the *CORE p* requirement [102], which requires that *each user, item or tag should appear at least p times in the data set*, although most of the real tagging recommendation problems can not satisfy CORE *p*. Guan *et al*. proposed a framework that modelled tag recommendation as a *query and ranking* problem by using *graph Laplacian*,and applied a graph-based ranking algorithm to rank the tags, considering both an item's relevance and user's preference [106]. Recently, Symeonidis *et al*. proposed a tensor-based tag recommendation algorithm, in which the users–items–tags entities were represented as a three-dimensional tensor unfolded into three matrices and combined in a more dense tensor; finally, tags with weights above a predefined threshold are recommended [107].
- *Collective tag recommendation*: This task aims to recommend tags that benefit searching and browsing so that resource items are more visible to users. Sigurbjornsson and Zwol attempted to generate tag recommendations by exploiting the co-occurrence of tags specified by a user [108]. Heymann *et al*. presented an association rule-based approach, recommending tags with low

[§]The collection of a user's tags.

DOI: 10.1002/cpe

term frequency–inverse document frequency [109]. To deal with the cold start problem in tag recommendation, Krestel *et al*. further introduced *latent Dirichlet allocation* to extract latent topics from resource items to which new items could be mapped [101] so that tags within the recommended topics could be identified and recommended to the new resource item. When the content of resources is available, tag recommendation could also be treated as a classification problem [110].

On the basis of the kind of information used, existing tag recommendation methods can be categorized into two groups: *content-based tag recommendation*, which focuses on the content of tags or items, and *model-based tag recommendation*, which exploits data mining techniques to capture the relationship amongst tags, items and users. An overview of existing tag recommendation algorithms is summarized in Table VI.

Moreover, other issues on tag recommendation have also been investigated. As tags are usually defined freely and informally by users, it can result in consequent ambiguity and redundancy. By using a clustering-based method, Gemmell *et al*. attempted to determine the influence of ambiguity and redundancy of tags in *folksonomies* [111], which are the collection of all *personomies* form the *folksonomy*. With the attempt to evaluate tag recommendation performance, Parra and Brusilovsky compared several user-based CF methods in the tagging systems of *CiteULike* and claimed that the rating scale should be treated very carefully when applying classical CF methods on social tagging systems [112].

4.5. Others

In addition to the aforementioned new research directions, in the following, we will present other promising research topics that have emerged recently.

4.5.1. Recommendation explanation. It has been well recognized that explanation is an important facility of recommender systems. For example, in group recommendation, it plays a key role in group consensus. On recent overview of designing and evaluating explanations can be found in [113]. Sinha and Swearingen investigated the contribution of explanations of how recommendations were generated to the system *transparency* [114]. Bilgic and Mooney investigated three explanation approaches in book recommendations and argued that explanations should be measured according to how accurately they could help users find their true opinions [115]. Tintarev and Mashoff investigated the possible advantages of explanations in recommendation systems [116].

Table VI. Overview of existing tag recommendation algorithms.

	Content-based method	Model-based method
Personalized tag recommendation	Commonly used techniques: · Co-occurrence of tags Representative algorithms: · Lipczak · Lipczak *et al*. · Hart *et al*.	Commonly used techniques: · Bayesian approach · Neural network · Graph theory · Matrix factorization Representative algorithms: · Jaschke *et al*. · Symeonidis *et al*. · Guan *et al*.
Collective Tag Recommendation	Commonly used techniques: · Co-occurrence of tags · Clustering · TF-IDF Representative Algorithms: · Sigurbjornsson and Zwol · Song *et al*.	Commonly used techniques: · Latent Dirichlet allocation · Association Rules · TF-IDF Representative Algorithms: · Krestel *et al*. · Heymann *et al*.

TF-DIF, term frequency–inverse document frequency.

DOI: 10.1002/cpe

Symeonidis *et al*. proposed a recommender system with explanations by exploiting content data and *biclustering* techniques [117].

4.5.2. Secure recommendation. The application of recommender systems also raises problems regarding *malicious attack*, which refers to the possibility of applying strategies to manipulate the output of recommender systems deliberately and maliciously. Some attack models have been proposed and a number of detection methods have also been investigated and evaluated. For instance, Mehta and Nejdl investigated the shilling detections in CF and proposed a PCA-based detection strategy [118]. Hurley *et al*. proposed a statistical detection model that detected more advanced attacks [119], whilst Cheng and Hurley discussed the vulnerabilities of model-based CF methods [120].

4.5.3. Latent information. The quest for more accurate recommendations does not stop inspiring researchers to pursue further understandings of data in the context of generating recommendations. One important aspect is what kinds of valuable features are extractable from the data. Some latent information can be further utilized to improve the quality of recommendations, for example, Wang *et al*. investigated the *latent aspect rating analysis* problem, which refers to analysing the opinions expressed in the form of text topical reviews about an entity [121]. They investigated the latent aspect rating analysis problem in the context of hotel review data set and proposed a regression-based model. Another not less important aspect is seeking implicit user feedback. One kind of feedback concerns that items a user has collected [18, 39], rather than ratings expressed by users. In addition, Pilaszy investigated the usefulness of ratings and other metadata and claimed that even a few ratings are more valuable than other metadata [122]. All the aforementioned algorithms focus on the performance of recommendation by improved or enhanced ways of modelling a user's behaviours, preferences and interests.

5. RECOMMENDATION EVALUATION AND OPEN SOURCE PLATFORMS

In this section, we will review the recommendation evaluation measurements and available open source tools, also summarize existing benchmark data sets which have been extensively used in research.

5.1. Evaluation measurements

5.1.1. Accuracy-related measurements. When evaluating the quality of recommendations, most recommendation techniques focus on *predictive accuracy*, which evaluates the difference between the known ratings and the predicted ratings or between the recommended items set and the items set consumed by a user.

- *Regression error*: As most recommendation techniques consider recommendation as a classification or regression task on the rating, the regression error is the most common evaluation measurement. Two basic errors exist: *mean absolute error* (MAE) averages the differences between true ratings and predicted ratings and *root-mean-squared error* (RMSE) penalizes large errors by amplifying the difference between the predicted ratings and the true ratings. Their definitions are as follows:

$$MAE = \frac{\sum_{x,i} |\hat{r}_{xi} - r_{xi}|}{\sum_x |R(u_x)|} \tag{18}$$

$$RMSE = \sqrt{\frac{\sum_{x,i} (\hat{r}_{xi} - r_{xi})^2}{\sum_x |R(u_x)|}}, \tag{19}$$

  where $\sum_x |R(u_x)|$ is the total number of known ratings over all users. Because of variant rating scales (such as [1,5] versus [1,10]), both measurements can be normalized so that the final measurement is in the form of percentage of the full scale [123, 124].

DOI: 10.1002/cpe

Learning methods that minimize regression error focus on judging the correctness of individual item recommendations. Two methods may not perform significantly different with respect to regression error but they can produce recommendations that are significantly different from another point of view, for example, in terms of diversity or novelty. For example, it has also been found that a small improvement in RMSE may generate a huge difference in the ranking of items, which is also related to top-k recommendations [18].

- *Top-k hit*: When the rating matrix satisfies the assumption of *missing not at random*, the top-k hit rate is a natural accuracy measurement for recommendations [125]:

$$Top_{u_x}(k) = \frac{|R(u_x) \cap L_k(u_x)|}{|R(u_x)|}, \tag{20}$$

where $|R(u_x)|$ is the number of relevant items rated by user u_x, $L_k(u_x)$ is the k recommended item list to user u_x, so $|R(u_x) \cap L_k(u_x)|$ denotes the number of relevant items in the recommendation list $L_k(u_x)$ for user u_x. Compared with regression errors, top-k hit deals with missing ratings inherently [125].

- *Accuracy*: For applications where the rating matrix $\mathcal{R}$ is binary, the objective of recommendation is to recommend items that a user would like to adopt, for example, *bookmark a webpage* or *instal an add-on*, and the recommendation result can be partitioned into *positive tuples* (items used by the user) and *negative tuples* (items the user did not use) [126], and four terms can be further defined: *TruePositive* (TP) where the positive tuples correctly recommended, *FalseNegative* (FN) where the positive tuples incorrectly filtered, *FalsePositive* (FP) where the negative tuples incorrectly recommended and *TrueNegative* (TN) where the negative tuples correctly filtered. Four accuracy measures could be then defined as follows:

$$precision = \frac{TP}{TP + FP} \tag{21}$$

$$sensitivity = \frac{TP}{TP + FN} \tag{22}$$

$$specificity = \frac{TN}{FP + TN} \tag{23}$$

$$falsepositiverate = \frac{FP}{FP + TN} \tag{24}$$

where *precision* is the percentage of items recommended to the user, which is actually rated by the user; *sensitivity* is the *true positive rate* (*TPR*), which is the percentage of positive tuples that are correctly recommended to the user; *specificity* is the *true negative rate*, which is the percentage of negative tuples that are correctly filtered, and the *false positive rate* (*FPR*) is the percentage of negative tuples incorrectly recommended.

5.1.2. Diversity-related measurements. Although accuracy measurement greatly helps the field of recommendations, it does not evaluate the serendipity of items to users or against user expectations [127]. To fill this void, several measurements have been proposed recently based on diversity, which has three facets of meaning: (i) recommendations to different users should be different; (ii) recommendations should contain unexpected items, rather than just items similar to those known by the users; and (iii) because a user's preference could change over time, it is helpful if *serendipitous* items were recommended.

- *Personalization*: reflects the interuser diversity by the uniqueness of each user's recommendation list [45].

$$Personalization = \frac{2}{m(m-1)} \sum_{x \neq y} \left(1 - \frac{|L_k(u_x) \cap L_k(u_y)|}{|L_k(u_x)|}\right), \tag{25}$$

DOI: 10.1002/cpe

where m is the number of users, $\left(1 - \frac{|L_k(u_x) \cap L_k(u_y)|}{|L_k(u_x)|}\right)$ is the *Hamming* distance between recommendation lists $L_k(u_x)$ and $L_k(u_y)$ for user u_x and u_y, respectively.

- *Novelty* measures the quality of recommendations in the perspective of novel and unexpected results. It implies that the recommended items include less popular ones, rather than popular ones the user already knows from other ways. Zhou *et al*. proposed a simple novelty measurement that averaged the collected times over all recommendations [45]:

$$Novelty = \frac{1}{m \times k} \sum_{x=1}^{m} \sum_{t_i \in L_k(u_x)} S(i), \tag{26}$$

where m is the number of users, k is the length of the recommendation list and $S(i)$ is the number of users who rated item t_i.

- *Serendipity* measures the degree to which the recommended items are surprising and attractive to the active user [76]. There are a number of variants to the serendipity definition. For instance, Shani and Gunawardana defined it as how *surprising* and *successful* the recommendations are [124]. Because of the subject sense of serendipity, Ge *et al*. assumed a primitive prediction model that generated high *ratability* and low *unexpectedness* and proposed a serendipity measurement to capture the *unexpectedness* and *usefulness* of recommendations [127]:

$$Serendipity = \sum_{x=1}^{m} \sum_{t_i \in L_k(u_x) \setminus PM} \frac{useful(t_i)}{k}, \tag{27}$$

$$useful(t_i) = \begin{cases} 1 & \text{if } t_i \text{ is considered useful to } u_x \\ 0 & \text{otherwise} \end{cases} \tag{28}$$

where PM is the recommendations generated from the primitive prediction model.

5.2. Benchmark data sets

In the context of recommender system research, a number of benchmark data sets have been used. Table VII gives a summary of these data sets, together with their size and content information.

Table VII. Overview of the benchmark data sets.

Data set	Ratings	Items	Users	Item Contents	User demographics	Time	Tags
			Explicit rating data set				
Bookcrossing	1 149 780	271 379	278 858	√	√		
Jester	Over 4.1 million	100	73 421	√			
	Over 1.7 million	150	63 974	√			
Movielens	100 000	1 682	943	√	√	√	
	1million	3 706	6 040	√	√	√	
	10million	10 681	71 567	√		√	√
Yahoo! Webscope R1	11 557 943	98 211	1 948 882		√		
Yahoo! Webscope R2	Over 717 million	136 000	1.8M	√			
Yahoo! Webscope R3	354 000	1 000	15 400				
Yahoo! Webscope R4	211 231	11 915	7 642	√	√		
Netflix	Over 100 million	17 770	48 000	√		√	
			Implicit binary data set				
CiteULike	NA			√		√	√
Delicious	132 500 391	50 222 260	947 835	√		√	√
Bibsonomy	NA			√		√	√

DOI: 10.1002/cpe

According to the types of user feedback captured in the data set, existing benchmark data set can be categorized into two categories.

5.2.1. Numeric rating data sets. In these data sets, a user's feedbacks are represented as numerical ratings. For example, in the *Movielens* data set, users' ratings are integers in the range from 1 to 5. In *Yahoo! Webscope R*4 data set, users' ratings are integers in the range from 1 to 13. Here, we briefly introduce common benchmark data sets in this category:

- *BookCrossing (http://www.informatik.uni-freiburg. de/∼cziegler/BX)*: Collected by Ziegler from the *Book-Crossing* community between August and September in 2004, *BookCrossing* contains 1 149 780 ratings on 271 379 books by 278 858 users, together with user demographic information such as *location* and age.
- *Jester (http://eigentaste.berkeley.edu)*: Two rating data sets were released by Ken Goldberg from the *Jester Joke recommender* system in UC Berkeley. One contains over 4.1 million ratings on 100 jokes by 73 421 users, and the other is over 1.7 million ratings on 150 jokes by 63 974 users, together with the raw text of these jokes.
- *MovieLens (http://www.grouplens.org)*: Collected by the *GroupLens* Group from the *University of Minnesota*, *MovieLens* data sets have been widely used [128–131]: One data set includes 100 000 ratings collected from 943 users on 1682 movies, the second one contains 1 million ratings on 3900 movies by 6040 users and the third one contains 10 million ratings on 10 681 movies by 71 567 users with 100 000 tags. Item content, user *demographics* and time are available for the first two data sets.
- *Yahoo! Webscope (http://research.yahoo.com)*: *Yahoo! Webscope* includes four data sets:
  - *R1* is a *Yahoo! Music* data set, containing *Yahoo!* users' rating for musical artists that are gathered within a one month period before March 2004. There are 11 557 943 ratings on 98 211 artists by 1 948 882 anonymous users, together with the names of musical artists involved.
  - *R2* is another *Yahoo! Music* data set, containing *Yahoo!* users' rating for songs, together with each song's *artist*, *album* and *genre*. There are over 717 million ratings for 136 000 songs by 1.8 million users.
  - *R3* includes ratings for songs data collected from two different sources. One includes ratings given by users in a normal interaction, whilst another one includes ratings gathered in an online survey. As a summary, there are around 300 000 user-supplied ratings from normal interactions and 54 000 ratings on randomly selected songs in the survey. The survey and responses are also provided.
  - *R4* is a movie-rating data set, which is collected by *Yahoo! Movies* on or before November 2003. The original rating scale is from $A+$ to F, but it has been converted into the range from 1 to 5. a large amount of movie description information and some user demographics, such as *gender* and *birth year* are available.
- *Netflix data (http://www.netflixprize.com)*: Collected from October 1998 to December 2005, this data includes over 100 million ratings on 17 770 movies by over 480 000 anonymous users. Together with ratings, the time when ratings were produced was also available, as well as the *title* and *release year* of each movie.

5.2.2. Binary data sets. In a binary data set, users' ratings were not represented numerically but were in the binary form of '*like*' or '*dislike*', '*booked*' or '*not booked*'. Here, we briefly describe popular implicit binary data sets as follows:

- *CiteULike data (http://www.citeulike.org/faq/data.adp)*: This data set is collected from the *CiteULike* website, which provides the service for users to bookmark, tagging and share scholarly articles. As an automatic programme is running on a daily basis to produce a snapshot summary of what articles have been posted with what tags by whom, the size of this data set is dynamically changing.

DOI: 10.1002/cpe

- *Delicious data (http://www.dai-labor.de)*: This data set is collected from the social bookmarking service website *Delicious*. Retrieved from December 2007 to April 2008, it contains 132 500 391 bookmarks of 50 222 260 webpages by 950 000 users, together with 420 million tag assignments. As items content could be retrieved by the corresponding URL, this data set can also be used for content-based recommendation.
- *Bibsonomy (http://www.bibsonomy.org)*: Known as *Benchmark Folksonomy Data from BibSonomy*, it was collected by *Knowledge and Data Engineering* Group in the *University of Kassel*. Several versions of data sets are provided periodically, for example, *version of 30 June 2006*, *version of 1 January 2011*. The data set contains information such as *who posts tags* to *what* at *what time*, together with tag–tag relations amongst users.

5.3. Open source tools

In the last two decades, the need for automatic generation of recommendation from huge volumes of data has been widely recognized and has led to significant development of open source tools that can help researchers or practitioners explore existing recommendation techniques. Representative open source tools include the following:

- *Duine Toolkit (http://sourceforge.net/projects/duine)*: Developed by *Telematica Institute*, *Novay*, the *Duine* Toolkit contains a set of *Java* libraries that allow users to develop recommender engines. One of Duine's well-known aspects is its framework, which includes modules for *user profile*, *feedback* and *explanation*. Moreover, the Duine Toolkit implemented a collection of recommendation algorithms, including *contest-based* algorithms and CF algorithms. It allows users to develop and add their own algorithms and also offers a variety of ways to combine existing techniques.
- *Cofi (http://www.nongnu.org/cofi)*: As a Java-based CF library, Cofi is developed by *Daniel Lemire* as a foundation of existing recommendation algorithms, although other aspects such as user profile and explanation are missing.
- *Taste (http://taste.sourceforge.net)*: *Taste* is another Java-based CF Library, but with *Enterprise Java Beans* support. Currently, *Taste* is developed as part of the *Apache Mahout* project and implemented a variety of recommender methods.
- *C/Matlab Toolkit* (http://www-2.cs.cmu.edu/∼ Lebanon): Maintained by *Guy Lebanon*, this toolkit contains a set of CF methods in *C/Matlab*, together with evaluating functions. It covers memory-based methods with a variety of similarity measurements, and the model-based methods for *personality diagnosis*.

6. CONCLUSIONS

The ever-increasing amount of user-generated rating data in different contexts and environments presents both an opportunity to recommend product items that are the most likely to be welcomed by users and a challenge to utilize available data effectively. As a young and promising field, automated recommendation generation still faces a number of unresolved problems. Here, we list and highlight issues that are important to future work:

- *Data sparsity*: This is probably the most difficult issue for recommendation techniques. In general, the more item and user information that a recommendation algorithm can utilize, the more appropriate the recommendations will be. Unfortunately, the item and user information are not always available, and users tend to only express preference on a small fraction of items. For instance, although there are around 100 million ratings in the Netflix data, it was given on 17 700 movies by 48 000 users, with the density only 1.17%. In order to generate satisfied recommendations with sparse data, specific techniques need to be designed.
- *Temporal dynamics*: As the item popularity and the user preference change over time [39, 41], how to exploit the effectiveness of temporal dynamics in the context of recommendation generation remains an open challenge.

DOI: 10.1002/cpe

- *Complexity*: It has been illustrated in several research works that recommendation generation is a complex process, involving a large number of variables even for a simple recommendation. To make a recommendation, many different information may be needed. Even more complex is the importance of the information in different contexts. Specifically, recommendation research has been well-investigated only in several fields, such as *movie*, *book* and *article* recommendation, whilst effective approaches how to recommend is still to be found for other fields.
- *Eccentrics*: It refers to both items that users either like or hate, and users whose preference significantly differ from any group of users, such as the movie *Napoleon Dynamite* in the *Netflix Prize*. This kind of items or users tends to be unpredictable. Although some work has been carried out on this issue [132, 133], how to properly deal with *eccentrics* still needs extra attention.
- *Open source platform*: No recommendation research platform is as commonly used as the *WEKA* system for *machine learning* research. Existing tools varies in terms of their coverage of recommendation techniques. Most of them focus on CF techniques, but no open source tool is yet available for recent research advances in emerging topics as reviewed in Section 4. Moreover, as each open source tool uses its own file format, it is difficult to compare different algorithms.

As an emerging research area, generating recommendations by offline processing of available customer data has its own issues, newly developed methods and contributions from researchers in a variety of fields. Nevertheless, this paper attempts to provide a comprehensive survey on recommendation techniques developed recently. Given the broad spectrum of viewpoints, this survey is inevitably limited in scope, but we hope it will assist those new to the field to gain a clearer understanding of recommendation techniques and lead to a better understanding of the variety of techniques and approaches in this multidisciplinary field.

REFERENCES

1. Resnick P, Iacovou N, Suchak M, Bergstrom P, Riedl J. GroupLens : An open architecture for collaborative filtering of netnews. *Proceedings of the 1994 ACM Conference on Computer supported Cooperative Work*, Chapel Hill, NC, USA, 1994; 175–186.
2. Adomavicius G, Tuzhilin A. 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* 2005; **17**(6):734–749.
3. Su X, Khoshgoftaar TM. A survey of collaborative filtering techniques. *Advances in Artificial Intelligence* 2009; **2009**:19. doi:10.1155/2009/421425.
4. Burke R. Hybrid recommender systems: survey and experiments. *User Modeling and User-Adapted Interaction* 2002; **12**(4):331–370.
5. Ricci F, Rokach L, Shapira B, Kantor PB (eds.) *Recommender Systems Handbook*. Springer: New York, NY, USA, 2010.
6. Ricci F, Rokach L, Shapira B. Chapter 1 introduction to recommender systems handbook. In *Recommender Systems Handbook*. Springer: US, 2011; 1–35. doi:10.1007/978-0-387-85820-3.
7. Abbar S, Indyk P. Real-time recommendation of diverse related articles. *Proceedings of the 22nd International Conference on World Wide Web*, Rio de Janeiro, Brazil, 2013; 1–12.
8. Diaz-Aviles E, Drumond L, Schmidt-Thieme L, Nejdl W. Real-time top-*n* recommendation in social streams. In *Proceedings of the 6th ACM Conference on Recommender Systems*. ACM Press: New York, USA, 2012; 59–66. doi:10.1145/2365952.2365968.
9. Chandramouli B, Levandoski JJ, Eldawy A, Mokbel MF. StreamRec : a real-time recommender system. *Proceedings of the 2011 ACM Sigmod International Conference on Management of Data*, Athens, Greece, 2011; 6–8.
10. Breese J, Heckerman D, Kadie C. Empirical analysis of predictive algorithms for collaborative filtering. *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, Madison, Wisconsin, 1998; 43–52.
11. Goldman S, Warmuth MK. Learning binary relations using weighted majority voting. *Machine Learning* September 1995; **20**(3):245–271. doi:10.1007/BF00994017.
12. Billsus D, Pazzani MJ. Learning collaborative information filters. *Proceedings of the 15th International Conference on Machine Learning*, San Francisco, CA, USA, 1998; 46–54.
13. Su X, Khoshgoftaar T. Collaborative filtering for multi-class data using belief nets algorithms. In *Proceedings of the 18th IEEE International Conference on Tools with Artificial Intelligence*. IEEE: Washington, DC, USA, November 2006; 497–504.
14. Miyahara K, Pazzani MJ. Improvement of collaborative filtering with the simple Bayesian classifier. *Information Processing Society of Japan* 2002; **43**(11):3429–3437.

DOI: 10.1002/cpe

15. Sarwar B, Karypis G, Konstan J. Recommender systems for large-scale e-commerce: scalable neighborhood formation using clustering. *Proceedings of the 5th International Conference on Computer and Information Technology East West University*, Dhaka, Bangladesh, 2002.
16. Hofmann T, Puzicha J. Latent class models for collaborative filtering. *Proceedings of the 16th International Joint Conference on Artificial Intelligence*, Stockholm, Sweden, 1999; 688–693.
17. Lemire D, Maclachlan A. Slope one predictors for online rating-based collaborative filtering. *Proceedings of the 2005 SIAM International Conference on Data Mining*, Newport Beach, California, 2005; 471–475.
18. Koren Y. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Ddata Mining*. ACM: New York, NY, USA, 2008; 426–434.
19. Ren Y, Li G, Zhou W. Learning rating patterns for top-*n* recommendations. *Proceedings of the 2012 IEEE/ACM International Cconference on Social Networks Analysis and Mining (ASONAM)*, Istanbul, Turkey, 2012; 472–479.
20. Ren Y, Li G, Zhou W. A learning method for top-*n* recommendations with incomplete data. *Social Network Analysis and Mining* 2013; **3**(4):1135–1148.
21. Baeza-Yates R, Ribeiro-Neto B. *Modern Information Retrieval*. Addison-Wesley Longman Publishing Co., Inc.: Boston, MA, USA, 1999.
22. Billsus D, Pazzani M. User modeling for adaptive news access. *User Modeling and User-Adapted Interaction* 2000; **10**(2):147–180.
23. Adomavicius G, Tuzhilin A. Multidimensional recommender systems: a data warehousing approach. In *Electronic Commerce*, vol. 2232, Fiege L, Mhl G, Wilhelm U (eds)., Lecture Notes in Computer Science. Springer Berlin Heidelberg: London, UK, 2001; 180–192.
24. Baltrunas L, Ricci F. Context-based splitting of item ratings in collaborative filtering. *Proceedings of the 3rd ACM Conference on Recommender Systems*, New York, New York, USA, 2009; 245–248.
25. Adomavicius G, Tuzhilin A. Context-aware recommender systems. *Proceedings of the 2nd ACM Conference on Recommender Systems*, Lausanne, Switzerland, 2008; 335–336.
26. Panniello U, Tuzhilin A, Gorgoglione M, Palmisano C, Pedone A. Experimental comparison of pre- vs. post-filtering approaches in context-aware recommender systems. In *Proceedings of the 3rd ACM Conference on Recommender Systems*. ACM: New York, NY, USA, 2009; 265–268.
27. Karatzoglou A, Amatriain X, Baltrunas L, Oliver N. Multiverse recommendation: *n*-dimensional tensor factorization for context-aware collaborative filtering. In *Proceedings of the 4th ACM Conference on Recommender Systems*. ACM: New York, NY, USA, 2010; 79–86.
28. Palmisano C, Tuzhilin A, Gorgoglione M. Using context to improve predictive modeling of customers in personalization applications. *IEEE Transactions on Knowledge and Data Engineering* November 2008; **20**(11): 1535–1549.
29. Zhen Y, Li W, Yeung D. TagiCoFi : tag informed collaborative filtering. *Proceedings of the 3rd ACM Conference on Recommender Systems*, New York, New York, USA, 2009; 69–76.
30. Lu Z, Agarwal D, Dhillon IS. A spatio-temporal approach to collaborative filtering. *Proceedings of the 3rd ACM Conference on Recommender Systems*, New York, New York, USA, 2009; 13–20.
31. Zimdars A, Chickering DM, Meek C. Using temporal data for making recommendations. *Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence*, UAI '01, Seattle, Washington, USA, 2001; 580–588.
32. Tang T, Winoto P, Chan K. On the temporal analysis for improved hybrid recommendations. In *Proceedings of the IEEE/WIC International Conference on Web Intelligence*. IEEE Computer Society: Washington, DC, USA, 2003; 4–10.
33. Tang TY, Winoto P, Chan KCC. Scaling down candidate sets based on the temporal feature of items for improved hybrid recommendations. *Proceedings of the 2003 International Conference on Intelligent Techniques for Web Personalization*, ITWP'03, Acapulco, Mexico, 2005; 169–186. doi:10.1007/11577935_9.
34. Lee T, Park Y, Park Y. A time-based approach to effective recommender systems using implicit feedback. *Expert Systems with Applications* May 2008; **34**(4):3055–3062. doi:10.1016/j.eswa.2007.06.031.
35. De Pessemier T, Dooms S, Deryckere T, Martens L. Time dependency of data quality for collaborative filtering algorithms. In *Proceedings of the 4th ACM Conference on Recommender Systems*. ACM: New York, NY, USA, 2010; 281–284.
36. Lathia N, Hailes S, Capra L. Temporal collaborative filtering with adaptive neighbourhoods. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM: New York, NY, USA, 2009; 796–797.
37. Ding Y, Li X. Time weight collaborative filtering. *Proceedings of the 14th ACM International Conference on Information and Knowledge Management*, Bremen, Germany, 2005; 485–492.
38. Ding Y, Li X, Orlowska M. Recency-based collaborative filtering. In *Proceedings of the 17th Australasian Database Conference*. Australian Computer Society, Inc.: Darlinghurst, Australia, 2006; 99–107.
39. Koren Y. Collaborative filtering with temporal dynamics. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM: New York, NY, USA, 2009; 447–456.
40. Xiong L, Chen X, Huang T, Schneider J, Carbonell J. Temporal collaborative filtering with Bayesian probabilistic tensor factorization. *Proceedings of the SIAM International Conference on Data Mmining*, Columbus, Ohio, USA, 2010; 211–222.

DOI: 10.1002/cpe

41. Xiang L, Yuan Q, Zhao S, Chen L, Zhang X, Yang Q, Sun J. 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: New York, NY, USA, 2010; 723–732.
42. Lathia N, Hailes S, Capra L, Amatriain X. Temporal diversity in recommender systems. In *Proceeding of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM: New York, NY, USA, 2010; 210–217.
43. Ren Y, Zhu T, Li G, Zhou W. Top-n recommendations by learning user preference dynamics. *Proceedings of the 17th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, Gold Coast, Australia, 2013; 390–401.
44. Zhou T, Ren J, Medo M, Zhang YC. Bipartite network projection and personal recommendation. *Physical Review E* October 2007; **76**:046115. doi:10.1103/PhysRevE.76.046115.
45. Zhou T, Jiang L, Su R, Zhang Y. Effect of initial configuration on network-based recommendation. *EPL (Europhysics Letters)* 2008; **81**(5):58004.
46. Zhou T, Su R, Liu R, Jiang L, Wang B, Zhang Y. Ultra accurate personalized recommendation via eliminating redundant correlations, *Arxiv preprint arXiv:0805.4127*, 2008.
47. Zhang ZK, Liu C, Zhang Y, Zhou T. Solving the cold-start problem in recommender systems with social tags. *A Letters Journal Exploring the Frontiers of Physics* 2010; **92**(2):1–6.
48. Zhou T, Kuscsik Z, Liu J, Medo M, Wakeling J, Zhang Y. Solving the apparent diversity-accuracy dilemma of recommender systems. *Proceedings of the National Academy of Sciences of the United States of America* 2010; **107**(10):4511–4515.
49. Guy I, Zwerdling N, Carmel D, Ronen I, Uziel E, Yogev S, Ofek-koifman S. Personalized recommendation of social software items based on social relations. *Proceedings of the 3rd ACM Conference on Recommender Systems*, New York, New York, USA, 2009; 53–60.
50. Massa P, Avesani P. Trust-aware collaborative filtering for recommender systems. In *Proceedings of Federated International Conference on the Move to Meaningful Internet Systems 2004: CoopIS, DOA, and ODBASE*. Springer: New York, NY, USA, 2004; 492–508.
51. Massa P, Avesani P. Trust metrics on controversial users: balancing between tyranny of the majority and echo chambers. *International Journal on Semantic Web and Information Systems* 2007; **3**(1):39–64.
52. Avesani P, Massa P, Tiella Roberto. A trust-enhanced recommender system application: moleskiing. In *Proceedings of the 2005 ACM Symposium on Applied Computing*. ACM: New York, NY, USA, 2005; 1589–1593.
53. Golbeck J. Computing and applying trust in Web-based social networks. *Ph.D. Thesis*, University of Maryland, College Park, MD, USA, 2005.
54. Golbeck J, Mannes A. Using trust and provenance for content filtering on the semantic web. *Proceedings of the Models of Trust for the Web Workshop*, Edinburgh, UK, 2006.
55. Kuter U, Golbeck J. Sunny: a new algorithm for trust inference in social networks using probabilistic confidence models. *Proceedings of the 22nd National Conference on Artificial Intelligence - Volume 2*, AAAI'07, Vancouver, British Columbia, Canada, 2007; 1377–1382.
56. Jamali M, Ester M. Using a trust network to improve top-*n* recommendation. *Proceedings of the 3rd ACM Conference on Recommender Systems*, New York, New York, USA, 2009; 181–188.
57. Chua F, Lim E. Trust network inference for online rating data using generative models. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM: New York, NY, USA, 2010; 889–898.
58. Ma H, King I, Lyu M. Learning to recommend with social trust ensemble. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM: New York, NY, USA, 2009; 203–210.
59. Ma H, Lyu MR, King I. Learning to recommend with trust and distrust relationships. *Proceedings of the 3rd ACM Conference on Recommender Systems*, New York, New York, USA, 2009; 189–196.
60. Walter FE, Battiston S, Schweitzer F. Personalised and dynamic trust in social networks. *Proceedings of the 3rd ACM Conference on Recommender Systems*, New York, New York, USA, 2009; 197–204.
61. Moghaddam S. FeedbackTrust : using feedback effects in trust-based recommendation systems. *Proceedings of the 3rd ACM Conference on Recommender Systems*, New York, New York, USA, 2009; 269–272.
62. Tavakolifard M. Situation-aware trust management. *Proceedings of the 3rd ACM Conference on Recommender Systems*, New York, New York, USA, 2009; 413–416.
63. Andersen R, Borgs C, Chayes J, Feige U, Flaxman A, Kalai A, Mirrokni V, Tennenholtz M. Trust-based recommendation systems: an axiomatic approach. In *Proceedings of the 17th International Conference on World Wide Web*. ACM: New York, NY, USA, 2008; 199–208.
64. Balabanovic M, Shoham Y. Fab: content-based, collaborative recommendation. *Communications of the ACM* March 1997; **40**(3):66–72.
65. Pazzani M. A framework for collaborative, content-based and demographic filtering. *Artificial Intelligence Review* 1999; **13**(5):393–408.
66. Gunawardana A, Meek C. A unified approach to building hybrid recommender systems. *Proceedings of the 3rd ACM Conference on Recommender Systems*, New York, New York, USA, 2009; 117–124.
67. Jahrer M, Töscher A, Legenstein R. Combining predictions for accurate recommender systems. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM: New York, NY, USA, 2010; 693–702.

DOI: 10.1002/cpe

68. Adomavicius G, Manouselis N, Kwon Y. Multi-criteria recommender systems. In *Recommender Systems Handbook*. Springer US: Boston, MA, 2011; 769–803. doi:10.1007/978-0-387-85820-3.
69. Lakiotaki K, Matsatsinis N. UTA-Rec : a recommender system based on multiple criteria analysis. *Proceedings of the 2nd ACM Conference on Recommender Systems*, Lausanne, Switzerland, 2008; 219–225.
70. Nnadi NJ. Applying relevant set correlation clustering to multi-criteria recommender systems. *Proceedings of the 3rd ACM Conference on Recommender Systems*, New York, New York, USA, 2009; 401–404.
71. Naak A, Hage H, Aimeur E. A multi-criteria collaborative filtering approach for research paper recommendation in papyres. *E-Technologies: Innovation in an Open World*, Ottawa, Canada, 2009; 25–39.
72. Adomavicius G, Kwon Y. New recommendation techniques for multicriteria rating systems. *IEEE Intelligent Systems* May 2007; **22**(3):48–55.
73. Sahoo N, Krishnan R, Duncan G, Callan J. Collaborative filtering with multi-component rating for recommender systems. *Proceedings of the 16th Workshop on Information Technologies and Systems*, Milwaukee, Wisconsin, USA, 2006.
74. Li Q, Wang C, Geng G. Improving personalized services in mobile commerce by a novel multicriteria rating approach. In *Proceeding of the 17th International Conference on World Wide Web*. ACM: New York, NY, USA, 2008; 1235–1236.
75. McNee S, Riedl J. Being accurate is not enough: how accuracy metrics have hurt recommender systems. In *CHI'06 Extended Abstracts on Human Factors in Computing Systems*, CHI EA'06. ACM: New York, NY, USA, 2006; 1097–1101.
76. Herlocker J, Konstan J, Terveen L, Riedl J. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)* January 2004; **22**(1):5–53.
77. Sarwar B, Karypis G, Konstan J, Reidl J. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on World Wide Web*. ACM: New York, NY, USA, 2001; 285–295.
78. Ziegler C, McNee S, Konstan J, Lausen G. Improving recommendation lists through topic diversification. In *Proceedings of the 14th International Conference on World Wide Web*. ACM: New York, NY, USA, 2005; 22–32.
79. Seyerlehner K, Flexer A, Widmer G. On the limitations of browsing top-*n* recommender systems. *Proceedings of the 3rd ACM Conference on Recommender Systems*, New York, New York, USA, 2009; 321–324.
80. Anderson C. *The long tail: why the future of business is selling less of more*. Hyperion: New York, NY, USA, 2006.
81. Kamahara J, Asakawa T, Shimojo S, Miyahara H. A community-based recommendation system to reveal unexpected interests. *Proceedings of the 11th International Multimedia modelling Conference, 2005*, Melbourne, Australia, January 2005; 433–438. doi:10.1109/MMMC.2005.5.
82. Yang Y, Li J. Interest-based recommendation in digital library. *Journal of Computer Science* January 2005; **1**(1):40–46. doi:10.3844/jcssp.2005.40.46.
83. Iaquinta L, de Gemmis M, Lops P, Semeraro G, Filannino M, Molino P. Introducing serendipity in a content-based recommender system. In *Proceedings of 8th International Conference on Hybrid Intelligent Systems*. IEEE: Washington, DC, USA, September 2008; 168–173. doi:10.1109/HIS.2008.25.
84. Onuma K, Tong H, Faloutsos C. Tangent: a novel, 'surprise me', recommendation algorithm. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM: New York, NY, USA, 2009; 657–666.
85. Kawamae N, Sakano H, Yamada T. Personalized recommendation based on the personal innovator degree. *Proceedings of the 3rd ACM Conference on Recommender Systems*, New York, New York, USA, 2009; 329–332.
86. Kawamae N. Serendipitous recommendations via innovators. In *Proceeding of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM: New York, NY, USA, 2010; 218–225.
87. Garcia I, Sebastia L, Onaindia E, Guzman C. A group recommender system for tourist activities. *Proceedings of the 10th International Conference on E-Commerce and Web Technologies*, EC-Web 2009, Linz, Austria, 2009; 26–37. doi:10.1007/978-3-642-03964-5_4.
88. McCarthy K, McGinty L, Smyth B. Case-based group recommendation: compromising for success. *Proceedings of the 7th International Conference on Case-Based Reasoning Research and Development*, Belfast, Northern Ireland, UK, 2007; 299–313.
89. Berkovsky S, Freyne J. Group-based recipe recommendations: analysis of data aggregation strategies. In *Proceedings of the 4th ACM Conference on Recommender Systems*. ACM: New York, NY, USA, 2010; 111–118.
90. Jameson A, Smyth B. Recommendation to groups. In *The Adaptive Web*. Springer-Verlag: Berlin, Heidelberg, 2007; 596–627.
91. Oconnor M, Cosley D, Konstan J, Riedl J. PolyLens: a recommender system for groups of users. In *ECSCW 2001*. Springer: New York, NY, USA, 2001; 199–218.
92. Jameson A. More than the sum of its members: challenges for group recommender systems. In *Proceedings of the Working Conference on Advanced Visual Interfaces*. ACM: New York, NY, USA, 2004; 48–54.
93. Chao D, Balthrop J, Forrest S. Adaptive radio: achieving consensus using negative preferences. In *Proceedings of the 2005 International ACM SIGGROUP Conference on Supporting Group Work*. ACM: New York, NY, USA, 2005; 120–123.
94. Masthoff J. Group modeling: selecting a sequence of television items to suit a group of viewers. *User Modeling and User-Adapted Interaction* February 2004; **14**(1):37–85. doi:10.1023/B:USER.0000010138.79319.fd.
95. Baltrunas L, Makcinskas T, Ricci F. Group recommendations with rank aggregation and collaborative filtering. In *Proceedings of the 4th ACM Conference on Recommender Systems*. ACM: New York, NY, USA, 2010; 119–126.

96. Ardissono L, Goy A, Petrone G, Segnan M, Torasso P. Tailoring the recommendation of tourist information to heterogeneous user groups. *Revised Papers from the International Workshops OHS-7, SC-3, and AH-3 on Hypermedia: Openness, Structural Awareness, and Adaptivity*, Aarhus, Denmark, 2002; 280–295.
97. Crossen A, Budzik J, Hammond K. Flytrap: intelligent group music recommendation. In *Proceedings of the 7th International Conference on Intelligent User Interfaces*. ACM: New York, NY, USA, 2002; 184–185.
98. Masthoff J. Chapter 21 group recommender systems: combining individual models. In *Recommender Systems Handbook*. Springer US: Boston, MA, 2011. doi:10.1007/978-0-387-85820-3.
99. Golder S, Huberman B, Golder S, Huberman B. Usage Patterns of Collaborative Tagging Systems. *Journal of Information Science* 2006; **32**(2):198–208.
100. Milicevic AK, Nanopoulos A, Ivanovic M. Social tagging in recommender systems: a survey of the state-of-the-art and possible extensions. *Artificial Intelligence Review* January 2010; **33**(3):187–209. doi:10.1007/s10462-009-9153-2.
101. Krestel R, Fankhauser P, Nejdl W. Latent Dirichlet allocation for tag recommendation. *Proceedings of the 3rd ACM Conference on Recommender Systems*, New York, New York, USA, 2009; 61–68.
102. Jaschke R, Marinho L, Hotho A, Schmidt-Thieme L, Stumme G. Tag recommendations in folksonomies. *Proceedings of the 11th European Conference on Principles and Practice of Knowledge Discovery in Databases*, Warsaw, Poland, 2007; 506–514.
103. Lipczak M. Tag recommendation for folksonomies oriented towards individual users. *Proceedings of the ECML/PKDD 2008 Discovery Challenge Workshop*, Antwerp, Belgium, 2008; 84–95.
104. Lipczak M, Hu Y, Kollet Y, Milios E. Tag sources for recommendation in collaborative tagging systems. *Proceedings the ECML/PKDD 2009 Discovery Challenge Workshop*, Bled, Slovenia, 2009; 157–172.
105. Hart M, Brook S, York N, Johnson R, Stent A. iTag : a personalized blog tagger. *Proceedings of the 3rd ACM Conference on Recommender Systems*, New York, New York, USA, 2009; 297–300.
106. Guan Z, Bu J, Mei Q, Chen C, Wang C. Personalized tag recommendation using graph-based ranking on multi-type interrelated objects. *Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Boston, MA, USA, 2009; 540–547.
107. Symeonidis P, Nanopoulos A, Manolopoulos Y. Tag recommendations based on tensor dimensionality reduction. *Proceedings of the 2nd ACM Conference on Recommender Systems*, Lausanne, Switzerland, 2008; 43–50.
108. Sigurbjörnsson B, van Zwol R. Flickr tag recommendation based on collective knowledge. In *Proceedings of the 17th International Conference on World Wide Web*. ACM: New York, NY, USA, 2008; 327–336.
109. Heymann P, Ramage D, Garcia-Molina H. Social tag prediction. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM: New York, USA, 2008; 531–538. doi:10.1145/1390334.1390425.
110. Song Y, Zhuang Z, Li H, Zhao Q, Li J, Lee W, Giles C. Real-time automatic tag recommendation. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM: New York, USA, 2008; 515–522. doi:10.1145/1390334.1390423.
111. Gemmell J, Ramezani M, Schimoler T, Christiansen L, Mobasher B. The impact of ambiguity and redundancy on tag recommendation in folksonomies. *Proceedings of the 3rd ACM Conference on Recommender Systems*, New York, New York, USA, 2009; 45–52.
112. Parra D, Brusilovsky P. Collaborative filtering for social tagging systems : an experiment with CiteULike. *Proceedings of the 3rd ACM Conference on Recommender Systems*, New York, New York, USA, 2009; 237–240.
113. Tintarev N, Masthoff J. Designing and evaluating explanations for recommender systems. In *Recommender Systems Handbook*. Springer US: Boston, MA, 2011; 479–510. doi:10.1007/978-0-387-85820-3.
114. Sinha R, Swearingen K. The role of transparency in recommender systems. In *CHI'02 Eextended Abstracts on Human Factors in Computing Systems*, CHI EA'02. ACM: New York, NY, USA, 2002; 830–831. doi:10.1145/506443.506619.
115. Bilgic M, Mooney R. Explaining recommendations: satisfaction vs. promotion. *Proceedings of Beyond Personalization 2005, the Workshop on the Next Stage of Recommender Systems Research*, San Diego, California, USA, 2005; 13–18.
116. Tintarev N, Masthoff J. A survey of explanations in recommender systems. *Proceedings of the 2007 IEEE 23rd International Conference on Data Engineering Workshop*, ICDEW '07, Istanbul, Turkey, 2007; 801–810. doi:10.1109/ICDEW.2007.4401070.
117. Symeonidis P, Nanopoulos A, Manolopoulos Y. MoviExplain : a recommender system with explanations. *Proceedings of the 3rd ACM Conference on Recommender Systems*, New York, New York, USA, 2009; 317–320.
118. Mehta B, Nejdl W. Unsupervised strategies for shilling detection and robust collaborative filtering. *User Modeling and User-Adapted Interaction* July 2008; **19**(1-2):65–97. doi:10.1007/s11257-008-9050-4.
119. Hurley N, Cheng Z, Zhang M. Statistical attack detection. *Proceedings of the 3rd ACM Conference on Recommender Systems*, New York, New York, USA, 2009; 149–156.
120. Cheng Z, Hurley N. Effective diverse and obfuscated attacks on model-based recommender systems. *Proceedings of the 3rd ACM Conference on Recommender Systems*, New York, New York, USA, 2009; 141–148.
121. Wang H, Lu Y, Zhai C. Latent aspect rating analysis on review text data: a rating regression approach. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM: New York, NY, USA, 2010; 783–792.
122. Pilászy I. Recommending new movies : even a few ratings are more valuable than metadata. *Proceedings of the 3rd ACM Conference on Recommender Systems*, New York, New York, USA, 2009; 93–100.

DOI: 10.1002/cpe

123. Goldberg K, Roeder T, Gupta D, Perkins C. Eigentaste: a constant time collaborative filtering algorithm. *Information Retrieval* 2001; **4**(2, July):133–151.
124. Shani G, Gunawardana A. Evaluating recommendation systems. *Recommender Systems Handbook* 2011:257–297.
125. Steck H. Training and testing of recommender systems on data missing not at random. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM: New York, NY, USA, 2010; 713–722.
126. Han J, Kamber M. *Data Mining: Concepts and Techniques* (1st edn). Morgan Kaufmann Publishers Inc.: San Francisco, CA, USA, 2000.
127. Ge M, Delgado-Battenfeld C, Jannach D. Beyond accuracy: evaluating recommender systems by coverage and serendipity. In *Proceedings of the 4th ACM Conference on Recommender Systems*. ACM: New York, NY, USA, 2010; 257–260.
128. Wang J, de Vries AP, Reinders MJT. Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM Press: New York, USA, 2006; 501–208. doi:10.1145/1148170.1148257.
129. Ma H, King I, Lyu MR. Effective missing data prediction for collaborative filtering. In *SIGIR 2007*. ACM Press: New York, USA, 2007; 39–46.
130. Ren Y, Li G, Zhang J, Zhou W. The efficient imputation method for neighborhood-based collaborative filtering. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*. ACM Press: New York, USA, 2012; 684–693.
131. Ren Y, Li G, Zhang J, Zhou W. Lazy collaborative filtering for data sets with missing values. *IEEE Transactions on Cybernetics* 2013; **43**(6):1822–1834.
132. Claypool M, Gokhale A, Miranda T, Murnikov P, Netes D, Sartin M. Combining content-based and collaborative filters in an online newspaper. *Proceedings of ACM SIGIR Workshop on Recommender Systems: Algorithms and Evaluation*, Berkeley CA, USA, 1999.
133. Mackey L, Weiss D, Jordan M. Mixed membership matrix factorization. *Proceedings of the Internaltinal Conference on Machine Learning*, Haifa, Israel, 2010; 711–718.

DOI: 10.1002/cpe