Information Sciences 414 (2017) 1–18

Contents lists available at ScienceDirect

Information Sciences

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

Context Neighbor Recommender: Integrating contexts via neighbors for recommendations

Lin Zheng[a,b], Fuxi Zhu[a,c,*], Sheng Huang[a], Jin Xie[a]

[a] Computer School, Wuhan University, Wuhan 430072, China
[b] Department of Computer Science, Hong Kong Baptist University, Hong Kong, China
[c] Information Department, Wuhan College, Wuhan 430212, China

ARTICLE INFO

Article history:
Received 25 June 2016
Revised 17 May 2017
Accepted 21 May 2017
Available online 22 May 2017

Keywords:
Recommender system
Item recommendation
Implicit feedback
Context neighbor
Neighbor aggregation
Group interaction

ABSTRACT

In Recommender Systems, the techniques used for modeling implicit feedback such as search, click or purchase actions have been well studied. As additional information, contexts are often available to assist the implicit feedback approaches. The existing context-aware methods are used to directly model original contextual factors for recommendations. However, this way of utilizing contextual information makes the methods dependent on specific contexts. Moreover, they may not achieve the desired performance if the contexts are changed to those derived from other domains. To address this issue, we propose a general approach to incorporate contexts into an implicit feedback modeling framework that can utilize specific contexts but is domain independent. First, we introduce context neighbors to integrate original contextual factors. The neighbors are aggregated to form several groups. Then, our recommender builds on group interactions that expand the traditional user-item interactions. Finally, the recommendations are obtained by combining the results of all the interaction models. We evaluate the Context Neighbor Recommender (CNR) for different choices of neighbor numbers and kernel settings to further compare it with other algorithms. The experimental results show the advantages and flexibility of CNR compared to both implicit feedback methods and common context-aware models.

1. Introduction

Recommender systems are information filtering tools that alleviate information overload for web users. As one of the most successful techniques in recommender systems, Collaborative Filtering (CF) has made significant progress in exploiting user-item relations [33]. A traditional problem of CF is estimating the explicit preferences of users toward items, which are usually represented as ratings. From this perspective, rating prediction is a basic task in typical CF algorithms [16,17] for creating recommendations. Rating prediction is considered a regression problem, and rating-based CF approaches aim to generate a recommendation list by sorting all of the predicted scores in descending order. In some practical scenarios, however, explicit ratings are unavailable because there are no incentives for users to rate items. For instance, users may only interact with items by browsing them on the website or adding them to their shopping carts. These types of actions (or events) are termed implicit feedback [11,30,31]. Since it is more accessible than explicit preferences, implicit feedback is often expressed in a binary data format to reflect users' actions or inactions. Hence, another category of CF consists of

* Corresponding author at: Computer School, Wuhan University, Wuhan 430072, China.
E-mail addresses: linzheng@whu.edu.cn (L. Zheng), fxzhu@whu.edu.cn (F. Zhu), 2014102110042@whu.edu.cn (S. Huang), jinxie@whu.edu.cn (J. Xie).

http://dx.doi.org/10.1016/j.ins.2017.05.034
0020-0255/

approaches that focus on modeling the implicit feedback to implement recommendations. This can be considered a classification or ranking problem because given a list of recommended items, users often express their attitudes as "like" or "dislike", and because users are occasionally sensitive to the order in which the items are presented.

The CF techniques concerning implicit feedback have various names, such as "One-Class Collaborative Filtering (OCCF)" [11,26,27] and "collaborative ranking" [2,12,18,43]. To unify these appellations, in this paper, we simply call these methods Implicit Feedback Modeling in Collaborative Filtering (ICF). Personalized item recommendations in ICF are substantially more complicated than are those based on rating prediction, since implicit feedback could be incomplete or less accurate than explicit ratings. Additionally, the impact of explicit scores should be considered. If only implicit actions are available, they are usually interpreted as positive feedback [11,24,31,40], such as search, click or purchase actions. Accordingly, negative feedback is defined as those "missing" events that can be inferred from the positive ones to reflect the latent preferences of users. When explicit ratings are accessible, they help improve performance. One way of utilizing score data is to integrate them into the learning process [8,37,40]. Another is to treat them as supplementary information, such as weights [11,27], item probabilities [12] or user preferences [19], to facilitate recommendations. In addition to the standard approaches, there are some ICF approaches that obtain the recommendation lists by directly optimizing the measures of information retrieval [31,36]. Furthermore, it is possible to blend various algorithms via some learning frameworks, e.g., neural network [12], boosting [6] and low-rank approximation [18], to obtain more accurate results.

Limitations of existing methods. Having a primary focus on implicit feedback makes ICF algorithms lightweight and flexible. For example, they can create recommendations without utilizing additional information such as user profiles, item attributes or expert knowledge. This feature is known as "context irrelevance". However, context irrelevance is also a shortcoming of ICF because it cannot take advantage of contextual information (also known as contextual factors [33]). Although Context-Aware Recommender Systems (CARS) [1] have attempted to compensate for this shortcoming by offering rich side information in addition to feedback modeling, researchers have shown that incorporating contexts into rating prediction methods is more natural than doing the same in ICF-based approaches [1,3,29,46]. More specifically, general context regression models, such as Factorization Machine [29], prefer to introduce a large number of parameters to simulate various contextual factors, e.g., *movie genre, movie director*, or *music album*. Then, the predicted ratings are naturally derived from the combinations of these contextual parameters. In ICF, the assumption of general algorithms [9,10,25,30,35] that all contextual factors have equal impacts may be inappropriate. This is because the outcomes in ICF are more sensitive to specific contexts compared to those in rating prediction according to different domains [39]. For instance, time factors have a huge influence on microblog [47] recommendation, whereas location [44] recommendation is more sensitive to geographical information. Hence, the generic context-aware recommenders may not achieve the desired results because they are unable to distinguish which contextual factors are helpful for performance enhancement. The difficulty lies in the fact that common algorithms are all established based on original contextual factors of different forms and meanings that are not standardized. From this perspective, general context-aware recommenders are still domain dependent; specifically, their performances are highly sensitive to different domains. If we can find a proper way to make contexts domain independent, then the negative impacts of specific contextual factors can be greatly reduced. In this case, traditional context modeling will not work since the original contexts have been reorganized into another uniform format. However, the ICF techniques can be employed to utilize such new contexts, not only because the ICF algorithms are powerful in mining user-item relations, but also because they are irrelevant to specific contexts. This motivates us to unify the original contexts to seamlessly integrate them into a new framework that employs the advantages of ICF models.

Our approach. In this paper, we propose a Context Neighbor Recommender (CNR) for item recommendation. By utilizing context neighbors to transmit contextual information, our recommender implements a domain independent representation for original contexts to mine their intrinsic meanings. Then, by utilizing and improving ICF techniques, we develop a context neighbor aggregation approach to carry out group interactions, which differ from the user-item interactions in traditional ICF algorithms. Each group interaction generates a sub-recommender to provide local predicted outcomes that are combined to provide the final recommendations.

To summarize, the contribution of our work is fourfold:

1. *Context Integration.* We integrate contexts into neighbors to solve the domain dependent deficiencies that are present in common context-aware methods. The original contexts are divided into user profiles and item attributes. Then, the context neighbors of users are found by using a similarity calculation of their contexts, as are item context neighbors. The goal of context integration can be achieved by replacing the contexts of current entities (users or items) with the information of their neighbors.
2. *Neighbor Aggregation.* To make full use of neighbor information, we employ kernel smoothing to convert context similarities into neighbor weights. Each neighbor, along with its kernel weight, is gathered with the current entity to form a group. A group containing only two entities can largely reduce the mutual influences between neighbors. This is the main point of neighbor aggregation.
3. *Group Interaction.* To refine the user-item interactions in classic ICF algorithms, we arrange groups to take the place of users, items, or both for interactions because such groups can offer more valuable information than individuals can. The modeling of each interaction produces a small recommender to give local predicted results. Kernel weights are utilized when training sub-recommenders and when combining the local recommendations from sub-recommenders.

4. *Flexible Recommendation.* Three categories of group interactions give rise to three recommenders in the CNR family: CNR-U (CNR with User aspects), CNR-I (CNR with Item aspects), and CNR-UI (CNR with User-Item aspects). CNR-U has the ability to implement recommendations without item contexts, CNR-I runs normally when user contexts are missing, whereas CNR-UI is capable of utilizing both user and item contexts. The flexibility is also reflected in the selection of recommenders according to their performances on different datasets.

The remainder of the paper is organized as follows. In Section 2, we highlight the relevant works of ICF. The model framework and detail constructions of our recommender are introduced in Section 3, whereas Section 4 presents the results and analysis of the experiments. The conclusions follow in Section 5 and finally, we give some perspective of further research.

2. Related works

Item recommendation algorithms in ICF used to figure out the relations between users and items by matrix factorization techniques as Eq. (1) shows:

$$\hat{y}_{ij} = p_i^T q_j \tag{1}$$

where p_i is the factor of user i and q_j denotes the factor of item j. Then, the predicted value $\hat{y}_{ij}$ is obtained by their inner product. In most relevant works of ICF, researchers usually carried out their recommenders on the foundation of such factorization framework. In these studies, implicit feedback is regarded as the most important aspect, whereas explicit feedback exists as the supplementary element for recommendation. Hence, the first two categories of ICF algorithms can be classified into *pure implicit-feedback approaches* and *implicit models utilizing explicit-feedback*. By considering context information, the approaches of the third type are the *context-aware methods* that pay more attention to contextual factors in addition to implicit feedback.

2.1. Pure implicit-feedback approaches

As a common way of implicit feedback modeling, existing user actions (or events) are treated as the positive feedback while the missing ones denote the negative feedback. E.g., Bayesian Personalized Ranking (BPR) [31] offered a generic optimization of *AUC* (Area Under the Curve) to distinguish the positive events from negative actions for users. Compared to the criterion optimization in BPR, [36] introduced a new approach to directly maximize another information retrieval metric—the Mean Reciprocal Rank (MRR). In contrast, Sparse Linear Methods (SLIM) [24] only concentrates on the positive feedback, which demonstrated a preferred item is reasonable to be expressed as an aggregation of its neighbors. The authors of SLIM continued to present another factor model [15] in the purpose of capturing item similarities to generate high quality recommendations. Besides the above methods, ensemble learning [48] highlights the performances of single algorithms by blending them together. Neural network [12] is one of the applicable ensemble tools having an intuitive structure to combine various techniques. Boosting [5,6], another widely used wrapper, has illustrated its efficiency in feature learning for *top-N* recommendation. Additionally, Local Collaborative Ranking (LCR) [18] employed local low-rank matrix approximation [13] techniques to show the ability of implicit feedback modeling. According to the characteristics of the CNR family, BPR, SLIM, and LCR are three appropriate algorithms to be compared with our recommender.

2.2. Implicit models utilizing explicit-feedback

Explicit feedback, typically user ratings, often plays a supporting role in implicit methodologies. The first auxiliary manner is sorting items according to the scores in descending order, as CofiRank [43] did, to minimize the loss using maximum-margin matrix factorization. EigenRank [23] made use of sorted ratings by similarity calculation and ranked items based on the preferences between similar users. The second way of rating integration is to couple them into the learning procedures tightly. E.g., Takács and Tikk [40] integrated scores into the objective function to be optimized directly based on BPR framework. Gantner et al. [8] combined the item recommendation with rating prediction through incorporating score data precisely, whereas Shi et al. [37] unified rating-oriented and ranking-oriented approaches to address the limitations of implicit feedback modeling. The third approach, more common than the first two, is converting the scores into another applicable forms, such as item probabilities [12] or user preferences [19]. One of the representative methods, the One-Class Collaborative Filtering (OCCF) [11,26,27], developed a general approach regarding ratings as weights to settle item recommendation. As an example, Weighted Regularized Matrix Factorization (WRMF) [11] formalized scores to be the confidence weights and suggested an Alternating-Least-Squares (ALS) optimization process to minimize their cost function. Compared to OCCF, the weights assigned to our loss function are the neighborhood relations rather than pure ratings, moreover, we change neighbor numbers and kernel settings instead of adjusting the weights to achieve the best performances.

2.3. Context-aware methods

Methodologies of this category can be divided into domain oriented applications [14,20,22,42,44,47] and generic ones [9,10,25,30,35,45]. Since our works tend to find a common way of item recommendation by integrating contexts into ICF,

we primarily pay attention to the general researches. The first kind of context-aware methods is derived from *pure implicit-feedback approaches*. E.g., Based on BPR, Rendle and Freudenthaler [30] designed a non-uniform item sampler to solve the learning problems of convergence slowing down in context-aware recommendation. SSLIM (Sparse Linear Methods with Side Information) [25] extended SLIM [24] by properly exploiting and incorporating side information, to achieve measurable improvements over original approaches. The second category of techniques is modeling contextual factors directly [45]. As a representative method, Tensor Factorization (TF) has the ability to model contexts intuitively. E.g., Hidasi and Tikk [9] proposed a ALS-based tensor factorization recommender to incorporate various contextual factors while maintaining its computational efficiency. Shi et al. [35] directly maximized Mean Average Precision (MAP) using tensor structure to simulate implicit feedback data. Although TF has been proved to provide high quality recommendations, it often requires a cubic runtime for both learning and prediction. To solve this problem, Rendle and Schmidt-Thieme [32] developed the PITF (Pairwise Interaction Tensor Factorization) model to outperform TF in runtime and prediction quality, especially for tag recommendation. PITF is not limited to tags and can be further applied to other contexts [10], which has a formalized form as follows:

$$\hat{y}_{ij} = p_i^T q_j + p_i^T x_c^{(i)} + q_j^T x_c^{(j)} \tag{2}$$

where $x_c^{(i)}$ is the generalized contexts of user i and $x_c^{(j)}$ denotes the contextual factors of item j. As further study of PITF, Hidasi and Tikk [10] compared different versions of pairwise interaction models to establish the General Factorization Framework (GFF) and demonstrated the efficiency and flexibility of GFF. In our experiments, we selected SSLIM and PITF as two representative context-aware approaches for comparisons, in order to demonstrate the advantages of utilizing contexts in CNR.

3. Context Neighbor Recommender

In Section 3.1, we first explain the basic idea and framework of CNR. Then, the processes of context integration via neighbor aggregation are introduced in Section 3.2. The interactions of neighbor groups follow in Section 3.3 to create three types of sub-models. At last, we gather the local results together to gain the final recommendation and analyze the complexity of our algorithm in Section 3.4.

3.1. Model framework

In this section, we illustrate the framework of CNR in Fig. 1 , to explain the main idea of our recommender in three steps.

First, contextual factors are divided into user-oriented categories, such as *age, gender* and *occupation*, and into item attributes, such as *music genre* or *movie director*. It should be noted that the ownership of some contexts is ambiguous because they are double oriented; e.g., tags can be either given by users or marked on items. In this case, we consider them to be the common attributes that belong to both users and items. After finishing this pre-filtering [3] work, we search for the context neighbors of users and items. For example, consider a user whose context neighbors are those who have profiles or exhibit behaviors similar to hers, e.g., *having a similar educational background, joining the same community*, or *purchasing an identical item*. Being analogous to users, item context neighbors contain similar contextual information. Thus, both user neighbors and item neighbors are chosen based on the computation of context similarities. In fact, only the neighbors with N highest similarity scores will be selected. The formal definition of context neighbor is given in Section 3.2. Further, the context similarities such as $\{c_{i1}, c_{i2}, c_{i3}\}$ and $\{c_{j1}, c_{j2}, c_{j3}\}$ are retained to be exploited in the next step.

In the second part, after context neighbor selection, we can extract both the user and item information by using their neighbors instead of the original contexts, because it was confirmed that neighbors can provide helpful information about each other [4,7] in recommendations. In other words, context neighbors have taken the place of original contextual factors in information representation to make our recommender domain independent, which is the key idea of context integration. The term “domain independent” does not mean that our approach is insensitive to the original contexts; instead, it indicates that the context neighbors have unified different domain contexts into the same format to get rid of their original representations. Therefore, the context neighbors will not be dependent on or limited to specific domains. The extracted information can be seen as the valuable suggestions offered by user neighbors or the additional descriptions collected from item neighbors. The context neighbors of user i, such as $\{u_{i1}, u_{i2}, u_{i3}\}$, should be aggregated so they can contribute to recommendations. In preparation for aggregation, kernel smoothing [41] is employed to convert the user similarities $\{c_{i1}, c_{i2}, c_{i3}\}$ into the kernel weights $\{k_{i1}, k_{i2}, k_{i3}\}$ because kernel estimators provide a simple and reliable way of fitting data. Additionally, the kernel weights are utilized not only in neighbor aggregation but also in training processes and recommendation combination. The *top-N* context neighbors are then sequentially aggregated with user i, which means that only one neighbor is gathered each time. Hence, this neighbor aggregation approach results in N groups; e.g., there are three groups $\{u_i, u_{i1}\}$, $\{u_i, u_{i2}\}$, $\{u_i, u_{i3}\}$ derived from three neighbors of user i. In contrast to traditional neighborhood models, our methods can largely reduce the mutual influences between neighbors by using the approach of single neighbor aggregation. In other words, the personal advice from each context neighbor can be received by user i without being disturbed.

In the last step, we try to establish our recommender based on groups derived from neighbor aggregation. In contrast to traditional ICF algorithms shown in Eq. (1), we extend the user-item interplay to group interactions by utilizing groups to

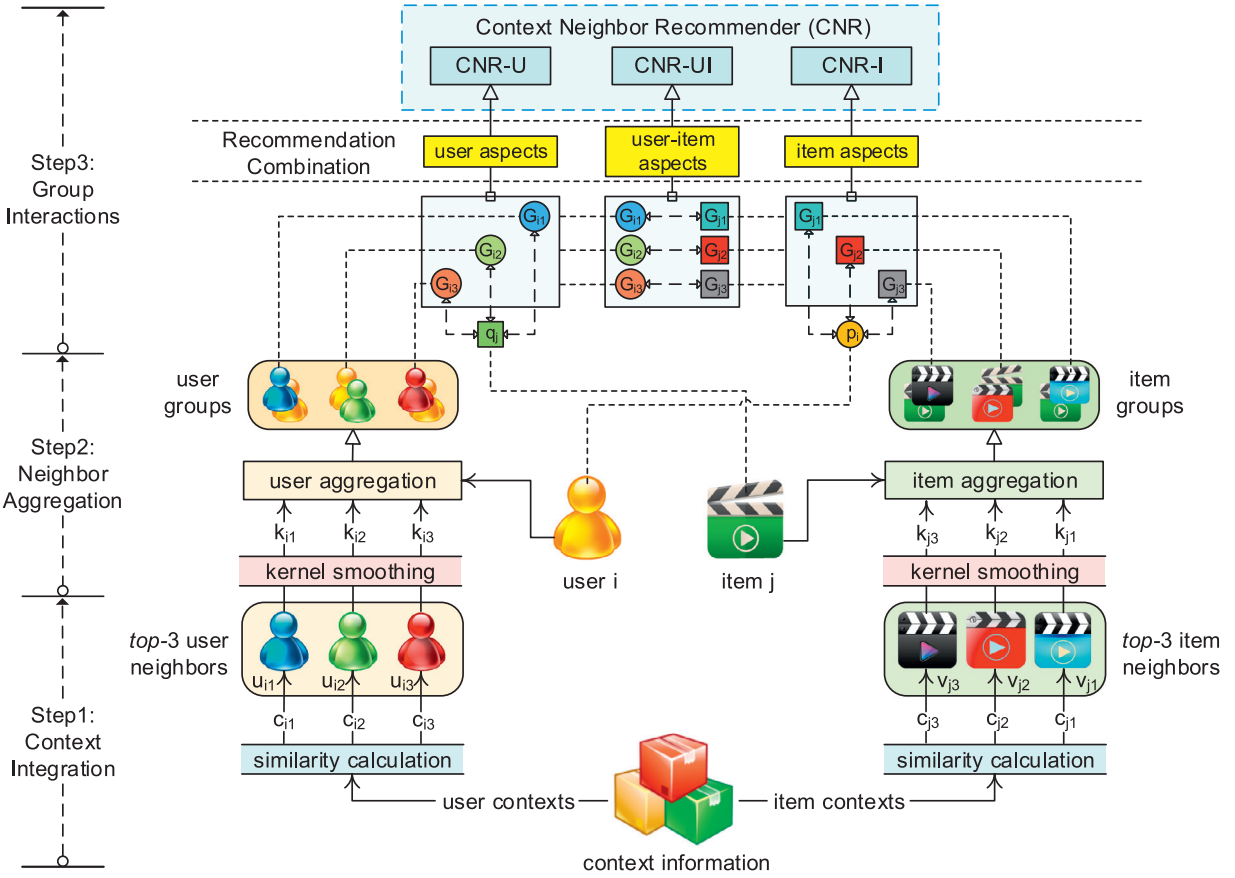

Fig. 1. The framework of Context Neighbor Recommender.

replace users, items, or both. The basic elements of groups are still users and items, and the entities (users or items) in a group have been supplemented with rich contextual information in the first two steps. Hence, when interactions are carried out, the information offered by groups exceeds that offered by individuals, thereby producing recommendation results that are more reliable and reasonable. The first type of group interaction in CNR involves user group interaction, in which the item factor q_j interacts with the user groups G_{i1}, G_{i2}, and G_{i3}. Each interaction can provide one predicted preference value from user i toward item j, which means that we have three sub-recommenders due to the three different group interactions. Then, the local recommendation results from sub-models are combined to form CNR-U. Likewise, from the item angles, the item groups G_{j1}, G_{j2}, and G_{j3} interact with the user factor p_i to produce three local recommendations. They are gathered together to form CNR-I. The last recommender, CNR-UI, is the most complicated one because it captures pure group interactions from both the user and item aspects. More details about group interactions are provided in Section 3.3 , whereas the way of recommendation combination is introduced in Section 3.4. Thus far, the CNR family is composed of these three categories of recommenders that make our approach domain independent and flexible in different datasets. The advantage of domain independence is that the dependence of specific contexts is largely reduced; consequently, the intrinsic meanings of the original contexts can be determined more easily, as validated throughout the experiments in Section 4.

3.2. Context integration via neighbor aggregation

The first step in the procedure for context integration is to look for context neighbors. In preparation, context similarities need to be calculated. Similarity or distance measure is the primary criterion for neighbor selection in most CF literature [4,7,33]. Representative metrics, such as Euclidean distance, Cosine similarity, and Pearson correlation, have been widely adopted in many pre-filtering works. In context-aware recommendations, Spertus et al. [38] carried out large-scale studies and argued that the best outcomes were algorithms created based on cosine similarity. Hence, we employ cosine similarity as our pre-filtering tool for neighbor finding. More emphasis is placed on neighbor modeling, e.g., finding the most suitable number of neighbors and aggregating them in a proper way. The optimal number of neighbors can be determined from experiments in Section 4.1, whereas neighbor aggregation requires the *top-N* context neighbors. Specifically, given a collection of one kind of entity (user or item but not both) $E := \{e_1, e_2, \ldots, e_i, \ldots\}$ and its context set S_e, let the function $C(e_i, e_j,$

S_e) represents the context similarity between e_i and e_j. The general definition of *top-N* context neighbor is formalized as follows:

Definition 1. (*Top-N* Context Neighbor). For a specific entity (user or item) e_i, suppose $e_{ij} \in E \backslash \{e_i\}$ is any other entity in the collection E except e_i, compute $C(e_i, e_{ij}, S_e)$ and sort the scores in descending order to obtain an ordered entity set $E_i := \{e_{i1}, e_{i2}, \ldots, e_{ij}, \ldots\}$, then the *top-N* context neighbors of e_i are defined as the first N entities in E_i.

For instance, u_{i1}, u_{i2}, and u_{i3} are the *top*-3 user context neighbors of u_i because $C(u_i, u_{i1}, S_u)$, $C(u_i, u_{i2}, S_u)$, and $C(u_i, u_{i3}, S_u)$ occupy the top three highest scores, where S_u denotes the user context collection and the similarities are abbreviated as c_{i1}, c_{i2}, and c_{i3}.

The purpose of searching the *top-N* context neighbors for u_i is to aggregate the N most similar users with respect to u_i. A common solution is to weight the neighbor factors by their similarities, and the weights are often normalized by their summation. Consider the classic user-item interaction in Eq. (1), if we extend it using such common way to aggregate user neighbors, then a user-enhanced recommender has the following form:

$$\hat{y}_{ij} = \left(p_i + \sum_{n=1}^{N} \frac{c_{in} \cdot u_{in}}{\sum_{m=1}^{N} c_{im}} \right)^T q_j \tag{3}$$

where N denotes the number of context neighbors, c_{in} is the context similarity between user i and her nth neighbor u_{in}. This explains the idea of merging factor models and neighborhood methods to combine the advantages of each one. This category of models has been demonstrated to be powerful for rating prediction [16]. However, in item recommendation, simply aggregating the neighbors using Eq. (3) may not achieve the desired results. Moreover, the context similarities have their connotations to be exploited because the neighbors are found according to them. Thus, we have to provide a simple and reliable approach for finding the structures of the context similarities first. Following common techniques in non-parametric statistics, we employ kernel smoothing [41] as a tool to smooth out context similarities. This approach is simple since it can be implemented without the imposition of a parametric model. It is also a reliable approach for establishing a stable structure from existing data. By kernel smoothing, the context similarities are converted into the kernel weights as follows:

$$k_{in} := K\Big(\frac{1}{h}(1 - c_{in})\Big) \tag{4}$$

here the kernel function K requires a distance between two entities as its input, so the context similarity c_{in} is converted into the distance by $(1 - c_{in})$. Cosine similarity has its range in [0, 1] which is convenient to be changed into a distance still ranging [0, 1], this is another reason why we choose cosine as our similarity metric. The scaling factor h plays a role of kernel width (or bandwidth) that determines the spread of a kernel. Three popular kernel functions we used are the Epanechnikov kernel (K_E), the Biweight kernel (K_B), and the Gaussian kernel (K_G) defined as:

$$K_E(u) := \frac{3}{4}(1 - u^2)\ \mathbb{I}(|u| \le 1) \tag{5}$$

$$K_B(u) := \frac{15}{16}(1 - u^2)^2\ \mathbb{I}(|u| \le 1) \tag{6}$$

$$K_G(u) := \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{u^2}{2}\right) \tag{7}$$

where $\mathbb{I}$ is an indicator function so that $\mathbb{I}(x) = 1$ if the condition x is satisfied, else $\mathbb{I}(x) = 0$. In our experiments, the kernel types and widths can be adjusted according to different datasets for best results.

Importing kernel weights provides good preparation for neighbor aggregation. Nonetheless, if we simply adopt Eq. (3) as the aggregation approach, the weighted contributions of neighbors may cancel each other out after being summed. More specifically, the impact of a positive neighbor factor such as u_{i1} might be offset by a negative factor such as u_{i3}; thus, the suggestion of each neighbor would not be clearly received by u_i. This is the reason that item recommendation would not achieve a satisfactory result if we treat it as a regression problem. However, if we gather only one neighbor at a time, then the mutual influences of neighbors can be greatly reduced. To achieve this goal, N context neighbors are divided into N groups. Each group contains two people: u_i and one of her neighbors. This method of aggregation creates a *one-on-one* conversation that allows users to directly hear the views of a single neighbor. It is the essential idea and highlight of neighbor aggregation. Although we illustrate this from user aspect as an example, the theoretical assumption is also tenable in the item perspective. The common definition of neighbor aggregation for group n is formalized as follows:

Definition 2. (Neighbor Aggregation of Group n). For an active entity e_i, given one of its *top-N* neighbors $t_{in} (n \le N)$, suppose k_{in} is the kernel weight between these two entities, and e_i is represented as e_{in} in group n. Then, the neighbor aggregation of group n is defined as $G_{in} := e_{in} + k_{in} \cdot t_{in}$.

where the nth group is denoted by G_{in}. If *top-N* context neighbors are chosen, then N small groups will be generated after single-neighbor aggregation. Each group can be seen as an entity that has been enhanced for participation in the user-item interplay. The details of group interactions are introduced in Section 3.3. Each interaction is essentially a sub-model

that provides local recommendation results. The predicted values of these sub-models are then gathered to create final recommendations in Section 3.4.

3.3. Group interactions in sub-models

Using groups that were created from neighbor aggregation, we expand the user-item interactions to group interactions, which means that each interaction must contain at least one group as the interacting element. Define the symbol of interaction as $\Leftrightarrow$. Three categories of group interactions are as follows: *user-group*$\Leftrightarrow$*item, item-group*$\Leftrightarrow$*user*, and *user-group*$\Leftrightarrow$*item-group*. Hence, CNR includes three recommenders derived from three types of group interactions. Each recommender consists of N sub-recommenders because there are N interactions for N groups. The sub-models in each recommender are the same so we choose one of them as representative to discuss their constructions. For preliminaries, the basic notations of our sub-recommenders are listed as follows:

- p_{in} - the factor of user i in the nth group.
- q_{jn} - the factor of the jth item in group n.
- u_{in} - the nth context neighbor belongs to user i.
- v_{jn} - the nth context neighbor of item j.
- k_{in} - the kernel weight of u_{in}.
- k_{jn} - the kernel weight to assist v_{jn}.
- $\hat{y}_{ij}^{(n)}$ - the predicted value of the nth sub-model.

In user aspects, the nth sub-model of CNR-U has the following form:

$$\hat{y}_{ij}^{(n)} = (p_{in} + k_{in} \cdot u_{in})^T q_{jn} \tag{8}$$

where the nth small user group $(p_{in} + k_{in} \cdot u_{in})$ interacts with item factor q_{jn}, to give a user-enhanced local decision $\hat{y}_{ij}^{(n)}$. Notice that the subscript n has double meanings: the nth group and the nth neighbor, since the nth neighbor u_{in} is assigned to group n according to Definition 2 . Nevertheless, the subscript n is not necessarily required when we just mention one of the sub-models. Hence, in this case, the factors p_{in}, u_{in}, and q_{jn} can be abbreviated as p_i, u_i and q_j, respectively. Similar to CNR-U, the nth sub-model of the item-aspect recommender CNR-I is defined as:

$$\hat{y}_{ij}^{(n)} = p_{in}^T (q_{jn} + k_{jn} \cdot v_{jn}) \tag{9}$$

where this kind of sub-model focuses on the interaction between item group $(q_{jn} + k_{jn} \cdot v_{jn})$ and user p_{in}. In addition to user and item aspects, the pure group interplay is defined as follows:

$$\hat{y}_{ij}^{(n)} = (p_{in} + k_{in} \cdot u_{in})^T (q_{jn} + k_{jn} \cdot v_{jn}) \tag{10}$$

This symmetric recommender is named CNR-UI for both the user group and the item group participate in the interaction.

CNR-UI considers both k_{in} and k_{jn}, whereas CNR-U and CNR-I utilize only one of the kernel weights. To make full use of the kernel weights and unify three versions of CNR, we improve the weighting approach in OCCF [11,26,27] to train our models. Specifically, the kernel weights are employed to replace the adjustable weights in OCCF. Then, the ratings on context neighbors are combined by the kernel weights to be the global weights. At last, our loss functions are weighted by the global weights to be minimized for final results. For example, in CNR-U, the explicit preference of user i toward item j is used to be the rating r_{ij}. However, we can expand this preference and express it as the average rating by user i of the context neighbors of item j. Defining $r_{i(jn)}$ as the rating by user i of the nth neighbor of item j, we employ k_{jn} (the kernel weight of the nth item neighbor) to obtain the average rating as Eq. (11) shows:

$$\bar{r}_{i(j)} := \sum_{n=1}^{N} \frac{k_{jn} \cdot r_{i(jn)}}{\sum_{m=1}^{N} k_{jm}} \tag{11}$$

where $\bar{r}_{i(j)}$ is the average weighted rating generated by the aggregation of $r_{i(jn)}$. The kernel weight k_{jn} is naturally incorporated to restrict the relevant rating. However, user i might have no ratings on all the neighbors of item j, which is actually related to the *cold-start problems* [21,28] in recommender systems. Most researchers tended to solve such problems by leveraging user profiles or item attributes to compensate for the absence of rating actions [14,21,28]. In fact, all the available information, including user profiles and item attributes, has been considered to be the original contexts in our work. Thus, the kernel weights k_{jn} and k_{jm} are not sensitive to the cold-start problems because the calculations of context similarities have involved all the user and item information. Nevertheless, the average weighted rating $\bar{r}_{i(j)}$ will be equal to zero when all the $r_{i(jn)}$ are also equal to zero, therefore, we cannot directly adopt $\bar{r}_{i(j)}$ as the global weight in this case. To consider this extreme situation, we generalize the definition of the global weight for CNR-U as follows:

$$w_{i(j)} := 1 + \bar{r}_{i(j)} \tag{12}$$

when $\bar{r}_{i(j)} = 0$, our loss function reduces to a normal non-weighted case, meaning $w_{i(j)} = 1$. In this way, the learning process can be carried out smoothly even if user i have no ratings on all the neighbors. The global weight of CNR-I also has a similar

Table 1
Three categories of loss functions for the sub-models.

Recommenders	Loss functions	Parameters
CNR-U	$\min \sum_{i,j} w_{i(j)} \left(e_{ij} - (p_i + k_i u_i)^T q_j\right)^2 + \lambda \sum \|\theta\|^2$	$\theta \in \{p_i, u_i, q_j\}$
CNR-I	$\min \sum_{i,j} w_{(i)j} (e_{ij} - p_i^T (q_j + k_j v_j))^2 + \lambda \sum \|\theta\|^2$	$\theta \in \{p_i, q_j, v_j\}$
CNR-UI	$\min \sum_{i,j} w_{ij} (e_{ij} - (p_i + k_i u_i)^T (q_j + k_j v_j))^2 + \lambda \sum \|\theta\|^2$	$\theta \in \{p_i, u_i, q_j, v_j\}$

Table 2
The descriptions of the notations.

The notations	The relevant descriptions
α	the number of users
β	the number of items
f	the dimension number in feature matrices
$\mathbf{P}$	the $\alpha \times f$ user matrix, its ith row p_i indicates user i
$\mathbf{Q}$	the $\beta \times f$ item matrix, its jth row q_j denotes item j
$\mathbf{U}$	the $\alpha \times f$ user neighbor matrix, the ith row u_i means neighbor i
$\mathbf{V}$	the $\beta \times f$ item neighbor matrix, the jth row v_j is neighbor j
$\widetilde{\mathbf{P}}$	an $\alpha \times f$ enhanced user matrix, with its ith row $\widetilde{p}_i := p_i + k_i u_i$
$\widetilde{\mathbf{Q}}$	a $\beta \times f$ enhanced item matrix, given $\widetilde{q}_j := q_j + k_j v_j$ as its jth row
$\widetilde{\mathbf{W}}$	an $\alpha \times \beta$ weight matrix in CNR-U, element $\widetilde{w}_{ij} := w_{i(j)}$ is defined by Eq. (12)
$\widehat{\mathbf{W}}$	an $\alpha \times \beta$ weight matrix in CNR-I, $\widehat{w}_{ij} := w_{(i)j}$ is given in Eq. (13)
$\overline{\mathbf{W}}$	an $\alpha \times \beta$ weight matrix in CNR-UI, $\overline{w}_{ij} := w_{ij}$ with its definition in Eq. (14)
$\mathbf{E}$	the $\alpha \times \beta$ implicit feedback matrix with element e_{ij} in Table 1
$\mathbf{I}$	an identity matrix with $f \times f$ dimensions
$i\cdot$	the subscript of the ith row vector, e.g. $\widetilde{\mathbf{W}}_{i\cdot}$ is the ith row vector of $\widetilde{\mathbf{W}}$
$\cdot j$	the subscript of the jth column vector, e.g. $\mathbf{E}_{\cdot j}$ is the jth column vector of $\mathbf{E}$
$i\circ$	the subscript of a $\beta \times \beta$ diagonal matrix adopting a row vector as its elements, e.g. the elements on the diagonal of $\widehat{\mathbf{W}}_{i\circ}$ are the same as are those in $\widehat{\mathbf{W}}_{i\cdot}$
$\circ j$	the subscript of an $\alpha \times \alpha$ diagonal matrix using a column vector as its elements, e.g. the elements on the diagonal of $\overline{\mathbf{W}}_{\circ j}$ derive from $\overline{\mathbf{W}}_{\cdot j}$

format to address this cold-start problem as defined in Eq. (13):

$$w_{(i)j} := 1 + \bar{r}_{(i)j} = 1 + \sum_{n=1}^{N} \frac{k_{in} \cdot r_{(in)j}}{\sum_{m=1}^{N} k_{im}} \tag{13}$$

where the meaning of $r_{(in)j}$ is quite different to $r_{i(jn)}$ in CNR-U, which means the rating given by the nth neighbor of user i to item j. To unify the CNR family, we arrange the global weight for CNR-UI as follows:

$$w_{ij} := 1 + \sum_{n=1}^{N} \frac{k_{jn} \cdot r_{i(jn)} + k_{in} \cdot r_{(in)j}}{\sum_{m=1}^{N} (k_{im} + k_{jm})} \tag{14}$$

here all the ratings of both user neighbors and item neighbors contribute to the global weight w_{ij}.

By now, all the information from context neighbors has been fully utilized and we are able to define the loss functions for the sub-models of three recommenders as Table 1 illustrates. Here e_{ij} indicates the implicit feedback: $e_{ij} = 1$ if user i has an action on item j, else $e_{ij} = 0$. λ is the factor of $L2$-regularization terms, which can be adjusted in the range of (0, 1]. Because only the nth sub-model is considered, the symbol n in subscript is eliminated for convenient analysis. To minimize the loss functions, we differentiate them to find the analytic expressions for all the parameters. The notations used in the solutions are first described in Table 2. For CNR-U, there are three factors whose expressions are derived from differentiation as follows:

$$p_i = \left(\mathbf{E}_{i\cdot} - k_i \mathbf{U}_{i\cdot} \mathbf{Q}^T\right) \widetilde{\mathbf{W}}_{i\circ} \mathbf{Q} \left(\mathbf{Q}^T \widetilde{\mathbf{W}}_{i\circ} \mathbf{Q} + \lambda \mathbf{I}\right)^{-1} \tag{15}$$

$$u_i = \left(\mathbf{E}_{i\cdot} - \mathbf{P}_{i\cdot} \mathbf{Q}^T\right) k_i \widetilde{\mathbf{W}}_{i\circ} \mathbf{Q} \left(\mathbf{Q}^T (k_i^2 \widetilde{\mathbf{W}}_{i\circ}) \mathbf{Q} + \lambda \mathbf{I}\right)^{-1} \tag{16}$$

$$q_j = \mathbf{E}_{\cdot j}^T \widetilde{\mathbf{W}}_{\circ j} \widetilde{\mathbf{P}} \left(\widetilde{\mathbf{P}}^T \widetilde{\mathbf{W}}_{\circ j} \widetilde{\mathbf{P}} + \lambda \mathbf{I}\right)^{-1} \tag{17}$$

The solutions of parameters in CNR-I are given as follows:

$$p_i = \mathbf{E}_{i\cdot} \widehat{\mathbf{W}}_{i\circ} \widetilde{\mathbf{Q}} \left(\widetilde{\mathbf{Q}}^T \widehat{\mathbf{W}}_{i\circ} \widetilde{\mathbf{Q}} + \lambda \mathbf{I}\right)^{-1} \tag{18}$$

$$q_j = \left(\mathbf{E}_{\cdot j} - k_j \mathbf{P} \mathbf{V}_{\cdot j}\right)^T \widehat{\mathbf{W}}_{\circ j} \mathbf{P} \left(\mathbf{P}^T \widehat{\mathbf{W}}_{\circ j} \mathbf{P} + \lambda \mathbf{I}\right)^{-1} \tag{19}$$

$$v_j = \left(\mathbf{E}_{\cdot j} - \mathbf{P}\mathbf{Q}_{\cdot j}\right)^T k_j \widehat{\mathbf{W}}_{\circ j}\mathbf{P}\left(\mathbf{P}^T(k_j^2\widehat{\mathbf{W}}_{\circ j})\mathbf{P} + \lambda\mathbf{I}\right)^{-1} \tag{20}$$

Finally, CNR-UI has four factor expressions more complicated than those of the first two methods:

$$p_i = \left(\mathbf{E}_{i\cdot} - k_i\mathbf{U}_{i\cdot}\widetilde{\mathbf{Q}}^T\right)\overline{\mathbf{W}}_{i\circ}\widetilde{\mathbf{Q}}\left(\widetilde{\mathbf{Q}}^T\overline{\mathbf{W}}_{i\circ}\widetilde{\mathbf{Q}} + \lambda\mathbf{I}\right)^{-1} \tag{21}$$

$$q_j = \left(\mathbf{E}_{\cdot j} - k_j\widetilde{\mathbf{P}}\mathbf{V}_{\cdot j}\right)^T\overline{\mathbf{W}}_{\circ j}\widetilde{\mathbf{P}}\left(\widetilde{\mathbf{P}}^T\overline{\mathbf{W}}_{\circ j}\widetilde{\mathbf{P}} + \lambda\mathbf{I}\right)^{-1} \tag{22}$$

$$u_i = \left(\mathbf{E}_{i\cdot} - \mathbf{P}_{i\cdot}\widetilde{\mathbf{Q}}^T\right)k_i\overline{\mathbf{W}}_{i\circ}\widetilde{\mathbf{Q}}\left(\widetilde{\mathbf{Q}}^T(k_i^2\overline{\mathbf{W}}_{i\circ})\widetilde{\mathbf{Q}} + \lambda\mathbf{I}\right)^{-1} \tag{23}$$

$$v_j = \left(\mathbf{E}_{\cdot j} - \widetilde{\mathbf{P}}\mathbf{Q}_{\cdot j}\right)^T k_j\overline{\mathbf{W}}_{\circ j}\widetilde{\mathbf{P}}\left(\widetilde{\mathbf{P}}^T(k_j^2\overline{\mathbf{W}}_{\circ j})\widetilde{\mathbf{P}} + \lambda\mathbf{I}\right)^{-1} \tag{24}$$

The local recommendation results can be obtained by substituting these parameter solutions into the predicted functions in Eqs. (8)–(10). The aggregation approaches of the local results and the analysis of our algorithm are coming in the next section.

3.4. Recommendation combination and complexity analysis

The local results from N sub-recommenders can be seen as the individual opinions given by each group interaction, which contribute to the final decision. It should be noticed that the nth sub-recommender is closely related to two kernel weights: k_{in} and k_{jn}. In other words, the importance of the nth local result can be simply reflected by k_{in} and k_{jn}. To consider both of them, we adopt their product $k_{in}k_{jn}$ as the weight of the nth local result. The weighted values are summed and normalized to provide the final recommendation as follows:

$$\hat{y}_{ij} = \sum_{n=1}^{N} \frac{k_{in}k_{jn}\cdot\hat{y}_{ij}^{(n)}}{\sum_{m=1}^{N} k_{im}k_{jm}} \tag{25}$$

where $\hat{y}_{ij}$ is the weighted average value of N sub-models. The summation in the denominator plays the role of normalization so that the ranges of the final predicted scores are bounded. This way of combining local results makes it possible to train each of the sub-models in parallel, which means that the core complexity of our algorithm is largely reduced and nearly equal to the complexity of one sub-model.

To explain the whole procedure of recommendation, we illustrate the details of CNR-U as an example in Algorithm 1. CNR-U consists of three parts: *Initialization, Training sub-models*, and *Recommendation combination. The initialization part (Line3–9)* aims to search context neighbors and compute kernel weights. For each user i and her nth neighbor, we compute the similarity c_{in} between them and convert c_{in} into k_{in} by kernel smoothing. Similar to c_{in} and k_{in}, c_{jn} and k_{jn} are calculated for items. Then, k_{in} is directly used in training, whereas k_{jn} hides in $\widetilde{\mathbf{W}}$ for parameter updating. *In the training processes (Line10–22), N* sub-models can be trained in parallel because each of them has their own parameters that are different to those of the others. The goal of learning the nth sub-model is to update three matrices: **Q, P**, and **U** sequentially by coordinate descent. **Q** is first updated according to Eq. (17) because the user matrix **P** and neighbor matrix **U** can be combined to be matrix $\widetilde{\mathbf{P}}$, which is more convenient than updating the other two matrices first. Then, the new values in matrix **Q** can be used immediately to update **P** and **U**. To improve efficiency, a temporary matrix **T** is employed to store the parts that can be pre-calculated. Then, **P** and **U** are updated according to Eqs. (15) and (16), respectively. *Finally, the Recommendation combination part (Line23–33)* combines the weighted results of N sub-models to provide a final prediction $\hat{y}_{ij}$ that indicates the preference of user i on item j. The kernel product $k_{in}k_{jn}$ is the weight of the nth predicted value $\hat{y}_{ij}^{(n)}$, which can be normalized by a pre-calculated denominator d_{ij}. Thus far, $\hat{y}_{ij}$ is employed as the output of CNR-U, meaning that the entire algorithm has been finished.

For complexity analysis, it can be observed that main cost comes from training N sub-models, however, we just need to estimate the computational consumption of the nth sub-model because each sub-model is learned in parallel. In the training process of the nth sub-model, consider the update of item factor q_j first, in which calculating $\mathbf{E}_{\cdot j}^T\widetilde{\mathbf{W}}_{\circ j}\widetilde{\mathbf{P}}$ and $\widetilde{\mathbf{P}}^T\widetilde{\mathbf{W}}_{\circ j}\widetilde{\mathbf{P}}$ are time consuming. However, similar to the discussion in WRMF [11], they can be expressed as $\mathbf{E}_{\cdot j}^T\widetilde{\mathbf{P}} + \mathbf{E}_{\cdot j}^T(\widetilde{\mathbf{W}}_{\circ j} - \mathbf{I})\widetilde{\mathbf{P}}$ and $\widetilde{\mathbf{P}}^T\widetilde{\mathbf{P}} + \widetilde{\mathbf{P}}^T(\widetilde{\mathbf{W}}_{\circ j} - \mathbf{I})\widetilde{\mathbf{P}}$ so that $(\widetilde{\mathbf{W}}_{\circ j} - \mathbf{I})$ has few non-zero elements on its diagonal for complexity reduction. By the pre-computation of $\mathbf{E}_{\cdot j}^T\widetilde{\mathbf{P}}$ and $\widetilde{\mathbf{P}}^T\widetilde{\mathbf{P}}$, the running time can be reduced to $\mathcal{O}(f^2 n_\alpha)$, where f is feature dimension and n_α represents the number of non-zero elements in $(\widetilde{\mathbf{W}}_{\circ j} - \mathbf{I})$. Because the inversion operation of computing $\left(\widetilde{\mathbf{P}}^T\widetilde{\mathbf{W}}_{\circ j}\widetilde{\mathbf{P}} + \lambda\mathbf{I}\right)^{-1}$ has an upper bound of $\mathcal{O}(f^3)$, whereas the final multiplication also takes $\mathcal{O}(f^3)$ to be finished. The update cost of each item is in the order of $\mathcal{O}(f^2 n_\alpha + f^3)$. In addition, there are β item factors. Hence, all the item parameters can be updated in $\mathcal{O}\left(\beta(f^2 n_\alpha + f^3)\right)$. When dealing with p_i and u_i, **T** is helpful in simplifying their forms to accelerate the learning process. Similar to the analysis of item factors, the complexity of solving p_i and u_i is in the same order of $\mathcal{O}\left(\alpha(f^2 n_\beta + f^3)\right)$, here n_β denotes the number of non-zero elements in $(\widetilde{\mathbf{W}}_{i\circ} - \mathbf{I})$. In the final step, the prediction of $\hat{y}_{ij}^{(n)}$ requires $\mathcal{O}(3f)$. Thus,

Algorithm 1 Context Neighbor Recommender with User aspect (CNR-U).

Input: N - the number of context neighbors, λ - the regularization factor,
K - the kernel type (function), h - the kernel width.
Define: α - the user number, β - the item number,
$\mathbf{T}$ - an $\alpha \times \beta$ temporary matrix, $\mathbf{D}$ - an $\alpha \times \beta$ denominator matrix.
▷ Initialization:
Calculate the context similarities to find out N context neighbors.
for $n = 1$ **to** N **do**
  for $i = 1$ **to** α **do**
    $k_{in} = K\left(\frac{1}{h}(1 - c_{in})\right)$ // c_{in}- the similarity between user i and neighbor n
  for $j = 1$ **to** β **do**
    $k_{jn} = K\left(\frac{1}{h}(1 - c_{jn})\right)$ // c_{jn}- the similarity between item j and neighbor n
Compute $\widetilde{\mathbf{W}}$ *according to* Eq. (12)
▷ Training N sub-models:
for all $n \in \{1, \cdots, N\}$ **in parallel do**
  // The subscript n which denotes the nth sub-model is omitted.
  repeat
    $\widetilde{\mathbf{P}}_{i\cdot} = \mathbf{P}_{i\cdot} + k_i \mathbf{U}_{i\cdot}$ // calculate $\widetilde{\mathbf{P}}$
    for $j = 1$ **to** β **do**
      $q_j = \mathbf{E}_{\cdot j}^T \widetilde{\mathbf{W}}_{\circ j} \widetilde{\mathbf{P}} \left(\widetilde{\mathbf{P}}^T \widetilde{\mathbf{W}}_{\circ j} \widetilde{\mathbf{P}} + \lambda \mathbf{I}\right)^{-1}$ // update $\mathbf{Q}$
    $\mathbf{T}_{i\cdot} = \mathbf{E}_{i\cdot} - k_i \mathbf{U}_{i\cdot} \mathbf{Q}^T$ // $\mathbf{T}$ stores the intermediate values for fast update.
    for $i = 1$ **to** α **do**
      $p_i = \mathbf{T}_{i\cdot} \widetilde{\mathbf{W}}_{i\circ} \mathbf{Q} \left(\mathbf{Q}^T \widetilde{\mathbf{W}}_{i\circ} \mathbf{Q} + \lambda \mathbf{I}\right)^{-1}$ // update $\mathbf{P}$
    $\mathbf{T}_{i\cdot} = \mathbf{E}_{i\cdot} - \mathbf{P}_{i\cdot} \mathbf{Q}^T$
    for $i = 1$ **to** α **do**
      $u_i = \mathbf{T}_{i\cdot} (k_i \widetilde{\mathbf{W}}_{i\circ}) \mathbf{Q} \left(\mathbf{Q}^T (k_i^2 \widetilde{\mathbf{W}}_{i\circ}) \mathbf{Q} + \lambda \mathbf{I}\right)^{-1}$ // update $\mathbf{U}$
  until stopping criterion met
▷ Recommendation combination:
for $i = 1$ **to** α **do**
  for $j = 1$ **to** β **do**
    $d_{ij} = 0$ // d_{ij} - the pre-calculated denominator
    for $m = 1$ **to** N **do**
      d_{ij} += $k_{im} k_{jm}$ // d_{ij} is the element of $\mathbf{D}$
for $i = 1$ **to** α **do**
  for $j = 1$ **to** β **do**
    $\hat{y}_{ij} = 0$ //the final predicted result from user i to item j
    for $n = 1$ **to** N **do**
      $\hat{y}_{ij}^{(n)} = (p_{in} + k_{in} \cdot u_{in})^T q_{jn}$ // predict the local score $\hat{y}_{ij}^{(n)}$
      $\hat{y}_{ij}$ += $\frac{k_{in} k_{jn}}{d_{ij}} \hat{y}_{ij}^{(n)}$ //add $\hat{y}_{ij}^{(n)}$ from the nth sub-model
Output: $\mathbf{Y}$ - an $\alpha \times \beta$ predicted matrix using $\hat{y}_{ij}$ as its elements.

the recommendation combination takes $\mathcal{O}(3\alpha\beta N f)$. To summarize, the total complexity of CNR-U is $\mathcal{O}(\mathcal{L}f^3 + \mathcal{M}f^2 + \mathcal{N}f)$, where $\mathcal{L} = 2\alpha + \beta$, $\mathcal{M} = 2\alpha n_\beta + \beta n_\alpha$ and $\mathcal{N} = 3\alpha\beta N$, respectively.

4. Experiments and results

We conducted experiments on three real world datasets from the Yahoo! Webscope program[1] and the MovieLens (ML) web-site.[2] The *Movie* (R4) and *Music* (C15-track1) datasets provided by Yahoo! were utilized throughout the experiments because they involved rich contexts to evaluate our approach, whereas the *ML-Latest* dataset from MovieLens played a role in comparing methods because it has been widely used in the existing literature as an ideal dataset for comparisons.

The *ML-Latest* dataset (small version) describes 5-star rating and free-text tagging activity from MovieLens. The rating timestamps were taken to be user contexts that can be extended to generate four contexts: *year, month, hour*, and *day*. In contrast, items carry contexts such as *genre* and *tag* to express more personalized information other than ratings. Both user and item contexts were involved in the comparisons of methods. The R4 dataset is also about movies and contains more user profiles than *ML-Latest*, including information such as *user id, birthyear*, and *gender*. In addition to user information, the contextual information of items in this dataset is far richer than that of users. Thus, we chose all the user contexts and eight representative item contexts, such as *genres, actors*, and *directors*, to carry out these experiments. In contrast to the two *Movie* datasets, the large-scale *Music* dataset (C15-track1) provides a snapshot of ratings for various musical items of four different types: *tracks, albums, artists*, and *genres*. These items are related to one another within a hierarchy, e.g., given a *track*, we know its *album*, performing *artist* and associated *genres*. Such descriptive information can be seen as the contexts of items, whereas the user actions, such as *rating time* and *rating number*, are treated as user contexts. Since it has been argued that similar rating time reflects similar user preferences [17], we employed binning for reorganizing the *rating time* to

[1] http://webscope.sandbox.yahoo.com.
[2] https://movielens.org/.

Table 3
The statistics of the experimental datasets.

Datasets	Users	Items	Observations	Contexts
ML-Latest	706	8552	100,023	$user - 5$, $item - 3$
Movie	7642	11,915	221,367	$user - 3$, $item - 8$
Track	53,109	159,687	16,695,188	$user - 4$, $item - 4$
Album	89,615	63,890	18,574,290	$user - 4$, $item - 4$

make full use of it. Specifically, we used week bins and month bins to divide the attribute *day offset* in the *rating time*, e.g., the *day offset* 3811 and 3818 are in the same week bin because the time distance between them is a week, namely, 7 days. Therefore, the *rating time* is expanded to provide more useful information. While the *Music* dataset involves four different items, we selected *tracks* and *albums* for our experiments because the rating data associated with them are much richer than are those of *artists* and *genres*. Hence, two sub-datasets, *Track* and *Album*, were created from the original *Music* dataset. For the experimental setup, 80% of the records in each dataset were randomly chosen for training, whereas the remaining 20% of observations were left for testing. This five fold cross-validation strategy was adopted in all of our experiments. The two movie datasets provided sufficient observations to be divided into a training set and a test set. However, the *Music* dataset requires a pre-filtering operation to make sure that there are enough items for both training and testing. Hence, by assuming that there were at least 10 items for each user in the test set, we randomly sampled 20 million observations to pick out users who had rated at least 30 items and items that had been rated at least 20 times. After the pre-filtering works, the statistics of four datasets were summarized in Table 3.

For evaluation metrics, both classification and ranking measures [34] are widely accepted in recommendation. Because we focused on classifying accuracy rather than sorting accuracy of the recommendation results, common classification metrics such as *Precision, Recall, F1-Score*, and *AUC* (Area Under the ROC Curve) were employed to evaluate our approach. Recommendation lists from the *top*-1 to the *top*-10 of each user can be generated for evaluation because the pre-filtering operation had ensured that there were sufficient items from each user for testing. In the following experiments, we first investigated the optimal settings of CNR for neighbor numbers and kernel parameters in Sections 4.1 and 4.2. Three representative metrics, including *Precision@5, Precision@10*, and *AUC*, were adopted to evaluate the results of the three datasets: *Movie, Track*, and *Album*. This was because we would like to determine the best settings simply by a few experiments to further carry out comparisons of our methods in Section 4.3. Hence, in the final experiment, all the metrics and datasets were employed to evaluate both our approach and other representative methods and to demonstrate the advantages of our approach.

4.1. The number of neighbors

The numbers of neighbors, or the numbers of groups, are the important hyper-parameters to be confirmed for different CNR recommenders. To obtain the optimal values, we gradually increased the number of neighbors (groups) from 1 to 10 in *top*-10 recommendations, because we assumed that one neighbor would provide at least one valuable recommendation. For preliminaries, parameters of CNR were initialized with a Gaussian Distribution $\mathcal{N}(0, 1)$. The feature dimension f and regularization factor λ were set to be 20 and 0.0008, respectively. In the kernel settings, the Epanechnikov kernel with width $h = 0.8$ was accepted because it was the common configuration for various applications of kernel smoothing [41]. We applied the above settings to run three recommenders (CNR-U, CNR-I, and CNR-UI) in 30 iterations and obtained the results as shown in Fig. 2.

The optimal neighbor number refers to a specific neighbor number, with which a CNR recommender can achieve its best performances on most metrics. In the *Movie* dataset, CNR-U took 4 as its optimal neighbor number, because it performed best on *Precision@5* and *Precision@10* among three metrics. For CNR-I and CNR-UI, they required 9 and 8 neighbors respectively for best results. The reason was that the item contexts were much richer than those of users, in other words, more item neighbors were needed to represent more contextual information from movies. The cases of the *Music* dataset were just opposite to those of movies. CNR-U required more neighbors than both CNR-I and CNR-UI did because the user contexts were more flexible than were those of items. Such flexible contexts were the *rating time* that had been refined by the binning approaches in the experiment preparations. By utilizing the user contexts, CNR-U employed 9 neighbors to gain the best performances in the *Track* data, whereas CNR-I and CNR-UI reached their peaks with 6 neighbors. In the *Album* dataset, the optimal neighbor numbers of CNR-U, CNR-I, and CNR-UI were 7, 3, and 2, respectively. From the perspective of average performances, CNR-U had the advantages of three metrics in all the datasets except *Precision@5* in the *Track* data. It was consistent with such common sense—a user neighbor was more likely to provide valuable suggestions than an item neighbor. Furthermore, in contrast to the user neighbors in CNR-U, the item neighbors in CNR-UI may distort the information provided by the existing user neighbors. This phenomenon demonstrated that targeted modeling [14] was more important than simply increasing the model complexity.

To summarize, the results of this experiment indicated that the optimal neighbor number of the CNR family varied according to different datasets. In fact, even context neighbors had replaced the original contextual factors to implement

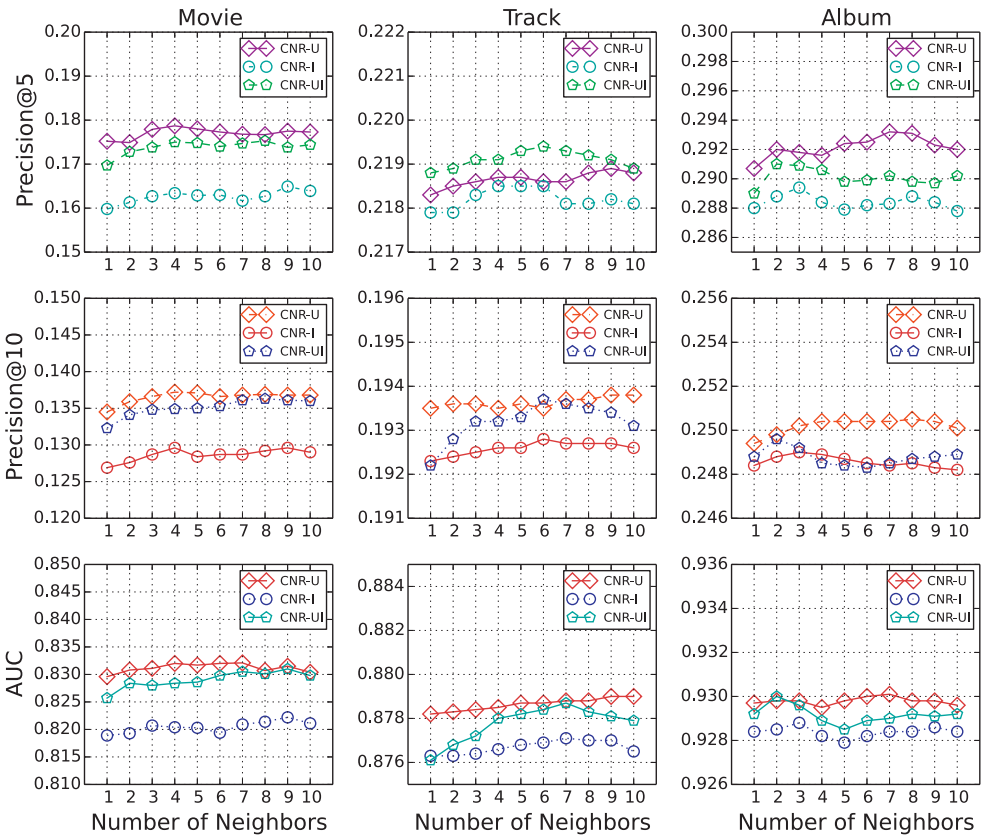

Fig. 2. The performances of CNR with different numbers of neighbors.

domain independent, the performances still reflected the characteristic of the original contexts. Because some valuable information of the original contexts was retained by context neighbors in another forms, e.g., context similarities. Consequently, the intrinsic meanings of the original contexts can be conveyed to our recommenders via context neighbors, which means that the characteristic of specific contexts can be reflected. In addition to the investigation of neighbor number, the kernel settings can also affect the performances. Because of the advantages of CNR-U, it was selected as the representative approach to investigate the influences of kernels in the next section.

4.2. The influences of kernels

Three categories of kernel functions, including the Epanechnikov (K_E), the Biweight (K_B), and the Gaussian (K_G) kernels, were employed in our experiments. The only parameter in kernel functions was the kernel width (or bandwidth) which determined the spread of the kernel. The choice of bandwidth was so important that an inappropriate width may cause the data to be under-smoothed or over-smoothed [41]. In CNR, the kernel weights played the roles of distance measure, loss optimization, and recommendation combining. Hence, the most suitable kernel types and widths must be figured out for best performances. Because the inputs of kernel smoothing were distances (or similarities) in the range of [0, 1], we restricted the kernel width in the range of [0.8, 1.2] to avoid distorting the distances. For each kernel, we gradually increased the bandwidth by step-size 0.1 to record the performances. The best neighbor number and other settings were the same as those in Section 4.1. Then, we ran CNR-U as the representative approach to obtain the impacts of kernel changing that were illustrated in Fig. 3.

On *Precision@5*, the Epanechnikov kernel performed well with width 0.8 in all the datasets, whereas the Biweight kernel achieved its best performances by using bandwidth 1.0 in the first two datasets. On *Precision@10*, the Biweight kernel gained significant advantages compared to the other two kernels. Consider the performances on all the metrics, both the Epanechnikov kernel and the Biweight kernel outperformed the Gaussian kernel; this confirmed that the Gaussian kernel was not suitable for our recommenders. Generally, the Biweight kernel had shown its adaptability for CNR in all the datasets. Hence, we chose it as our kernel smoothing function in the following experiments. To simplify the settings of parameters, the kernel width of each dataset was set to be the same for three CNR models according to the comprehensive performances in Fig. 3.

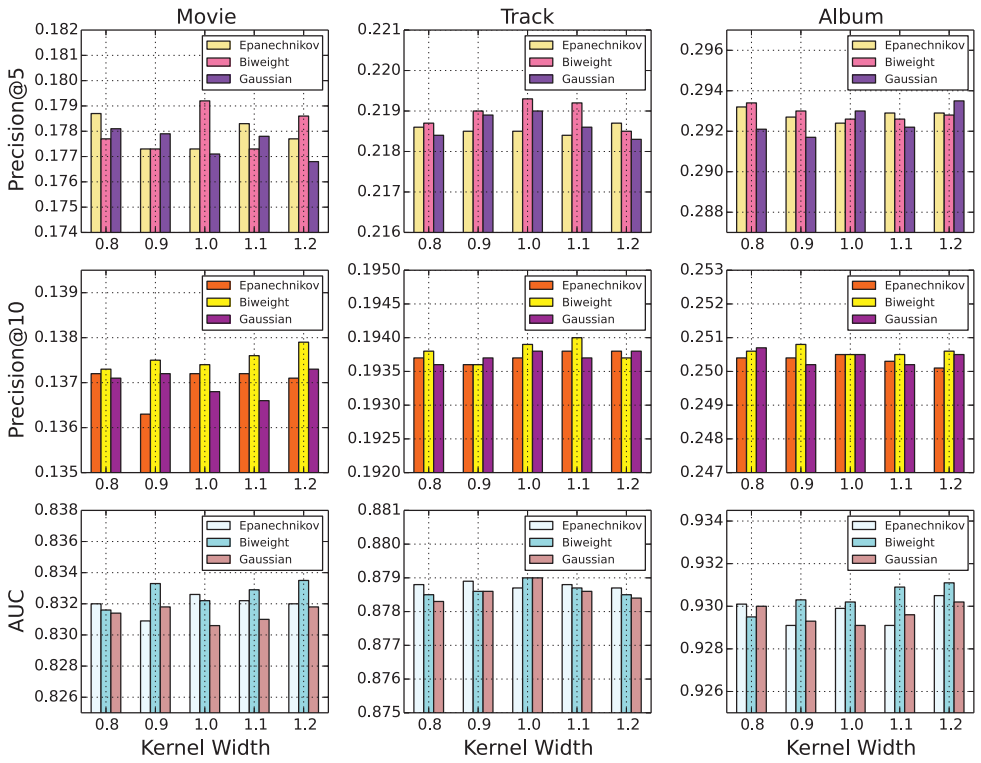

Fig. 3. The performances of CNR with different kernel settings.

Then, we were able to carry out comparisons between CNR and other advanced approaches according to our preliminary experiments.

4.3. Method comparisons

In this section, several advanced methods were selected to be compared with our approach according to their relations to the CNR family. First, three ICF algorithms involve BPR-MF [31], WRMF [11], and SLIM [24] were chosen because our approach was based on the ICF techniques. BPR-MF was a factorization model for distinguishing the "like" and "dislike" items like our method, whereas WRMF utilized a way of training parameters that was similar to ours. In contrast, SLIM leveraged rating-neighbors for suggestions to assist factorization algorithm, which was comparable to our context neighbors. In addition to the above single ICF models, another state-of-the-art ensemble ICF method named LCR [18] was employed for comparisons. Because LCR also considered neighborhood information and leveraged kernel smoothing to benefit sub low-rank models as the CNR family did. Finally, we chose two more context-aware recommenders to enrich the experiments and make evaluation reliable. One was the extension of SLIM called SSLIM [25] that introduced contextual factors for neighbors to assist recommendation. The second efficient and widely used approach was PITF [10,32], whose contexts were often reduced as one dimensional contextual factors. Thus, we followed this way to adopt the average value of contextual factors to be $x_c^{(i)}$ or $x_c^{(j)}$ in Eq. (2) in the learning processes of PITF. The hyper-parameters of each methods were appropriately adjusted to help these approaches reach their best performances, whose settings were listed in Table 4.

In most cases, the values of *feature dimensions* were proportional to the numbers of users and items in the datasets, whereas SLIM and SSLIM did not require *feature dimensions* because they employed the whole coefficient matrix for training. The *regularization factors* had fixed values throughout the learning processes. In contrast, the *learning rates* needed by SGD (Stochastic Gradient Descent) based algorithms such as BPR-MF, PITF, and LCR were initialized to be some values and changed dynamically according to the speeds of convergence. The other approaches were not affected by the *learning rates* because their parameters were updated by analytic expressions based on Coordinate Descent. There were six methods leveraging neighborhood information. The SLIM and CNR family required only a few neighbors to achieve their best performances, whereas LCR asked for more neighbors to implement local low-rank approximation. The performances of all the models in four datasets were shown in Figs. 4, 5, 6, and 7 , respectively.

ML-Latest was selected as the first comparison dataset shown in Fig. 4 for the reason that it was widely utilized by many state-of-the-art algorithms, in which WRMF performed well on the *AUC* metric whereas LCR achieved outstanding performances and even exceeded CNR-I and CNR-UI on the *Precision* and *Recall* measures. However, the two context-aware approaches, SSLIM and PITF, had no advantages in contrast to the above ICF-based methods. Because *ML-Latest* was born for

Table 4
The settings of hyper-parameters for all the comparative methods. The values of *Feature dimensions* and *Other parameters* were expressed in a quaternion form {$value_L$, $value_M$, $value_T$, $value_A$} to denote the corresponding settings of *ML-Latest, Movie, Track*, and *Album* dataset, respectively. Both *Regularization factors* and *Learning rates* were unified as the same values for four datasets except those of SLIM and SSLIM, whose regularization terms involved both $L1$ and $L2$ that were represented as {L_1, L_2}. In addition, WRMF employed α as the *rating scalar* while SLIM, LCR, and CNR had *neighbor number* (N) to be set. Furthermore, CNR utilized the Biweight kernel (K_B) with different values, whereas LCR employed the Epanechnikov kernel (K_E) with the same bandwidth according to the original settings by its authors.

Methods	*Feature dimensions*	*Regularization factors*	*Learning rates*	*Other parameters*
BPR-MF	{10, 8, 20, 16}	0.00008	0.016	–
WRMF	{15, 20, 30, 25}	0.0008	–	$\alpha = \{4.0, 1.0, 8.0, 5.0\}$
SLIM	–	{0.01, 0.001}	–	$N = \{2, 2, 4, 3\}$
SSLIM	–	{0.01, 0.001}	–	$N = \{3, 3, 5, 4\}$
PITF	{10, 8, 30, 25}	0.00008	0.018	–
LCR	{15, 18, 25, 20}	0.00008	0.016	$N = \{20, 30, 50, 40\}, K_E = \{0.8, 0.8, 0.8, 0.8\}$
CNR-U	{12, 16, 30, 25}	0.0008	–	$N = \{6, 4, 9, 7\}, K_B = \{0.9, 1.2, 1.0, 0.9\}$
CNR-I	{12, 16, 30, 25}	0.0008	–	$N = \{3, 9, 6, 3\}, K_B = \{0.9, 1.2, 1.0, 0.9\}$
CNR-UI	{12, 16, 30, 25}	0.0008	–	$N = \{5, 8, 6, 2\}, K_B = \{0.9, 1.2, 1.0, 0.9\}$

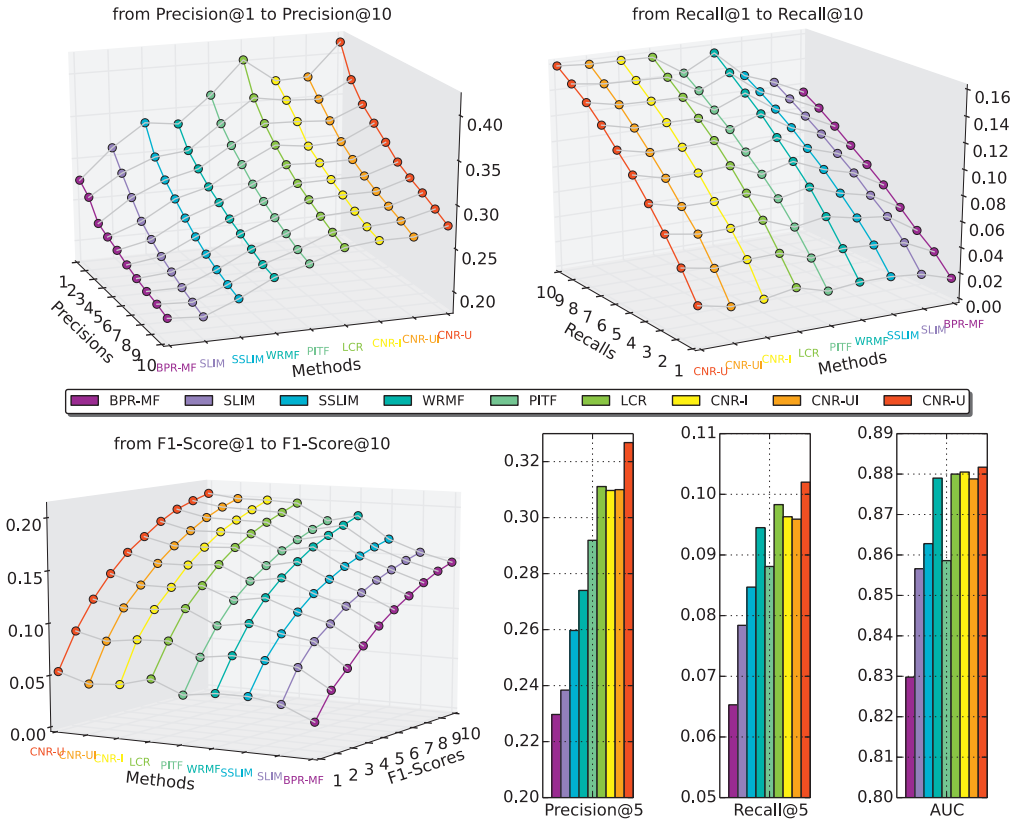

Fig. 4. Performances from *Top*-1 to *Top*-10 recommendations in the *ML-Latest* dataset.

collaborative filtering rather than context-aware recommendation. Specifically, from the data perspective, the average rating number per user (typically 140) in this dataset was large enough for ICF algorithms to mine user preferences, nevertheless, the context information in particular the item contexts that involved only 2488 tag applications and 19 movie genres, were insufficient for context-aware models to carry out recommendations. From the model perspective, the relatively high performances of WRMF were derived from its training way and weighted matrices, which were controlled by the constant α to precisely model user preferences. The LCR model actually consisted of many sub ICF models that enhanced the entire performances. Although SSLIM directly used contextual factors in the training process to refine SLIM, this way of leveraging contexts was rough compared to that of the CNR family. In contrast, the advantage of the CNR methods was that they

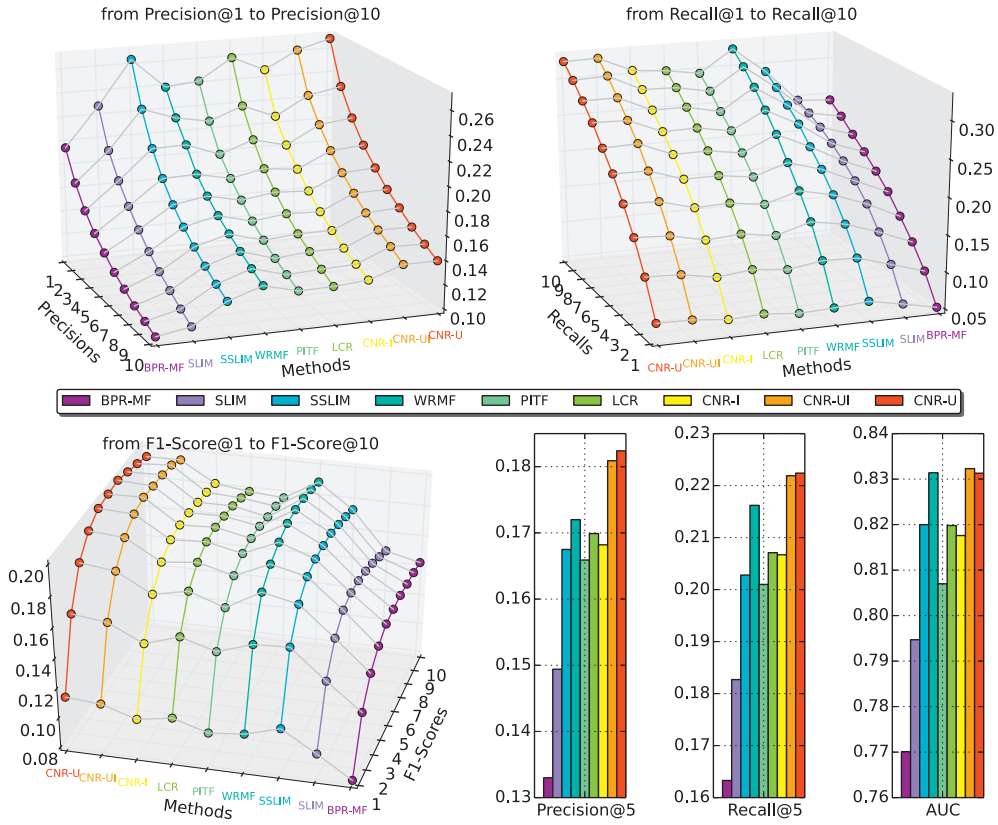

Fig. 5. Performances from *Top*-1 to *Top*-10 recommendations in the *Movie* dataset.

utilized contexts by changing them into context neighbors, which were different to the rating neighbors of the SLIM family. Although the efficiency of context neighbors was still relevant to the original contexts, context neighbors can largely reduce the negative impacts of original contextual factors such as the number, form, and meaning variations, which made our approach performed well.

In Fig. 5, WRMF continued to maintain its advantages in the *Movie* dataset compared to other models except CNR-U and CNR-UI. However, the performances of LCR and CNR-I were relatively lower than WRMF. The first reason was that the *Movie* dataset contained many more item contexts than user contexts as shown in Table 3. Moreover, the item contexts such as *list of directors* and *list of actors* involved lots of features, which were difficult to be integrated by CNR-I. Meanwhile, the average rating number per user (typically 27) were insufficient for the ICF algorithms such as BPR-MF, SLIM, and LCR to mine user preferences. Nonetheless, WRMF had an adjustable hyper-parameter α to control the weighted matrices for capturing valuable latent preferences. In contrast, CNR-U and CNR-UI performed better than any other models. Not only because the user contexts such as *birthyear* and *gender* were useful and simple to be integrated, but also because the user context neighbors were capable of offering more valuable suggestions than were the item context neighbors as discussed in Section 4.1. In a summary, the CNR family can take advantage of the ICF techniques and fully utilize the context information of the two movie datasets, which means that our recommenders have strong adaptability in either rating-oriented datasets or context-oriented datasets.

The performances in the *Track* (Fig. 6) and *Album* (Fig. 7) dataset were similar. Two context-aware recommenders, SSLIM and PITF outperformed other approaches except three CNR recommenders on most metrics. The reason from the data perspective was that the hierarchy contexts of the *Music* dataset, including *tracks, albums, artists*, and *genres*, had more standardized formats than those of the two movie datasets. These hierarchy contexts benefited PITF to achieve best performances in the *Album* dataset, because such context structure facilitated the combination of contexts into one dimension by PITF. However, SSLIM had the limitations on *AUC* in the two *Music* datasets. This phenomenon can be explained from the model angle that SSLIM expressed user preferences by learning aggregation coefficients from rating neighbors, which was quite different to the optimization of AUC that aimed to distinguish the “like” and “dislike” items. In contrast, the CNR family performed better than other methods. Because both the hierarchy contexts and the pre-processing work, such as binning the *rating time*, can benefit the calculations of kernel similarities to accurately reflect the neighbor relations. In fact, the essential

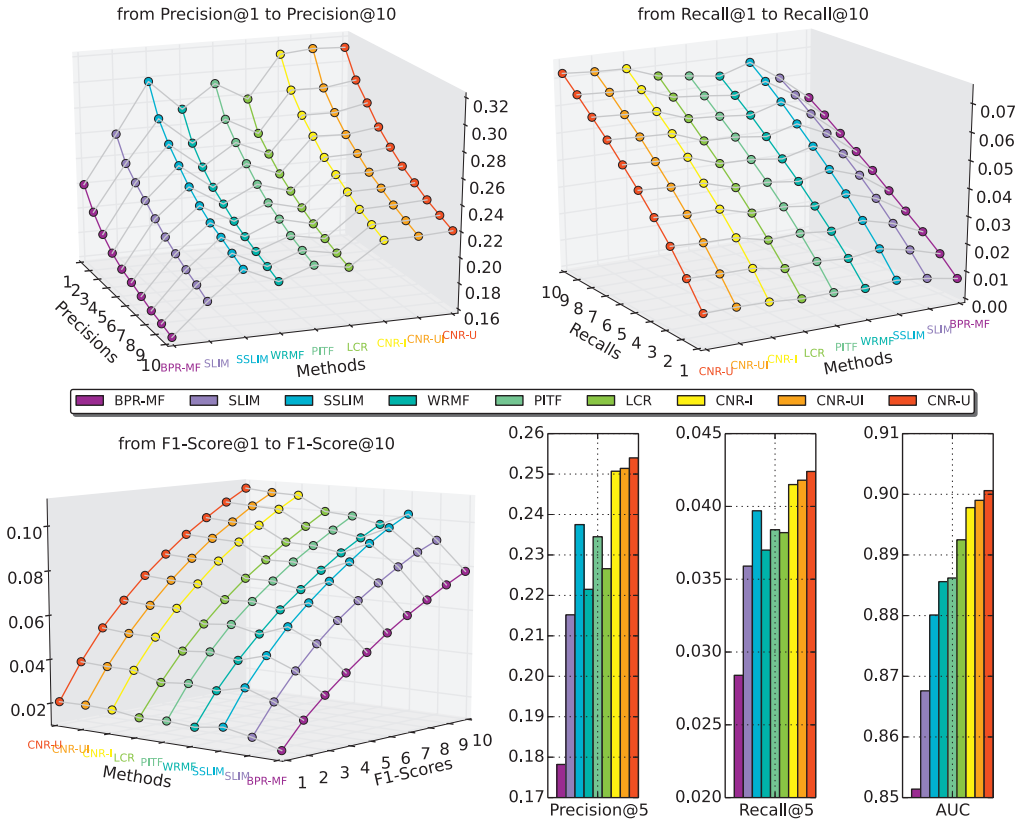

Fig. 6. Performances from *Top*-1 to *Top*-10 recommendations in the *Track* dataset.

reason was that our recommenders were able to mine user preferences via the steady structures of context neighbors. In this way, the dependence of original contextual factors was largely reduced. Therefore, the intrinsic meanings of the original contexts can be found out more easily.

In a summary, the CNR family performed steadily and had advantages in contrast to other models in all the datasets. The CNR-U recommender was the most powerful approach among them. This characteristic was precious because the item contexts may be unknown in practical applications. Analogously, we can use CNR-I if the user contexts were not provided. Our experiments demonstrated that the CNR family was flexible in the handling of different contexts according to their categories. Finally, if both user contexts and item contexts are available, we can simply run three CNR recommenders to further select the best one from them.

5. Conclusions and future work

In this work, we develop the CNR family which contains three categories of domain independent recommenders. One highlight of our approaches is introducing the context neighbors to create a domain independent learning framework. Another is implementing the group interactions to take the place of traditional user-item interactions. The evaluations in two real world datasets have shown the advantages of the CNR family. In addition, our recommenders are general and flexible because they can be applied in both non-context and context-aware recommendations.

As future work, we mainly focus on exploiting user actions in our models. Because the experimental results has proved that CNR-U outperformed the other two recommenders in most cases. Moreover, the user actions are more meaningful than the item contexts. In the modeling of user actions, the dynamic events such as browsing, clicking, and purchasing should be paid more attentions to mine the user preferences, whereas the static events such as rating and tagging are utilized to assist recommendations. In contrast to CNR-U, we will model the dynamic and the static events separately to establish an event recommender that focuses on precise modeling of user actions in our further research.

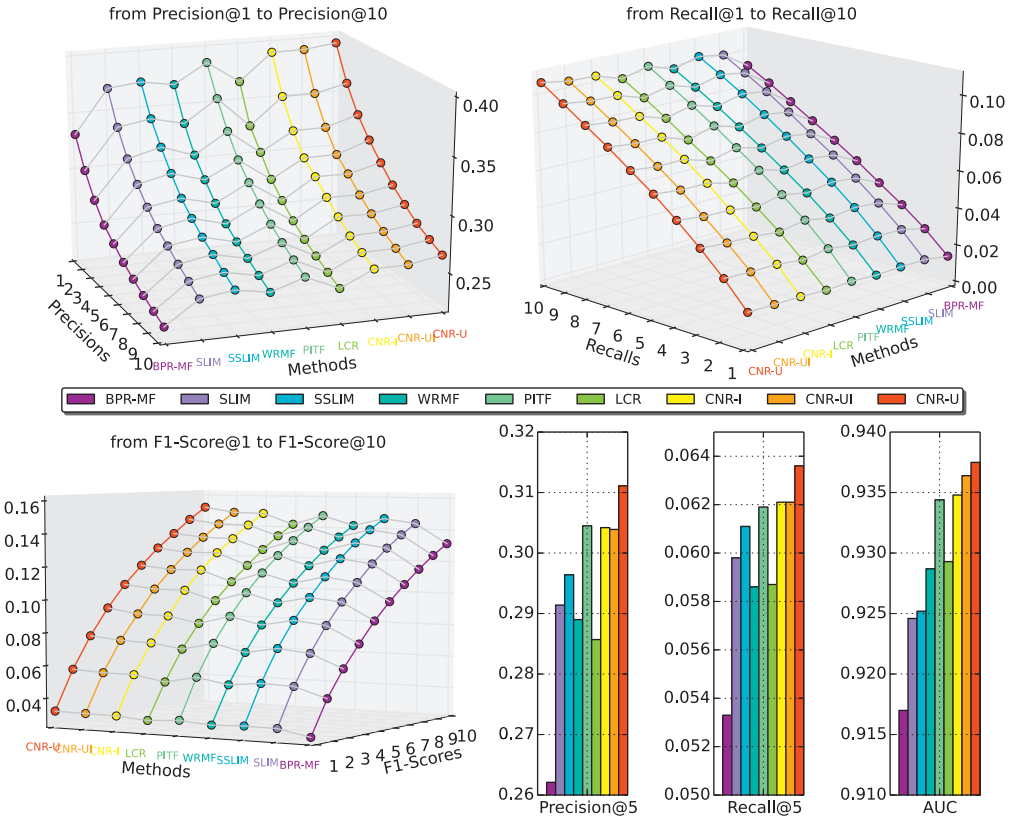

Fig. 7. Performances from *Top*-1 to *Top*-10 recommendations in the *Album* dataset.

Acknowledgments

This research is supported by the National Natural Science Foundation of China with Grant No. 61272277. We acknowledge the editors and other anonymous reviewers for insightful suggestions on this work.

References

[1] G. Adomavicius, R. Sankaranarayanan, S. Sen, A. Tuzhilin, Incorporating contextual information in recommender systems using a multidimensional approach, ACM Trans. Inf. Syst. 23 (1) (2005) 103–145, doi:10.1145/1055709.1055714.
[2] S. Balakrishnan, S. Chopra, Collaborative ranking, in: Proceedings of the Fifth ACM International Conference on Web Search and Data Mining, WSDM '12, ACM, NY, USA, Seattle, Washington, USA, 2012, pp. 143–152, doi:10.1145/2124295.2124314.
[3] L. Baltrunas, B. Ludwig, F. Ricci, Matrix factorization techniques for context aware recommendation, in: Proceedings of the Fifth ACM Conference on Recommender Systems, in: RecSys '11, ACM, Chicago, Illinois, USA, 2011, pp. 301–304, doi:10.1145/2043932.2043988.
[4] A. Bellogín, P. Castells, I. Cantador, Neighbor selection and weighting in user-based collaborative filtering: a performance prediction approach, ACM Trans. Web 8 (2) (2014). 12:1–12:30 doi: 10.1145/2579993.
[5] T. Chen, H. Li, Q. Yang, Y. Yu, General functional matrix factorization using gradient boosting, in: Proceedings of the 30th International Conference on Machine Learning (ICML-13), in: JMLR Workshop and Conference Proceedings, 28, Microtome Publishing, Brookline, MA, USA, Atlanta, Georgia, USA, 2013, pp. 436–444.
[6] N. Chowdhury, X. Cai, C. Luo, Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2015, Porto, Portugal, September 7–11, 2015, Proceedings, Part II, Springer International Publishing, Cham, pp. 3–18. doi:10.1007/978-3-319-23525-7_1.
[7] C. Desrosiers, G. Karypis, Recommender Systems Handbook, Springer US, Boston, MA, pp. 107–144. doi:10.1007/978-0-387-85820-3_4.
[8] Z. Gantner, L. Drumond, C. Freudenthaler, L. Schmidt-Thieme, Personalized ranking for non-uniformly sampled items, in: Proceedings of KDD Cup 2011 Competition, in: JMLR Workshop and Conference Proceedings, 18, Microtome Publishing, Brookline, MA, USA, 2012, pp. 231–247.
[9] B. Hidasi, D. Tikk, Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2012, Bristol, UK, September 24–28, 2012. Proceedings, Part II, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 67–82. doi:10.1007/978-3-642-33486-3_5.
[10] B. Hidasi, D. Tikk, General factorization framework for context-aware recommendations, Data Min. Knowl. Discov. 30 (2) (2015) 342–371, doi:10.1007/s10618-015-0417-y.
[11] Y. Hu, Y. Koren, C. Volinsky, Collaborative filtering for implicit feedback datasets, in: Proceedings of the 8th IEEE International Conference on Data Mining, in: ICDM '08, IEEE, Pisa, Italy, 2008, pp. 263–272, doi:10.1109/ICDM.2008.22.
[12] M. Jahrer, A. Töscher, Collaborative filtering ensemble for ranking., in: Proceedings of KDD Cup 2011 competition, in: JMLR Workshop and Conference Proceedings, 18, Microtome Publishing, Brookline, MA, USA, 2012, pp. 61–74.
[13] P. Jain, P. Netrapalli, S. Sanghavi, Low-rank matrix completion using alternating minimization, in: Proceedings of the Forty-fifth Annual ACM Symposium on Theory of Computing, in: STOC '13, ACM, Palo Alto, California, USA, 2013, pp. 665–674, doi:10.1145/2488608.2488693.

[14] M.N. Jelassi, S. Ben Yahia, E. Mephu Nguifo, Towards more targeted recommendations in folksonomies, Soc. Netw. Anal. Min. 5 (1) (2015) 68:1–68:18, doi:10.1007/s13278-015-0307-8.
[15] S. Kabbur, X. Ning, G. Karypis, Fism: factored item similarity models for top-n recommender systems, in: Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, in: KDD '13, ACM, Chicago, Illinois, USA, 2013, pp. 659–667, doi:10.1145/2487575.2487589.
[16] Y. Koren, Factorization meets the neighborhood: a multifaceted collaborative filtering model, in: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, in: KDD '08, ACM, NY, USA, Las Vegas, Nevada, USA, 2008, pp. 426–434, doi:10.1145/1401890.1401944.
[17] Y. Koren, Collaborative filtering with temporal dynamics, Commun. ACM 53 (4) (2010) 89–97, doi:10.1145/1721654.1721677.
[18] J. Lee, S. Bengio, S. Kim, G. Lebanon, Y. Singer, Local collaborative ranking, in: Proceedings of the 23rd International Conference on World Wide Web, in: WWW '14, ACM, Seoul, Korea, 2014, pp. 85–96, doi:10.1145/2566486.2567970.
[19] J. Lee, D. Lee, Y.-C. Lee, W.-S. Hwang, S.-W. Kim, Improving the accuracy of top-n recommendation using a preference model, Inf. Sci. 348 (2016) 290–304. http://dx.doi.org/10.1016/j.ins.2016.02.005.
[20] W.-P. Lee, K.-H. Lee, Making smartphone service recommendations by predicting users intentions: a context-aware approach, Inf. Sci. 277 (2014) 21–35. http://dx.doi.org/10.1016/j.ins.2014.04.033.
[21] B. Lika, K. Kolomvatsos, S. Hadjiefthymiades, Facing the cold start problem in recommender systems, Expert Syst. Appl. 41 (4, Part 2) (2014) 2065–2073. http://dx.doi.org/10.1016/j.eswa.2013.09.005.
[22] C. Lin, R. Xie, X. Guan, L. Li, T. Li, Personalized news recommendation via implicit social experts, Inf. Sci. 254 (2014) 1–18. http://dx.doi.org/10.1016/j.ins.2013.08.034.
[23] N.N. Liu, Q. Yang, Eigenrank: a ranking-oriented approach to collaborative filtering, in: Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, in: SIGIR '08, ACM, New York, NY, USA, 2008, pp. 83–90, doi:10.1145/1390334.1390351.
[24] X. Ning, G. Karypis, Slim: sparse linear methods for top-n recommender systems, in: Proceedings of the 11th IEEE International Conference on Data Mining, Vancouver, Canada, in: ICDM '11, 2011, pp. 497–506, doi:10.1109/ICDM.2011.134.
[25] X. Ning, G. Karypis, Sparse linear methods with side information for top-n recommendations, in: Proceedings of the Sixth ACM Conference on Recommender Systems, in: RecSys '12, ACM, Dublin, Ireland, 2012, pp. 155–162, doi:10.1145/2365952.2365983.
[26] R. Pan, M. Scholz, Mind the gaps: weighting the unknown in large-scale one-class collaborative filtering, in: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, in: KDD '09, ACM, Paris, France, 2009, pp. 667–676, doi:10.1145/1557019.1557094.
[27] R. Pan, Y. Zhou, B. Cao, N.N. Liu, R. Lukose, M. Scholz, Q. Yang, One-class collaborative filtering, in: Proceedings of the 8th IEEE International Conference on Data Mining, Pisa, Italy, in: 15–19 ICDM '08, 2008, pp. 502–511, doi:10.1109/ICDM.2008.16.
[28] S.-T. Park, W. Chu, Pairwise preference regression for cold-start recommendation, in: Proceedings of the Third ACM Conference on Recommender Systems, in: RecSys '09, ACM, New York, NY, USA, 2009, pp. 21–28, doi:10.1145/1639714.1639720.
[29] S. Rendle, Factorization machines with libfm, ACM Trans. Intell. Syst. Technol. 3 (3) (2012) 57:1–57:22, doi:10.1145/2168752.2168771.
[30] S. Rendle, C. Freudenthaler, Improving pairwise learning for item recommendation from implicit feedback, in: Proceedings of the 7th ACM International Conference on Web Search and Data Mining, in: WSDM '14, ACM, New York, NY, USA, 2014, pp. 273–282, doi:10.1145/2556195.2556248.
[31] S. Rendle, C. Freudenthaler, Z. Gantner, L. Schmidt-Thieme, Bpr: Bayesian personalized ranking from implicit feedback, in: Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, in: UAI '09, AUAI Press, Arlington, Virginia, United States, Montreal, Quebec, Canada, 2009, pp. 452–461.
[32] S. Rendle, L. Schmidt-Thieme, Pairwise interaction tensor factorization for personalized tag recommendation, in: Proceedings of the Third ACM International Conference on Web Search and Data Mining, in: WSDM '10, ACM, NY, USA, New York City, USA, 2010, pp. 81–90, doi:10.1145/1718487.1718498.
[33] F. Ricci, L. Rokach, B. Shapira, P.B. Kantor, Recommender Systems Handbook, 1, Springer US, USA, 2011, doi:10.1007/978-0-387-85820-3.
[34] A. Said, A. Bellogín, Comparative recommender system evaluation: benchmarking recommendation frameworks, in: Proceedings of the 8th ACM Conference on Recommender Systems, in: RecSys '14, ACM, Foster City, Silicon Valley, California, USA, 2014, pp. 129–136, doi:10.1145/2645710.2645746.
[35] Y. Shi, A. Karatzoglou, L. Baltrunas, M. Larson, A. Hanjalic, N. Oliver, Tfmap: optimizing map for top-n context-aware recommendation, in: Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval, in: SIGIR '12, ACM, Portland, Oregon, USA, 2012a, pp. 155–164, doi:10.1145/2348283.2348308.
[36] Y. Shi, A. Karatzoglou, L. Baltrunas, M. Larson, N. Oliver, A. Hanjalic, Climf: learning to maximize reciprocal rank with collaborative less-is-more filtering, in: Proceedings of the Sixth ACM Conference on Recommender Systems, in: RecSys '12, ACM, Dublin, Ireland, 2012b, pp. 139–146, doi:10.1145/2365952.2365981.
[37] Y. Shi, M. Larson, A. Hanjalic, Unifying rating-oriented and ranking-oriented collaborative filtering for improved recommendation, Inf. Sci. 229 (2013) 29–39. http://dx.doi.org/10.1016/j.ins.2012.12.002.
[38] E. Spertus, M. Sahami, O. Buyukkokten, Evaluating similarity measures: a large-scale study in the Orkut social network, in: Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining, in: KDD '05, ACM, Chicago, Illinois, USA, 2005, pp. 678–684, doi:10.1145/1081870.1081956.
[39] P. Symeonidis, D. Ntempos, Y. Manolopoulos, Recommender Systems for Location-based Social Networks, Springer New York, New York, NY, pp. 7–20. doi:10.1007/978-1-4939-0286-6_2.
[40] G. Takács, D. Tikk, Alternating least squares for personalized ranking, in: Proceedings of the Sixth ACM Conference on Recommender Systems, in: RecSys '12, ACM, Dublin, Ireland, 2012, pp. 83–90, doi:10.1145/2365952.2365972.
[41] M.P. Wand, M.C. Jones, Kernel Smoothing, Chapman and Hall/CRC, 1995.
[42] R. Wang, O.R. Zaïane, Discovering process in curriculum data to provide recommendation, in: Proceedings of the 8th International Conference on Educational Data Mining, Madrid, Spain, in: EDM '15, 2015, pp. 580–581.
[43] M. Weimer, A. Karatzoglou, Q.V. Le, A.J. Smola, Cofi rank - maximum margin matrix factorization for collaborative ranking, in: J.C. Platt, D. Koller, Y. Singer, S.T. Roweis (Eds.), Advances in Neural Information Processing Systems 20, Curran Associates, Inc., 2008, pp. 1593–1600.
[44] J.-D. Zhang, C.-Y. Chow, Core: exploiting the personalized influence of two-dimensional geographic coordinates for location recommendations, Inf. Sci. 293 (2015) 163–181. http://dx.doi.org/10.1016/j.ins.2014.09.014.
[45] L. Zheng, F. Zhu, Preference integration in context-aware recommendation, in: Database Systems for Advanced Applications: 22nd International Conference, DASFAA 2017, March 27–30, Part I, Springer International Publishing, Suzhou, China, 2017, pp. 475–489, doi:10.1007/978-3-319-55753-3_30.
[46] L. Zheng, F. Zhu, A. Mohammed, Attribute and global boosting: a rating prediction method in context-aware recommendation, Comput. J. (2017), doi:10.1093/comjnl/bxw016.
[47] X. Zhou, S. Wu, C. Chen, G. Chen, S. Ying, Real-time recommendation for microblogs, Inf. Sci. 279 (2014) 301–325. http://dx.doi.org/10.1016/j.ins.2014.03.121.
[48] Z.-H. Zhou, Ensemble Methods: Foundations and Algorithms, 1st, Chapman & Hall/CRC, 2012.