Knowledge-Based Systems 98 (2016) 68–75

Contents lists available at ScienceDirect

Knowledge-Based Systems

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

User identification for enhancing IP-TV recommendation

Zhijin Wang, Liang He*

Shanghai Key Laboratory of Multidimensional Information Processing, Department of Computer Science and Technology, East China Normal University, Shanghai 200241, China

ARTICLE INFO

Article history:
Received 9 May 2015
Revised 10 December 2015
Accepted 14 January 2016
Available online 10 February 2016

Keywords:
User identification
Shared account
IP-TV recommendation

ABSTRACT

Internet Protocol Television (IP-TV) recommendation systems are designed to provide programs for groups of people, such as a family or a dormitory. Previous methods mainly generate recommendations to a group of people via clustering the common interests of this group. However, these methods often ignore the diversity of a group's interests, and recommendations to a group of people may not match the interests of any of the group members. In this paper, we propose an algorithm that first identifies users in accounts, then provides recommendations for each user. In the identification process, time slots in each account are determined by clustering the factorized time subspace, and similar activities among these slots are combined to represent members. Experimental results show that the proposed algorithm gives substantially better results than previous approaches.

1. Introduction

In recent years, due to the rapid growth and increasing popularity of Internet Protocol Television (IP-TV) services, IP-TV services have been widely consumed in our daily life, and it has been a common phenomenon that families or roommates share programs after they get back home or dormitory from work. To help users (viewers) benefit from the abundant resources, such as channels, programs and videos, and easily find what they are actually interested in, recommender systems [1] are integrated into the services.

However, the main challenge in developing a IP-TV recommendation system is user identification [2]. Intuitively, the recommendations provided to a shared account, comprising the ratings of two dissimilar users, may not match the interests of either of both users [3]. The use of a single account shared by multiple users poses more personalized requirements in providing programs to this account.

Our goal is to improve the recommendation performance by alleviating the problem of user identification in IP-TV services. However, none of the individual user information [4] can be directly used for the identification, since the interaction between a user and a set-top-box (STB) is very weak. Typically, users do not have easy access to the keyboard, mouse or touch screen. Moreover, the services are indistinctly shared by the users in a shared account. The history logs recorded by STBs contain the following data fields: *AccountId, ProgramId, StartTime, EndTime* and *Genre*. In reality, a log recalls that an account starts and ends a program, and marks a program to a genre.

The assumption is that users within a shared account not only have distinct temporal habits, such as after dinner or at weekends, but also have different preferences for television programs (or genres). There are two questions: (1) how to accurately detect temporal habits over accounts? (2) how to accurately obtain preferences based on the detected temporal habits for a user?

To address these questions, we propose a novel algorithm that consists of a partition process and a consolidation process. In the partition process, the time is divided into several nonoverlapping time slots to present temporal habits of users. More specifically, we use the consumption logs to construct an account-item-time play count tensor. We decompose this tensor into the multiplication of a few (low-rank) latent matrices of accounts, items, time intervals, and a core tensor. And then, time slots are obtained via clustering the latent matrix of time intervals. In the consolidation process, we introduce virtual user to represent preferences of an account in a clustered time slot, and similar virtual users are combined to extract users.

A simple overview of our proposed algorithm is drawn in Fig. 1. The contributions of this paper are summarized as follows:

1. We study the problem of user identification in IP-TV services as mining groups of time slots and preferences within accounts.
2. We propose an algorithm to fulfill the identification task. In this algorithm, we try a tensor factorization based subspace clustering method to discover groups of time slots. And then

* Corresponding author. Tel.: (+86)021-54345153.
E-mail addresses: zhijin@ecnu.cn (Z. Wang), lhe@cs.ecnu.edu.cn, zhijin@ecnu.cn (L. He).

http://dx.doi.org/10.1016/j.knosys.2016.01.018
0950-7051/

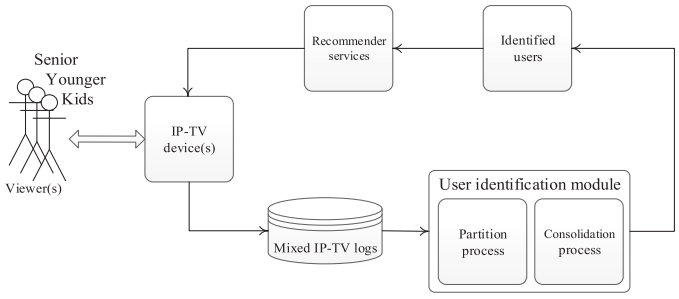


Fig. 1. An overview of user identification for IP-TV recommendation.

similar preferences over these time slots are combined to present users.

3. Finally, we demonstrate how this algorithm above can be applied to improve recommendation. Experimental results on a real IP-TV dataset show that our algorithm outperforms comparable methods.

A preliminary result was reported previously [5]. This paper substantially extends this work.

The rest of this paper is organized as follows. Section 2 provides a brief review of related work on IP-TV recommendation. Section 3 gives the problem definition and notations. Section 4 describes our proposed approach to carry out identification task. Section 5 shows the settings in our experiments. Section 6 presents the experimental results and analysis. Finally, Section 7 concludes the paper.

2. Related work

The work in this paper closely relates to the research areas: collaborative filtering and context-aware recommendation. We present the most relevant previous work in each of them.

2.1. Collaborative filtering

Collaborative filtering (CF) has been the most popular technique for recommender systems [6,7]. Typically, collaborative filtering approaches can be divided into two types: memory-based methods as well as model-based approaches. Memory-based methods focus on using predefined similarity calculation functions to find similar users or items for generating predictions [8]. Memory-based methods can be further classified as user-based [9–11] and item-based [12–14] approaches based on whether similar users or similar items are used. In contrast, the model-based approaches use the observed ratings to train a predefined learning model, and the unobserved ratings are then predicted via the trained model. Algorithms in this category include but not limited to clustering model [15], the aspect models [16], the Bayesian hierarchical model [17], the ranking model [18], etc.

Recently, a particular group of methods, referred to as matrix factorization methods, have become dominant in the field [19]. The performance of the group of methods for the rating prediction problem has been tested in Netflix Prize Contest [20,21] and the KDD CUP 2011 [22]. Matrix factorization methods normally seek to factorize the user-item rating matrix into two low rank latent matrices of users and items, and then utilize the factorized matrices to make further predictions. The factorized latent matrix of user is also employed to cluster groups of users with similar tastes. Zhang et al. [3] believed that users within a household have similar interests, and applied several clustering algorithms to identify groups of users as households. Matrix factorization methods have been further extended to incorporate content metadata information [23], so that the rich side information of users and items beyond the user-item relations can be exploited for improving recommendation. In addition, the matrix factorization framework has also been developed for the top-K recommendation problem in domains with implicit feedback data [24,25], and the binary rating technique is employed to represent users' implicit preferences.

However, these methods cannot be directly used to alleviate our problem, since a television is indistinctly shared by multiple users. In this paper, we consider users within a shared account do not have common interests and temporal habits in IP-TV services.

Table 1
Notations and semantics.

Notation	Semantic
A, I	set of accounts, set of items
U	set of identified users
$U(a)$	set of identified users within account a
u_{ah}	the hth identified user within account a
P	period (a.k.a, set of sub-period)
p_k	the kth sub-period within P
v_{ak}	virtual user within account a in p_k
s_{ah}	several sub-period consumed by u_{ah}
$I(v_{ak})$	set of items consumed by account a in p_k
$\mathcal{C}$	account-item-time play count tensor
T	time dimension of $\mathcal{C}$ (a.k.a, time interval set)
$\mathcal{M}$	factorized core tensor
X, Y, Z	latent matrix of accounts, latent matrix of items, latent matrix of time intervals
c_{ait}	play count of account a to item i in time interval t
$\hat{c}_{ait}$	predicted play count of account a to item i in time interval t
d_{aik}	duration of account a consumes item i in p_k
r_{aik}	implicit rating of account a to item i in p_k
G	preference similarity graph of virtual users
$S_{vv'}$	preference similarity between virtual user v and v'
ρ	threshold of preference similarity
$\|\cdot\|$	length of a set

2.2. Context-aware recommendation systems

The context-aware recommender systems (CARS) have received lots of attentions. Early work in CARS utilized contextual information (e.g., demography, location and time) for pre-processing, where the context drives data selection, or post-processing, where the context is used to filter recommendations [26]. Said et al. [27] used time to split an user into two contextual user profiles, and recommendations are provided to each contextual user profiles. A significant portion of recent work has focused on incorporating context variables into the matrix factorization methods [28–30]. Due to the success of matrix factorization for modeling the user-item relations, one major group of approaches exploited the tensor factorization techniques [31] for modeling the 3-way user-item-context relations [32–34]. A tensor in this case is a generalization of matrix from 2-dimension to n-dimension. Another contribution for modeling the contextual information is factorization machines [35,36], which models the interactions between each pair of entities in terms of their latent factors, such as user–user, user-item, user-context interactions.

The work in this paper also builds upon tensor factorization models. The latent matrix of the context (time) is factorized and clustered to detect temporal habits.

3. Problem definition and notations

Table 1 gives the main notations used in this paper.

To start with, let us consider a common scene that multiple users share a common account in IP-TV services. As Fig. 2 shows,

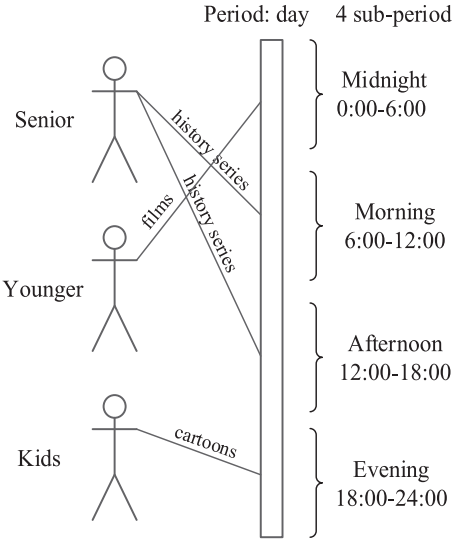


Fig. 2. An example of users sharing an account.

an account corresponding to a STB shared by 3 kinds of family members: senior, younger and kids. The senior get used to demanding history series in the morning or afternoon, kids would like to play the sort of cartoon programs after school or dinner, and younger might prefer films after kids go to the bed.

From the common scene above we can find: (1) the consuming behaviors are periodic; (2) different users get used to consuming the services in different part(s) of a period; (3) different users often have different preferences for programs (genres) provided by the services. Based on this common phenomenon, the problem of identifying users sharing a common STB can be regarded as distinguishing user preferences over time.

We introduce period P to describe periodic behavior. The period is defined as

$$P := \bigcup_{k=1}^{|P|} p_k \quad s.t. \quad \bigcap_{k=1}^{|P|} p_k = \emptyset. \tag{1}$$

This definition means that the continuous period P (e.g., day) consists of several, non-overlapping sub-period p_k (e.g., hour of day). A user may consume television services in more than one sub-period, and is defined as

$$u_{ah} := \{(a, s_{ah}) | a \in A, s_{ah} \subseteq P, s_{ah} \neq \emptyset\}, \tag{2}$$

where u_{ah} is the hth user within account a who consumes services in s_{ah}, s_{ah} denotes several sub-period within P (e.g., Morning and Afternoon), and A denotes all accounts in the system. As Fig. 2 provides, the senior consume services in the Morning and Afternoon, the younger consume services in the Morning, and kids in the Evening. Hence, all users U in system can be defined as

$$U := \bigcup_{a \in A} \bigcup_{s_{ah} \subseteq P} u_{ah}. \tag{3}$$

Our goal is to identify U, and provide accurate recommendations for U. We are trying to reach the goal by addressing the two questions: (1) how to capture the preferences of user u_{ah} (hth user within account a)? (2) how to determine the consuming time s_{ah} of hth user within account a?

In order to capture the preferences of user u_{ah}, we introduce virtual user v_{ak} to present activities of an account in a sub-period

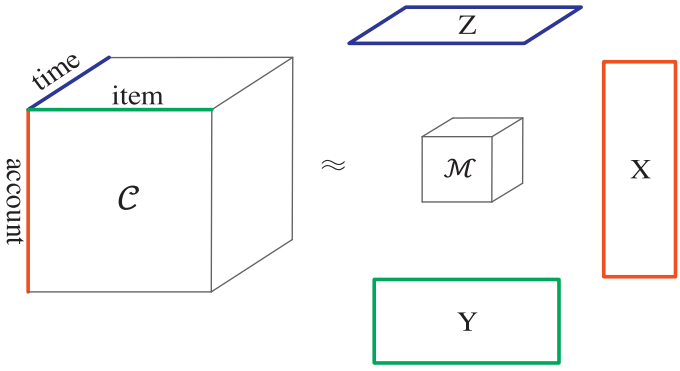


Fig. 3. Structure of the play count tensor.

p_k. The virtual user is defined as

$$v_{ak} := \{(a, p_k) | a \in A, p_k \in P, p_k \neq \emptyset\}, \tag{4}$$

where v_{ak} is the activities of account a in sub-period p_k, and an identified user is a composite of virtual user(s). Therefore, the preferences of a real user can be composed by the preferences of corresponding virtual users.

We suppose that users in reality have different preferences for both programs (or genres) and different periodic behaviors. Hence, the users in an account can be identified by consolidation of virtual users. We introduce the similarity graph G, which uses vertexes to denote virtual users, and uses edges to denote the similarity between virtual users. The preference similarity graph G is defined as

$$G := G(v(a), s(a)), \tag{5}$$

where $v(a)$ is the set of virtual users within account a, and $s(a)$ presents pairs of similar virtual users within account a. The benefits of using graph G are that the process of combining virtual users can be easily controlled and the complexity of this process can be reduced.

4. Algorithm for user identification

In this section, we propose an algorithm, namely Tensor factorization based subspace clustering and preferences consolidation (TCC), to identify users within shared accounts. The algorithm mainly consists of the partition process for determining time slots, and the consolidation process for fusing similar preferences in different time slots.

Detailed steps of the proposed TCC algorithm are given in Algorithm 1.

4.1. Partition process

We try three different methods to carry out the partition task: (1) empirical split method. The time slots are assigned according to experiments; (2) average split method. In this method, the period is equally divided into a given number of time slots. We use VUI to present this method as well as [5]; (3) while previous methods are greatly affected by manual setup and cannot fit the dataset well. In this paper, we use latent time space to cluster the time slots via factorizing the account-item-time play count tensor.

As Fig. 3 shows, we use symbol $\mathcal{C}$ to present the play count tensor, use c_{ait} to denote a given entry in $\mathcal{C}$, and use T to denote time dimension of $\mathcal{C}$. More realistically, c_{ait} means the play count of item i consumed by account a in time interval $t \in T$. The time dimension T in $\mathcal{C}$ is equally divided into several intervals, and each interval in $\mathcal{C}$ must be smaller than any sub-period p_k. Therefore,

Algorithm 1: Pseudo code of TCC to identify users within shared accounts.

Input: Television history log L

1 $\mathcal{C} \leftarrow L$; /* Construct the play count tensor */
2 $P \leftarrow$ **Partition**$(\mathcal{C})$;
3 **for** *each account a in A* **do**
4 $\quad U \leftarrow U +$ **Consolidation**(P, a);
5 **return** U;

Procedure Partition$(\mathcal{C})$

6 Set d_X, d_Y, d_Z, initialize $\mathcal{M} \in \mathbb{R}^{d_X \times d_Y \times d_Z}, X \in \mathbb{R}^{|U| \times d_X}$, $Y \in \mathbb{R}^{|I| \times d_Y}$, $Z \in \mathbb{R}^{|T| \times d_Z}$ with random values in $[0, 0.1]$;
7 **while** $t <$ *MaxEpoch* **do**
8 $\quad$ **for** *non-zero entry* c_{ait} *in* $\mathcal{C}$ **do**
9 $\quad\quad e_{ait} \leftarrow c_{ait} - \mathcal{M} \times_X X_{a*} \times_Y Y_{i*} \times_Z Z_{t*}$;
10 $\quad\quad X_{a*} \leftarrow X_{a*} + \gamma (e_{ait} \cdot \mathcal{M} \times_Y Y_{i*} \times_Z Z_{t*} - \lambda X_{a*})$;
11 $\quad\quad Y_{i*} \leftarrow Y_{i*} + \gamma (e_{ait} \cdot \mathcal{M} \times_X X_{a*} \times_Z Z_{t*} - \lambda Y_{i*})$;
12 $\quad\quad Z_{t*} \leftarrow Z_{t*} + \gamma (e_{ait} \cdot \mathcal{M} \times_Y Y_{i*} \times_Z Z_{t*} - \lambda Z_{t*})$;
13 $\quad t \leftarrow t + 1$;
14 $P \leftarrow$ cluster Z^T by k-Means;
15 **return** P;

Procedure Consolidation(P, a)

16 $V \leftarrow$ calculate implicit rating by P using Eq. 7;
17 $G \leftarrow$ generate similarity graph by V, ρ using Eq. 8;
18 Set $h = 1$; /* The first identified user */
19 **for** *each vertex v in G.vertexes* **do** /* DFS in G */
20 $\quad$ **if** *v.visited* == *false* **then**
21 $\quad\quad u_{ah}.add(v)$; // Extract v as hth user;
22 $\quad\quad v.visited = true$;
23 $\quad\quad$ Get list of vertexes that connect to v as L_v;
24 $\quad\quad$ Visit each vertex in L_v and add to u_{ah} recursively;
25 $\quad\quad h \leftarrow h + 1$;
26 $\quad\quad U(a) \leftarrow U(a) + u_{ah}$;
27 **return** $U(a)$;

the sub-period can be obtained by clustering the factorized sub-spaces.

A common approach to obtaining the sub-space is to decompose $\mathcal{C}$ into the multiplication of a few (low-rank) matrices and a core tensor (or just a few vectors), based on $\mathcal{C}$'s non-zero entries. For example, as illustrated in the right part of Fig. 3, we can decompose $\mathcal{C}$ into the multiplication of a core tensor $\mathcal{M} \in \mathbb{R}^{d_X \times d_Y \times d_Z}$, and three matrices, $X \in \mathbb{R}^{|A| \times d_X}, Y \in \mathbb{R}^{|I| \times d_Y}, Z \in \mathbb{R}^{|T| \times d_Z}$, using a tucker decomposition model [37]. The objective function to control the error of the decomposition is usually defined as

$$L(\mathcal{M}, X, Y, Z) = \frac{1}{2} \|\mathcal{C} - \mathcal{M} \times_X X \times_Y Y \times_Z Z\|^2 + \frac{\lambda}{2} (\|\mathcal{M}\|^2 + \|X\|^2 + \|Y\|^2 + \|Z\|^2), \quad (6)$$

where $\|\cdot\|^2$ denotes the Frobenius norm, the first part is to control the decomposition error and $\frac{\lambda}{2}(\|\mathcal{M}\|^2 + \|X\|^2 + \|Y\|^2 + \|Z\|^2)$ is a regularization term to avoid over-fitting, d_X, d_Y and d_Z are usually very small, denoting the number of latent factors, λ is a parameter controlling the contribution of the regularization term. The symbol "$\times$" denotes the matrix multiplication; "$\times_X$" stands for the tensor-matrix multiplication, where the subscript "X" stands for the mode a tensor, e.g., $W = \mathcal{M} \times_X X$ is $W_{ijk} = \sum_{i=1}^{d_X} \mathcal{M}_{ijk} \times X_{ij}$.

As the partition procedure in Lines 6–15, Algorithm 1 shows, we exploit stochastic gradient descent (SGD)[1] to learn the tensor, where X_{a*}, Y_{i*}, Z_{t*} denote the ath, ith, tth columns of X, Y, Z respectively, $\mathcal{M} \times_Y Y_{i*} \times_Z Z_{t*}$ is the gradient of X_{a*}, and γ is the learning rate. We choose k-Means to cluster the latent time space Z [39], and the similarity between two time intervals is measured by Squared Euclidean Distance.

4.2. Consolidation process

As presented in Consolidation procedure in Lines 16–26, Algorithm 1, members in a shared account are represented via consolidating the similar preferences among the parted time slots. The duration from an account to an item is used to calculate the preferences. The implicit rating of an account over a time slot is formulated as

$$r_{aik} = \frac{\exp(d_{aik})}{\sum_{k=1}^{|P|} \exp(d_{aik})}, \quad (7)$$

where d_{aik} is the duration of item i consumed by account a in sub-period p_k, and $\exp()$ stands for the exponential function [40]. The $\exp()$ is adopted to smooth d_{aik}, since d_{aik} varies as item changes. For example, the length of a played movie can easily exceed one hour, but the length of a cartoon is usually limited within half of an hour. Note that, an account may demand an item more than one time, therefore, we do not choose the binary rating techniques or percentage of a program watched to the length of it [7]. For convenience, the virtual user (Eq. (4)) is introduced to represent an account's activities in a given time slot, and the implicit rating of a given virtual user is captured by using Eq. (7).

In order to consolidate the similar virtual users within an account, we use Cosine function to measure the similarity between two virtual users, then a similarity threshold $\rho \in [0, 1]$ is adopted to determine the consolidation. Given account a, the preference similarity between virtual user v and v' within a is defined as

$$S_{vv'} = \cos(v, v') = \cos(v_{ak}, v_{ak'}) = \frac{\sum_{i \in I(v_{ak}) \cap I(v_{ak'})} r_{aik} \cdot r_{aik'}}{\sqrt{\sum_{i \in I(v_{ak})} r_{aik} \cdot \sum_{i \in I(v_{ak'})} r_{aik'}}}, \quad (8)$$

where r_{aik} denotes implicit rating of account a to item i in p_k, and $I(v_{ak})$ denotes set of items consumed by virtual user v_{ak}. We introduce $s'_{vv'} \in \{0, 1\}$ to denote binary similarity. The binary similarity $s'_{vv'}$ is defined as

$$s'_{vv'} = \begin{cases} 1, & S_{vv'} \geqslant \rho \\ 0, & \text{otherwise} \end{cases} \quad (9)$$

where ρ denotes the threshold of preference similarity. This means if the similarity of two virtual users is greater than ρ, the virtual users are regarded as similar, otherwise dissimilar. The advantage of ρ is that the effects on recommendation performance of the consolidation process can be explicitly observed.

To combine the similar virtual users throughout an account, we regard a virtual user as a node in a graph and the similarity between virtual users as edges of this graph. Therefore, the consolidation process is a problem of graph traverse. The consolidation operation is described in Lines 18–27 in Algorithm 1 using deep-first-search (DFS). An alternative way to carry out the task is bread-first-search (BFS) algorithm.

The complexity of the proposed algorithm TCC is $\mathcal{O}(t_p|\mathcal{C}|d_X d_Y d_Z + t_c|P||T|d_Z + |A|(|P|^2/2 + 2|P| + |S|))$, and we

[1] The SGD [31] is chosen due to its speed and ease of implementation. An alternative strategy is alternating least square (ALS) [38]. While ALS can be parallelized, these advantages are irrelevant in our case.

Table 2
Fields of the provided television history logs.

Fields	Semantics
AccountId	Account's ID value
ProgramId	Program's ID value
StartTime	Watching start time
EndTime	Watching end time
Genre	Program genre

summarize it as follows. Given the max epoch of tensor factorization t_p and the max epoch of k-Means t_c, the tensor factorization scales linearly to the number of non-zero entries $|\mathcal{C}|$ and the dimensionality of the factors d_X, d_Y, d_Z, the complexity of tensor factorization is $\mathcal{O}(|\mathcal{C}|d_X d_Y d_Z)$. The complexity of k-Means is $\mathcal{O}(t_c|P||T|d_Z)$, $|T|$ denotes the number of time intervals of time dimension in $\mathcal{C}$. In the consolidation process, given the number of pairs of similar virtual users (i.e., edges) $|S|$, the complexity of implicit rating is $\mathcal{O}(|P|)$, and it takes $\mathcal{O}(|P|^2 + |P| + |S|)$ to generate and traverse the graph. From these complexity analysis, we can see that the TCC complexity mainly depends on the number of split sub-period $|P|$.

5. Experimental setup

In this section, we illustrate dataset collection, evaluation metrics and algorithms for recommendation. We evaluate algorithm TCC on the dataset collected from the content provider SMG[2] in Shanghai, China. It should be noted that we focus on the evaluation of recommender performance by means of identified users, rather than the accuracy evaluation of algorithm TCC.

5.1. Dataset collection

The logs in the services from SMG are during the period between March 1, 2011 till March 31, 2011. We filter out logs of play time (calculated by start time and end time) less than 10 min. It contains 376,038 records, 5933 videos categorized into 66 genres consumed by 14,856 accounts after being filtered. The records before March 25, 2011 are used for training, and the rest are as test set. The fields of this dataset are shown in Table 2.

In order to avoid problems related to cold start, for both accounts and items, we decide that accounts in the evaluation sets have to consume at least 50 programs. Three subsets of 100, 200, and 500 accounts are randomly selected to carry out the experiments.

5.2. Evaluation metrics

We use precision and recall metrics to measure the performance of all the mentioned algorithm, since they often attract lots of attention in a running system and are well known. The precision metric is defined as

$$Precision@N = \frac{\sum_{u\in U} |R(u,N) \cap T(u)|}{\sum_{u\in U} |T(u)|}, \tag{10}$$

where N denotes the length of a recommendation list, $R(u, N)$ denotes the recommendation list to user u with length N, $T(u)$ means items has been consumed by identified user u in test set. The recall metric is defined as

$$Recall@N = \frac{\sum_{u\in U} |R(u,N) \cap T(u)|}{\sum_{u\in U} |R(u,N)|}. \tag{11}$$

[2] http://www.smg.cn/.

From these definitions, we can see that a larger *Precision@N* or *Recall@N* indicates a better performance.

We use the cross-validation technique to study the parameter period P, similarity threshold ρ, and the number of clustered time slots in Algorithm 1 as well. According to validated experiments, we first hold $|P| = 4$, and observe the evaluation metrics by changing ρ from 0 to 1 with step size 0.1, then the performance are measured by ranging $|P|$ in {1, 2, 3, 4, 5, 6, 8, 12} while holding $\rho = 0.8$. For the number of clusters k in TCC, we set $k = 4$ to compare with average split method with $|P| = 4$ and fix $\rho = 0.8$ while changing the recommendation list N.

5.3. Recommendation algorithms

We adopt, one of the most famous collaborative filtering methods, K-Nearest Neighbor (KNN) method to provide recommendations, since it performs very well in practice (e.g., [41–43]), and we can also learn the benefit from identified users by comparing with recommendations without identification. The Cosine method is used to measure the similarity among accounts/users in algorithm KNN. Hereafter, for easy presentation of figures, we name the recommendations for users identified by Algorithm 1 as TCC, and the recommendations for users identified by the average split method in [5] is named as VUI, respectively.

To study the improvement of TCC (or VUI), we choose the following comparable methods.

1. *AccountKNN*: this method is the account-based collaborative filtering. The unknown ratings are predicted by considering the ratings given by similar accounts. Account similarity is computed by cosine similarity of ratings.
2. *CUPs* [27]: this method regards that an account consists of two contextual user profiles (CUPs), and these users are determined by the time of a film started (or ended).
3. *TF* [44]: this method incorporates time as an independent dimension, and the preferences of missing values in the account-item-time tensor are recovered via factorization techniques. The top-N predicted preferred items within time intervals are provided to accounts.

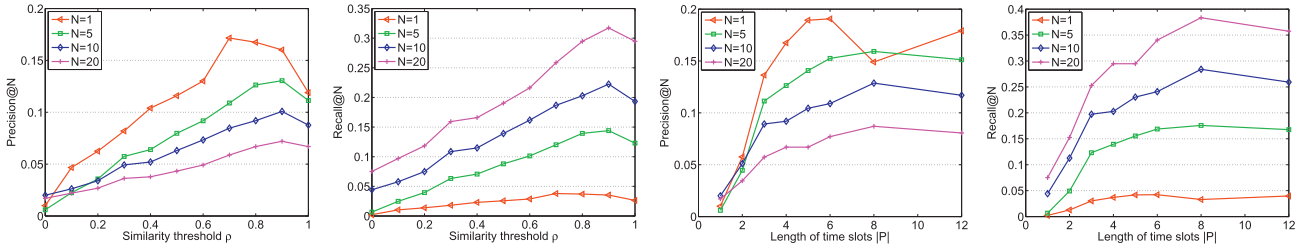
6. Experimental results and analysis

In this section, we conduct several experiments to compare different parameters of VUI and different methods. Our experiments are intended to address the following questions:

1. How the parameters (P and ρ) affect recommendations? In other words, how the partition of a period and consolidation of virtual users affect recommendations?
2. How the split methods affect recommendations? Can the time consuming behavior be obtained by clustering the factorized subspace?
3. Can TCC outperform other comparable methods?
4. Finally, what is the connection between the number of identified users and the recommendation performance?

6.1. Effects on P and ρ

To study how partition process and consolidation process affect user identification and recommendations, we measure the performance in terms of precision and recall as P or ρ change while holding other parameter. Note that, when $|P| = 1$ or $\rho = 0$, the TCC regards an account as a user, and items are directly recommend to an account. In reality, users tend to consume items in specific hours of a day. Here, we use average split method to equally assign the time slots.



(a) *Precision@N* as ρ changes, fix $|P| = 4$ (b) *Recall@N* as ρ changes, fix $|P| = 4$ (c) *Precision@N* as $|P|$ changes, fix $\rho = 0.8$ (d) *Recall@N* as $|P|$ changes, fix $\rho = 0.8$

Fig. 4. The recommendation performance are obtained by validating parameter $|P|$ and ρ.

Table 3
The split up sub-period ($|P| = 4$).

Methods	Midnight	Morning	Afternoon	Evening
Average	0:00–6:00	6:00–12:00	12:00–18:00	18:00–24:00
Empirical	**23:50**–6:00	6:00–12:00	12:00–**19:00**	**19:00–23:50**

Table 4
The sub-period clustered by k-Means (k=4).

Subspace	Cluster1	Cluster2	Cluster3	Cluster4
Z^T	{0,3,5}	{16,18,19}	{11,12,15,17,20}	other slots

The results are presented in Fig. 4. As Fig. 4(a) and (b) state, we hold $|P| = 4$ and change ρ from 0 to 1 with step size 0.1, (1) the precision and recall values are still increasing as ρ increases, (2) the optimal values are found when ρ is set to 0.8, and (3) the *Precision*@10, *Precision*@20 value are both higher than the *Precision*@1, *Precision*@5 value when no users are identified ($\rho = 0$), a possible reason is that less recommendations to an account have a higher chance of mismatch of members' interest.

To study effects on the split of period P, we fix ρ at 0.8 and change $|P|$ to measure *Precision@N* and *Recall@N* when making N genre(s) recommendation with $N = \{1, 5, 10, 20\}$. Here, we set $\rho = 0.8$ since the optimal precision and recall values are found at ($|P| = 4, \rho = 0.8$) according to results in Fig. 4(a) and (b). As revealed in Fig. 4(c) and (d), (1) the precision and recall values are significantly improved, when compared with AccountKNN ($|P| = 1$, Precision: 1%, Recall: $< 1\%$); (2) the two optimal values are obtained at $|P| = 4$ and $|P| = 5$, and the precision value is slight over 17%; (3) the performance starts degrading when $|P| > 5$, and (4) the *Precision*@1 value is smaller than *Precision*@5 when $|P| = 8$, which reveals miss match of interests for shared accounts.

6.2. Effects on partition methods

According to the conducted experiments above, we learn the effects on the parameters (P and ρ) with respect to average split methods, and the two optimal values can be found at $|P| = 4$ and $|P| = 5$ (a slight better). In this section, we compare Algorithm 1 with empirical and average split methods at [5].

We set $|P| = 4$ to compare the split methods, since it's more easier to empirically split up four sub-period than five sub-period. Another reason we choose four sub-period to compare is that four sub-period are more close to our daily life. The split up sub-period are shown in Table 3, and the differences are in bold type.

We use the partition procedure in TCC (see Algorithm 1) to cluster 4 time slots. The play count tensor are learned until convergency by the configurations as follows: $d_x = d_y = d_z = 5$, $\gamma = \lambda = 0.01$ [34]. Table 4 shows the results via clustering factorized time space Z^T. According to the status of a television, a possible status of televisions in intervals of cluster4 in Table 4 is under suspension. In contrast, the empirical method and the average split method both regard televisions work all the day, which may lead to a slip on user identification.

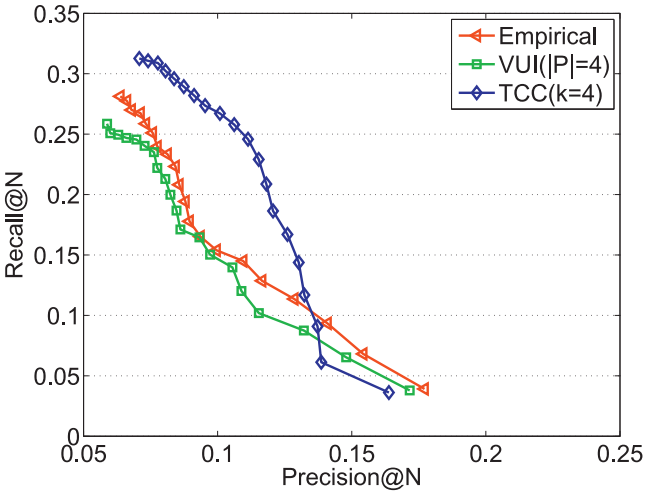


Fig. 5. Comparison of partition methods. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

We compare the results of 3 partition mentioned methods on top-N recommendation as the length of recommendation list N changes. For these 3 methods, ρ is set to 0.8. As Fig. 5 states, (1) the empirical split method gains a slight improvement when compared with the average split method, but the improvement is not stable. A possible reason for the improvement is that users are off work after 18:00, and they need to spend time on the way and cannot receive programs immediately; (2) the TCC (blue line with diamond sign) outperforms other two split methods with respect to *Precision@N* and *Recall@N* when $N > 3$. The benefit of VUI is its simpleness, meanwhile VUI greatly depends on manual setup. The TCC avoids this problem, and it can automatically cluster the sub-period.

6.3. Method comparison

The predictive accuracy with respect to *Precision@N* and *Recall@N* (N ranges from 1 to 20) is measured and plotted in Fig. 6.

Firstly, the predictive accuracy is greatly enhanced when an account is decomposed into several users, and the best recommendation performance is achieved by TCC. We compare TF and TCC ($\rho = 0.8$). It can be observed that the recommendation performances of both two methods tend to be close as N increases. However, the precision values of TF are still smaller than TCC's.

Secondly, the predictive accuracy of AccountKNN (black line with cross sign) is instable and worse than any other comparable

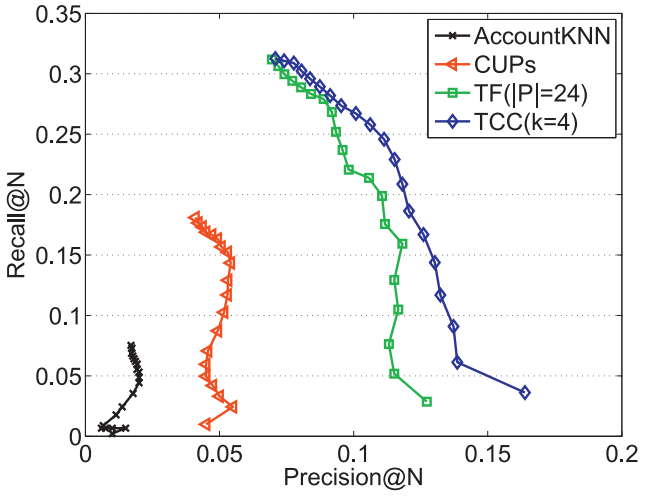
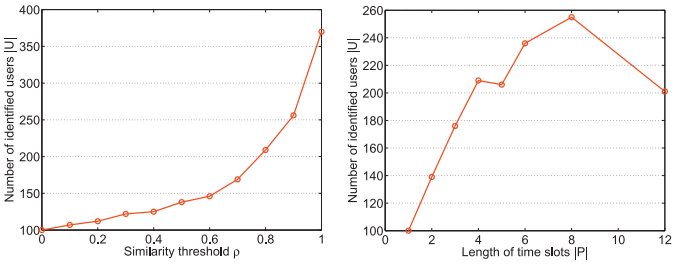


Fig. 6. Comparison of comparable methods.



(a) |U| versus ρ, fix |P| = 4 (b) |U| versus |P|, fix ρ = 0.8

Fig. 7. Number of identified users $|U|$ as parameters change.

methods. This provides an evidence for the essential of user identification. A possible reason is that the recommendations provided to an account do not match the interests of either of its members, and the preferences of an account consisting of several members.

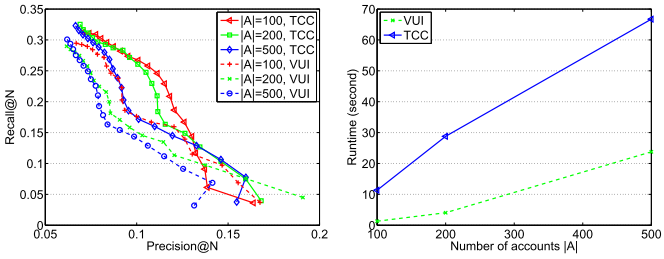
6.4. Identification and performance analysis

We try to discover the relationship between the number of identified users and the recommender performance. The numbers of identified users are plotted in Fig. 7, and the performance in terms of *Precision@N* and *Recall@N* is measured in Fig. 4.

According to the comparison of Figs. 4(a), (b) and 7(a), we can see that (1) the *Precision@N* and *Recall@N* values are increasing when users are identified; (2) the two optimal performances are found at ($|U| = 209, \rho = 0.8, |P| = 4$) and ($|U| = 256, \rho = 0.9, |P| = 4$); (3) the precision starts decreasing when $|U| > 256$. The comparison among Figs. 4(c), (d) and 7(b), states, (1) the optimal performance is found at ($|U| = 255, |P| = 5, \rho = 0.8$), and (2) the recommendation performance degrades when $|P| > 5$, $\rho = 0.8$.

The following summarizes the key conclusions we observe from the results: (1) the recommender performance is improved when users within accounts are identified for personalized recommendations. A reason for the improvement is that recommendations based on identified users alleviate the problem of recommending given items to wrong users within a shared account; (2) too many or too few identified users ($|U|$ is too small or too large) will degrade the performance. The possible reasons are as follows:

1. Too few users identified: when P is split into few sub-period and ρ is set very close to 1, which may regard two (or more than two) real users as one. Hence, a possible reason for the low performance is that items are recommended to users who dislike them.
2. Too many users identified: when P is split into too many sub-period and ρ is set very close to 0, which may regard a real user as two (or more than two) identified users, and the preferences of the real user are divided into several parts by the identified users. Hence, the opportunity of recommending right items to the real user may decrease since the KNN recommends items other users also preferred.



(a) The recommendation accuracy of VUI (b) A runtime comparison of VUI and TCC and TCC on different data scales (Given $|P|$ = methods on the SMG dataset as the number 4, $\rho = 0.8$). of accounts changes.

Fig. 8. The recommendation performance and runtime of TCC and VUI on different data scales.

(3) the optimal precision and recall values are found when 2.5 users per account are identified, and it reflects the rate of real user number to account number.

6.5. Scalability

To study the scalability of the average split method and the proposed algorithm TCC, we randomly select 3 subsets of 100, 200, and 500 accounts. As shown in Fig. 8(a), given *Recall@N* > 0.05, the TCC (solid lines) outperforms average split method (dash lines) as the data scale increases. For both algorithms, the performance degrades as the data scale increases, however the performance is instable when $N = 1$ (i.e., *Recall@N* < 0.05), which shows that less recommended items get a higher probability of interests mismatch.

Although TCC has a better recommendation accuracy, the runtime of TCC is still longer than VUI's. Fig. 8(b) reports the runtime comparison of TCC and VUI on SMG dataset. We can see that TCC spends much time in clustering the time slots. The rate of TCC runtime to VUI runtime tends to be smaller as the data scale increases. The TCC scales linearly to the number of accounts by benefiting from SGD.

7. Conclusions and future work

In this paper, we study the problem that multiple users share a common account in IP-TV services. To enhance recommendation performance for each user, we propose the algorithm TCC to decompose an account into several users. This algorithm consists of two processes: the partition process is designed for detecting time slots, and the consolidation process is used for combining similar preferences to extract real users.

Experimental results on a commercial dataset show that about 2.5 users per account are identified in average, and the recommendation performance is significantly enhanced with respect to precision and recall. The advantage of the proposed TCC is that the temporal habits (i.e., time slots) can be automatically learned from the provided dataset. Moreover, the proposed VUI and TCC have been officially adopted by the provider SMG. The algorithm VUI has been applied as an option of recommendation strategy in the television system with excellent user satisfaction since 2013, and TCC is deploying.

In the future, we plan on extending our work on the study of factorization methods for partition process, cross-validation techniques in terms of the number of clusters and ρ and try different recommendation algorithms.

Acknowledgments

This research has been supported in part by the Shanghai Science and Technology Commission Foundation (nos. 12dz1500205 and 13430710100), and the National High Technology Research and Development Program ("863" Program no. 2015AA015801). The authors are grateful to the editor and anonymous reviewers for their helpful comments in improving the quality of the paper.

References

[1] J. Bobadilla, F. Ortega, A. Hernando, A. Gutiérrez, Recommender systems survey, Knowl. Based Syst. 46 (2013) 109–132.
[2] R. Bambini, P. Cremonesi, R. Turrin, A recommender system for an IP-TV service provider: a real large-scale production environment, in: Recommender Systems Handbook, 2011, pp. 299–331.
[3] A. Zhang, N. Fawaz, S. Ioannidis, A. Montanari, Guess who rated this movie: Identifying users through subspace clustering, in: Proceedings of the Twenty-eighth Conference on Uncertainty in Artificial Intelligence, 2012, pp. 944–953.
[4] S. Seko, T. Yagi, M. Motegi, S.y. Muto, Group recommendation using feature space representing behavioral tendency and power balance among members, in: Proceedings of the Fifth ACM Conference on Recommender Systems, 2011, pp. 101–108.
[5] Z. Wang, Y. Yang, L. He, J. Gu, User identification within a shared account: Improving IP-TV recommender performance, in: Proceedings of the Eighteenth Advances in Databases and Information Systems, 2014, pp. 219–233.
[6] 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.
[7] F. Ricci, L. Rokach, B. Shapira, P.B. Kantor, Recommender Systems Handbook, Springer, 2011.
[8] H. Ma, An experimental study on implicit social recommendation, in: Proceedings of the Thirty-sixth ACM International Conference on Research and Development in Information Retrieval, 2013, pp. 73–82.
[9] H. Liu, Z. Hu, A.U. Mian, H. Tian, X. Zhu, A new user similarity model to improve the accuracy of collaborative filtering, Knowl. Based Syst. 56 (2014) 156–166.
[10] R. Jin, J.Y. Chai, L. Si, An automatic weighting scheme for collaborative filtering, in: Proceedings of the Twenty-seventh ACM International Conference on Research and Development in Information Retrieval, 2004, pp. 337–344.
[11] J.L. Herlocker, J.A. Konstan, A. Borchers, J. Riedl, An algorithmic framework for performing collaborative filtering, in: Proceedings of the Twenty-second ACM International Conference on Research and Development in Information Retrieval, 1999, pp. 230–237.
[12] M. Deshpande, G. Karypis, Item-based top-*N* recommendation algorithms, ACM Trans. Inf. Syst. 22 (1) (2004) 143–177.
[13] G. Linden, B. Smith, J. York, Amazon.com recommendations: Item-to-item collaborative filtering, IEEE Internet Comput. 7 (1) (2003) 76–80.
[14] B.˜M. Sarwar, G. Karypis, J.A. Konstan, J. Riedl, Item-based collaborative filtering recommendation algorithms, in: Proceedings of the Tenth International World Wide Web Conference, 2001, pp. 285–295.
[15] A.K.-B. Merialdo, Clustering for collaborative filtering applications, Intell. Image Process. Data Anal. Inf. Retr. 3 (1999) 199.
[16] T. Hofmann, Collaborative filtering via Gaussian probabilistic latent semantic analysis, in: Proceedings of the Twenty-sixth ACM International Conference on Research and Development in Information Retrieval, 2003, pp. 259–266.
[17] Y. Zhang, J. Koren, Efficient Bayesian hierarchical user modeling for recommendation system, in: Proceedings of the Thirtieth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2007), Amsterdam, The Netherlands, 2007, pp. 47–54.July 23--27, 2007.
[18] N.N. Liu, Q. Yang, Eigenrank: A ranking-oriented approach to collaborative filtering, in: Proceedings of the Thirty-first Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2008), Singapore, 2008, pp. 83–90.July 20--24, 2008.
[19] Y. Koren, R.M. Bell, C. Volinsky, Matrix factorization techniques for recommender systems, IEEE Comput. 42 (8) (2009) 30–37.
[20] Y. Koren, Collaborative filtering with temporal dynamics, in: Proceedings of the Fifteenth ACM International Conference on Knowledge Discovery and Data Mining, 2009, pp. 447–456.
[21] R.M. Bell, Y. Koren, Scalable collaborative filtering with jointly derived neighborhood interpolation weights, in: Proceedings of the Seventh IEEE International Conference on Data Mining, 2007, pp. 43–52.
[22] G. Dror, N. Koenigstein, Y. Koren, M. Weimer, The yahoo! music dataset and kdd-cup'11, in: KDD Cup, 2012, pp. 8–18.
[23] N. Koenigstein, G. Dror, Y. Koren, Yahoo! music recommendations: modeling music ratings with temporal dynamics and item taxonomy, in: Proceedings of the Fifth ACM Conference on Recommender Systems, 2011, pp. 165–172.
[24] Y. Hu, Y. Koren, C. Volinsky, Collaborative filtering for implicit feedback datasets, in: Proceedings of the Eighth IEEE International Conference on Data Mining, 2008, pp. 263–272.
[25] M. Xu, S. Berkovsky, S. Ardon, S. Triukose, A. Mahanti, I. Koprinska, Catch-up TV recommendations: show old favourites and find new ones, in: Proceedings of the Seventh ACM Conference on Recommender Systems, 2013, pp. 285–294.
[26] C. Palmisano, A. Tuzhilin, M. Gorgoglione, Using context to improve predictive modeling of customers in personalization applications, IEEE Trans. Knowl. Data Eng. 20 (11) (2008) 1535–1549.
[27] A. Said, E.W.D. Luca, S. Albayrak, Inferring contextual user profiles – Improving recommender performance, in: Proceedings of the Third Workshop on Context-Aware Recommender Systems, IEEE, 2011.
[28] O. Luaces, J. Díez, A. Alonso-Betanzos, A.T. Lora, A. Bahamonde, A factorization approach to evaluate open-response assignments in MOOCs using preference learning on peer assessments, Knowl. Based Syst. 85 (2015) 322–328.
[29] P. Pirasteh, D. Hwang, J.J. Jung, Exploiting matrix factorization to asymmetric user similarities in recommendation systems, Knowl. Based Syst. 83 (2015) 51–57.
[30] Y. Zheng, B. Mobasher, R.D. Burke, Deviation-based contextual SLIM recommenders, in: Proceedings of the Twenty-third ACM International Conference on Conference on Information and Knowledge Management, 2014, pp. 271–280.
[31] T.G. Kolda, B.W. Bader, Tensor decompositions and applications, SIAM Rev. 51 (3) (2009) 455–500.
[32] Y. Shi, A. Karatzoglou, L. Baltrunas, M. Larson, A. Hanjalic, CARS2: Learning context-aware representations for context-aware recommendations, in: Proceedings of the Twenty-third ACM International Conference on Conference on Information and Knowledge Management, 2014, pp. 291–300.
[33] Y. Xin, H. Steck, Multi-value probabilistic matrix factorization for IP-TV recommendations, in: Proceedings of the Fifth ACM Conference on Recommender Systems, 2011, pp. 221–228.
[34] A. Karatzoglou, X. Amatriain, L. Baltrunas, N. Oliver, Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering, in: Proceedings of the Fourth ACM Conference on Recommender Systems, 2010, pp. 79–86.
[35] S. Rendle, Factorization machines, in: Proceedings of the Tenth IEEE International Conference on Data Mining, 2010, pp. 995–1000.
[36] S. Rendle, Z. Gantner, C. Freudenthaler, L. Schmidt-Thieme, Fast context-aware recommendations with factorization machines, in: Proceedings of the Thirty-fourth ACM International Conference on Research and Development in Information Retrieval, 2011, pp. 635–644.
[37] T.G. Kolda, B.W. Bader, J.P. Kenny, Higher-order web link analysis using multilinear algebra, in: Proceedings of the Fifth IEEE International Conference on Data Mining, 2005, pp. 242–249.
[38] Z. Wang, Y. Yang, Q. Hu, L. He, An empirical study of personal factors and social effects on rating prediction, Proceedings of the Nineteenth Pacific-Asia Conference on Knowledge Discovery and Data Mining, 2015, pp. 747–758.
[39] Y. Ma, A.Y. Yang, H. Derksen, R.M. Fossum, Estimation of subspace arrangements with applications in modeling and segmenting mixed data, SIAM Rev. 50 (3) (2008) 413–458.
[40] N. Koenigstein, Y. Koren, Towards scalable and accurate item-oriented recommendations, in: Proceedings of the Seventh ACM Conference on Recommender Systems, 2013, pp. 419–422.
[41] G. Katz, N. Ofek, B. Shapira, L. Rokach, G. Shani, Using Wikipedia to boost collaborative filtering techniques, in: Proceedings of the Fifth ACM Conference on Recommender Systems, 2011, pp. 285–288.
[42] N.N. Liu, M. Zhao, E.W. Xiang, Q. Yang, Online evolutionary collaborative filtering, in: Proceedings of the Fourth ACM Conference on Recommender Systems, 2010, pp. 95–102.
[43] Y. Zhao, X. Feng, J. Li, B. Liu, Shared collaborative filtering, in: Proceedings of the Fifth ACM Conference on Recommender Systems, 2011, pp. 29–36.
[44] P. Adamopoulos, A. Tuzhilin, Estimating the value of multi-dimensional data sets in context-based recommender systems, in: Poster Proceedings of the Eighth ACM Conference on Recommender Systems (Recsys'14), 2014.