Expert Systems With Applications 126 (2019) 9–19

Contents lists available at ScienceDirect

Expert Systems With Applications

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

Graph-based context-aware collaborative filtering

Tu Minh Phuong[a,1,*], Do Thi Lien[b,2], Nguyen Duy Phuong[a,3]

[a] Department of Computer Science, Posts and Telecommunications Institute of Technology, Km 10 Nguyen Trai rd. Ha Dong, Hanoi, Vietnam
[b] Department of Multimedia Technologies, Posts and Telecommunications Institute of Technology, Km 10 Nguyen Trai rd. Ha Dong, Hanoi, Vietnam

ARTICLE INFO

Article history:
Received 11 September 2018
Revised 9 February 2019
Accepted 10 February 2019
Available online 11 February 2019

Keywords:
Context-aware recommender systems
Context-aware collaborative filtering
Item splitting
Graph-based models

ABSTRACT

Context-aware recommender systems (CARS) are specially designed to take into account the contextual conditions under which a user experiences an item, with the goal of generating improved recommendations. A known difficulty when constructing recommender systems is data sparseness, which reduces the effectiveness of collaborative filtering algorithms. While using contextual information provides fine-grained signals for the recommendation process, it makes the data even sparser and increases the computational complexity. In this paper, we present a method for making context-aware recommendations, which is less sensitive to data sparseness. The proposed method exploits the transitivity of the interactions between users and items on the user-item graph to augment the direct interactions, thus reducing the negative effect of sparse data. Based on graph transitivity we introduce a new graph-based association measure that we use as a measure of similarity between two users or two items in nearest neighbor recommendation methods. This combination of graph-based similarity measure with nearest neighbor methods allows considering more contextual conditions at a lower risk of being affected by data sparseness caused by additional contextual dimensions. We experimentally evaluated the proposed method on three contextually-tagged data sets. The results show that our method outperforms several baselines and state-of-the-art context-aware recommendation methods.

1. Introduction

Recommender systems are popular tools for assisting users in navigating through huge archives and alleviating the side effects of information overload in big data era. They have proved useful in different domains and online services such as Amazon.com, Netflix, Pandora, YouTube, and Twitter. A recommender system provides a personalized list of items, e.g. products, videos, songs, or news a user would be interested in. This is done by constructing models that learn from past interactions of users with items, e.g. purchases, views, clicks, and comments. Along this direction a great effort has been made to develop various collaborative filtering (Goldberg, Nichols, Oki, & Terry, 1992), content-based filtering algorithms (Balabanović & Shoham, 1997), or their combination in hybrid recommender systems (Burke, 2002). A common characteristic of these approaches is that they mainly focus on modeling users and items, e.g. via latent factors or other low rank structures of the user-item matrix.

In parallel, there is an understanding that it is also important to consider the context in which a recommendation is made. The context could be location, time of consuming the item, who is nearby, and the user's mood, etc. A user's preferences over the same item may vary as the context changes. For example, the user may choose a romantic movie to watch with her partner, but probably a cartoon if she is going to watch with children. Context-aware recommender systems (CARS) extend traditional recommendation approaches by considering the specific context in which the user rates or consumes an item and adapt the recommendation accordingly. There are three main approaches to make context-aware recommendations, namely pre-filtering, post-filtering, and contextual modeling (Adomavicius, Mobasher, Ricci, & Tuzhilin, 2011). Pre-filtering methods use contextual information to filter out irrelevant user-item interactions before applying conventional (context-free) recommendation methods. In contrast, post-filtering methods apply contextual information to the output of conventional (context-free) methods. In contextual modeling, contextual information is integrated with user and item information inside the recommendation algorithm. Experimental comparison of these approaches has shown that there are no clear winners among them

* Corresponding author.
E-mail addresses: phuongtm@ptit.edu.vn (T.M. Phuong), liendt@ptit.edu.vn (D.T. Lien), phuongnd@ptit.edu.vn (N.D. Phuong).
[1] **Tu Minh Phuong**: Conceptualization, Methodology, Investigation, Writing- Original draft preparation, Writing- Reviewing and Editing, Supervision.
[2] **Do Thi Lien**: Methodology, Software, Investigation, Writing- Original draft preparation, Writing- Reviewing and Editing, Visualization.
[3] **Nguyen Duy Phuong**: Conceptualization, Writing- Reviewing and Editing, Supervision.

https://doi.org/10.1016/j.eswa.2019.02.015
0957-4174/© 2019 Elsevier Ltd. All rights reserved.

(Panniello, Tuzhilin, & Gorgoglione, 2014), but pre-filtering and contextual modeling have attracted more research interest as witnessed by the number of published works.

A main difficulty when applying pre-filtering and contextual modeling approaches is data sparseness. A typical user has usually rated only a small number of items, making it difficult even for conventional recommendation methods to generate accurate prediction. The introduction of context further reduces the number of relevant ratings, in case of pre-filtering, or increases the data dimensionality, in case of contextual modeling, which both result in the increase of data sparseness. Another problem, especially for contextual modeling, is the substantial increase of computational complexity due to the introduction of additional data dimensions. A number of works have addressed these problems. For example, the *item splitting* method by Baltrunas and Ricci (2009) considers only a single context factor that is the most discriminative of rating values. Another approach groups similar context factors into clusters and consider each cluster as a new factor to reduce the number of context dimensions (Yin & Cui, 2016). However, consider only a single or smaller number of context factors may risk losing relevant information. And there is no single clustering method that works well for all type of context information.

In this paper, we propose a graph-based method for context-aware recommendation, which is designed to alleviate the negative effect of sparse data. To model context, our method uses the item splitting technique that introduces fictitious items by splitting original item ratings based on contextual conditions (Baltrunas & Ricci, 2009). However, unlike the original item splitting technique that allows a single split by a single contextual condition, here we split items using all combinations of context conditions. We represent the new user-item matrix as a bipartite user-item graph and calculate the similarity between two users or two items based on their connections on the graph. Using graph representation allows modeling distant, indirect connections among users or items and between them in forms of transitive relations on the graph, in addition to direct user-item connections that are very sparse after the item splitting step. By utilizing transitive relations on graph, the proposed method can reduce the negative impact of data sparseness, while does not require searching over combinations of the context conditions. The idea of using graph transitivity has been proposed in prior works (Huang, Chen, & Zeng, 2004; Phuong, Thang, & Phuong, 2008), in which items with the highest levels of connectivity with a user over the graph are selected for recommendations. In contrast, our method uses graph connectivity to calculate the similarity score between two users or two items and uses the calculated score as input for user-based or item-based nearest neighbor recommendations. We empirically show that the combination of graph-based similarity with nearest neighbor method yields more accurate recommendations than pure graph-based approach. In addition to the ability of modeling transitive relations between users and items, our method is computationally efficient since graph-based similarity computation can be performed by using efficient graph-propagation methods, the computational complexity of which is linear in the number of nodes (users and items) and number of edges (ratings). This allows taking into account more contextual factors while remaining efficient enough to be practical. We experimentally evaluated the proposed method on three context-tagged data sets and compared with other context-aware methods. The results demonstrate the effective ness of our method, which outperforms several baselines and a state-of-the-art context-aware recommendation method.

The rest of the paper is organized as follows. We discuss related works in Section 2. Then we present the proposed method Section 3. The experiments and evaluation results are presented in Section 4. Finally, Section 5 concludes the paper and discusses our future work.

2. Related work

The importance of considering context when generating recommendations has been recognized soon after the invention of recommender systems. One of the first work on CARS is by Adomavicius and Tuzhilin (2001), who proposed using additional data dimensions, besides users and items, to represent contextual information. Since then, a great research effort has been made on different aspects of CARS such as the impact of different types of context factors on recommendation quality, potential application domains, as well as ways to incorporate contextual information into recommendation generation process. Different types of context factors have been used with recommender systems, depending on specific applications and the availability of context-containing data. Popular context factors include time (Kefalas & Manolopoulos, 2017; Cai, Xu, Liu, & Pei, 2018), locations (Fan et al., 2017), types of devices, who is near by Odic, Tkalcic, Tasic, and Kosir (2013) and Sundermann, Domingues, Da Silva Conrado and Rezende (2016). Implicit contextual information is also considered, for example the temporal order of items in short sessions (Hidasi, Karatzoglou, Baltrunas, & Tikk, 2016; Tuan & Phuong, 2017; Phuong, Thanh, & Bach, 2018). More complex factors such as user activity can also be useful (Uslu & Baydere, 2015; Pham, Diep, & Phuong, 2013). CARS have shown to improve the recommendation quality in various application domains, ranging from movie and music recommendations (Odic et al., 2013; Baltrunas, Ludwig, & Ricci, 2011), social media (Yin & Cui, 2016; Macedo, Marinho, & Santos, 2015; Bach, Hai, & Phuong, 2016), point of interest recommendation (Cai et al., 2018), e-commerce (Sulthana & Ramasamy, 2018), web service recommendations (Fan et al., 2017), to more narrow applications such as wellness recommendations (Afzal et al., 2018), online learning (Wang & Wu, 2011), or transportation services (Tang, Chen, & Khattak, 2018). It has been shown that taking contextual information into account when generating recommendations improves customers' trust and other business performance measures (Panniello, Gorgoglione, & Tuzhilin, 2016). The importance of context has also been recognized in other decision-making scenarios such as in the Internet of Things (Garcia-de-Prado, Ortiz, & Boubeta-Puig, 2017).

To generate accurate context-aware recommendations, it is important to appropriately model and incorporate contextual information into the recommendation process. Methods for context modeling are categorized into three main approaches: pre-filtering, post-filtering and contextual modeling (Adomavicius et al., 2011). In pre-filtering approaches, contextual information is used to filter out irrelevant user-items interactions before applying conventional recommendation methods (Adomavicius, Sankaranarayanan, Sen, & Tuzhilin, 2005). A special case of pre-filtering is the techniques called "item splitting", proposed by Baltrunas and Ricci (2009). The item splitting method splits an item into fictitious items based on a contextual condition that discriminates the most the ratings of the original item. Symmetrically, users can also be split based on context of using items. In post-filtering approaches, contextual information is applied to the output of conventional recommendation methods to filter out irrelevant recommendations (Panniello et al., 2014). In contrast to pre-filtering and post-filtering approaches, contextual modeling methods use contextual information inside the recommendation algorithm by combining context with user and item information. Following the success of matrix factorization (MF) techniques (Koren, Bell, & Volinsky, 2009) for collaborative filtering, Karatzoglou, Amatriain, Baltrunas, and Oliver, (2010) proposed to apply MF on multi-dimensional tensors that contain also contextual dimensions. Following this approach, a number of factorization-based methods have been proposed, which aim to reduce the computational complexity (Baltrunas et al., 2011; Zou, Li, Tan, & Chen, 2015; Rendle, Gantner, Freudenthaler,

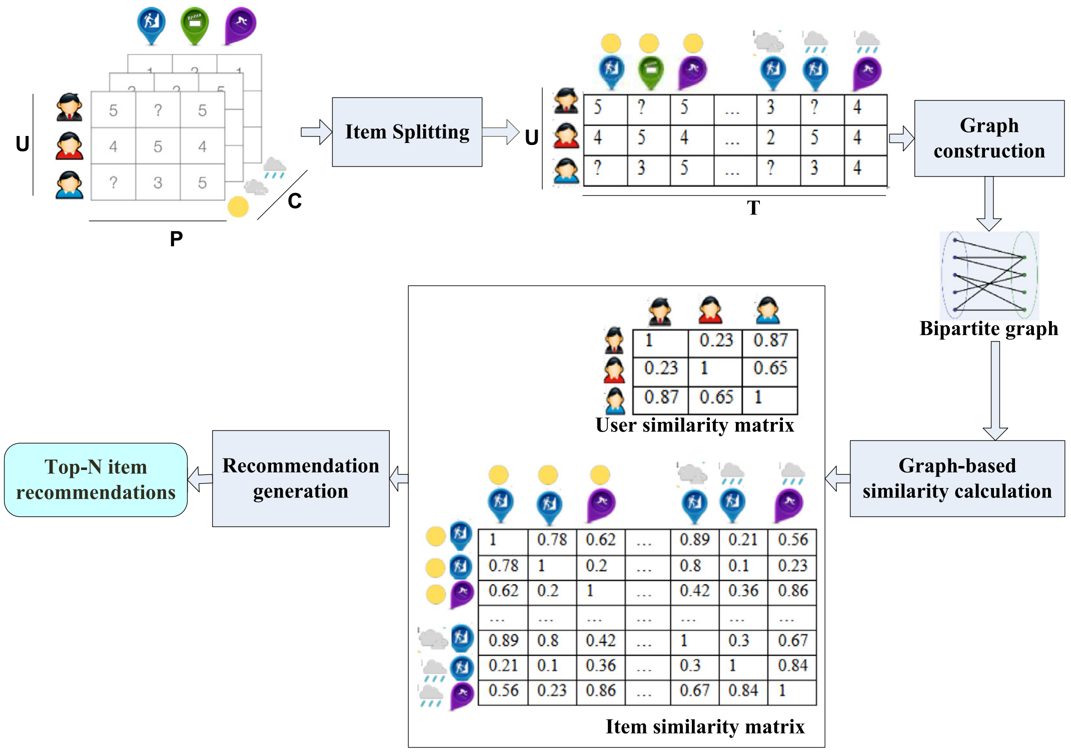


Fig. 1. Graph-based context-aware collaborative filtering framework.

& Schmidt-Thieme, 2011), or make the factorization machine to work with different types of input (Hidasi & Tikk, 2016). Besides MF, other matrix completion algorithms that are used in conventional collaborative filtering have also been modified to incorporate contextual information. Among such methods are CSLIM (Zheng, Mobasher, & Burke, 2014) that extends the SLIM algorithm (Ning & Karypis, 2011), context-aware low-rank matrix completion algorithm (Yu, Chan, & Yang, 2017) that extends standard matrix completion by using finite Abelian group algebras. Because introducing additional contextual dimensions increases data sparseness, many attempts to alleviate the negative effect of sparse data have been made. The original item splitting method constrains the number of resulting items by using only a single context factor for splitting. Other pre-filtering approaches try to reduce the number of contextual conditions by grouping them into clusters and using the clusters instead of original conditions (Yin & Cui, 2016), or applying dimensionality reduction techniques to reveal a smaller number of latent contextual factor (Unger, Bar, Shapira, & Rokach, 2016). In contrast to those methods, in this work, we do not constrain the number of context segments but address the data sparsity problem by using graph-based transitivity that can catch indirect associations among users and items. Another problem with pre-filtering and contextual modeling approaches is computational complexity, which can grows exponentially with the data dimensions. Our method uses an efficient graph-based propagation algorithm, the complexity of which is linear in the number of edges (ratings) and nodes (users and items) (see Section 3.5 for the complexity analysis), to compute user-user or item-item associations and can allow considering a large number of contextual segments.

Graph-based methods have proved useful in developing recommendation algorithms with a number of works reported. One of the first graph-based collaborative filtering algorithm was proposed by Huang et al. (2004), who compute the association between a user and an item as the connections between the corresponding nodes on the user-item graph. Fouss, Pirotte, Renders, and Saerens (2007) propose a random walk method to compute the similarities between a user and an item nodes on the user-item graph. A somewhat different approach is predicting user-item association as a link prediction using supervised learning techniques (Li & Chen, 2013). Yang and Toni (2018) propose a method to reduce the dimensionality of the recommendation problem by constructing and clustering user graphs. Another use of graph is to represent domain knowledge that is used to improve recommendations in complex domains (Deladienne & Naudet, 2017). Graph-based representation has also been extended to represent content features in hybrid recommender systems as described in Phuong et al. (2008). In this work, we also use graphs to represent the relations between users and split items, but unlike previous methods, we combine graph-based similarity computation with traditional nearest neighbor approach, which we will experimentally to outperform pure graph based recommendation.

3. Methods

In this section we describe our method for making context-aware recommendations. The proposed methods generates recommendations using the following steps (shown in Fig. 1):

- *Item splitting*. This step transforms the original multidimensional user-item rating matrix over multiple context dimensions into a two-dimensional matrix by creating fictitious items.
- *Graph construction*. In this step, we create graph representation of the rating matrix formed in the previous step.
- *Graph-based similarity calculation*. In this step, we calculate the similarity scores between users or between items as the association measures over the graph.
- *Recommendation generation*. This is the final step, which generates recommendations by using k-NN with the similarity scores computed in the previous step.

We describe each of these steps in detail in the following sections.

Table 1
Example of *user* × *item* × *context* matrix of ratings.

User	Item	Rating	Time	Location	Companion
u_1	p_1	5	Weekend	Home	Kids
u_1	p_1	4	Weekday	Home	Family
u_2	p_1	3	Weekend	Cinema	Partner
u_2	p_1	4	Weekday	Home	Family
u_3	p_1	3	Weekend	Cinema	Partner
u_3	p_2	2	Weekend	Cinema	Partner

3.1. Item splitting

Let $U=\{u_1,u_2,...,u_N\}$ denote the set of N users, $P=\{p_1,p_2,..., p_M\}$ denote the set of M items and $C_1,C_2,...,C_n$ denote n context factors or dimensions, each context dimension has $N_{c_1}, N_{c_2}, \ldots, N_{c_n}$ context conditions respectively. We consider the settings in which we are given an incomplete matrix of ratings R_o that the users have given to the items in concrete context situations. Table 1 shows an example of such a matrix, where Time, Location, and Companion are the context factors. Note that this is a 2D representation of the original matrix, which is multidimensional by nature and contains many blank entries.

We first transform this multidimensional matrix into a 2D user-item rating matrix by introducing fictitious items, each of which is a combination of an original item and a contextual condition. This procedure is known as "Item splitting" (Baltrunas & Ricci, 2009). Since item splitting increases the number of items, which consequently makes the *user* × *item* matrix sparser and increases computational complexity, existing item splitting techniques mainly consider one context factor at a time and select a single most significant split by using statistical measures such as information gain (Baltrunas et al., 2011; Zheng, Burke, & Mobasher, 2014). We will show that our proposed graph-based method is able to deal with data sparseness and thus is less sensitive to the negative effect of item splitting. This allows us to consider all possible context condition combinations without any pruning or selection step. Specifically, we generate context combinations and split items using the following steps:

- Step 1. Calculate the Cartesian product of all context dimensions. That is, we create a new context dimension C with N_c context conditions, where $N_c = N_{c_1} \times N_{c_2} \times \ldots \times N_{c_n}$, whose elements are combinations of context conditions from the original context factors.
- Step 2. Calculate the Cartesian product of the item set P and the context dimension C. Each element of this product is a combination of an item from P with a context condition from C and is considered as a new fictitious item. This results in a new item set T of size $H=M\times N_c$.
- Step 3. Transform the original multi-dimensional matrix of ratings to two-dimensional matrix of ratings by eliminating contexts from the original context-aware data set and replacing the item set P by the item set T.

Table 2 shows the results of applying these steps to data shown in Table 1, in which, for example t_1 is a new fictitious item corresponding to the combination of original item p_1 and the context situation "Weekend, Home, Kids". From two original items and 12 contextual situations, the item splitting produces 24 new items. Note that, for clarity, we intentionally number the new items and arrange the table so that all user-item pairs with ratings are placed on the top rows (there are six such pairs in this example). Rows that correspond to user-item pairs with no ratings are not shown fully to save space.

Table 2
Results of item splitting.

User	Item	Rating
u_1	t_1	5
u_1	t_3	4
u_2	t_2	3
u_2	t_3	4
u_3	t_2	3
u_3	t_4	2
u_1	t_2	$\emptyset$
...	...	...
u_3	t_{24}	$\emptyset$

Table 3
The transformed rating matrix (the last columns contain no ratings).

	t_1	t_2	t_3	t_4	t_5	...	t_{24}
u_1	1	$\emptyset$	0.8	$\emptyset$	$\emptyset$	...	$\emptyset$
u_2	$\emptyset$	0.6	0.8	$\emptyset$	$\emptyset$	...	$\emptyset$
u_3	$\emptyset$	0.6	$\emptyset$	0.4	$\emptyset$	...	$\emptyset$

3.2. Graph construction

Let R denote the new rating matrix of size $|U| \times |T|$ resulted from the item splitting procedure so that element r_{ij} is the rating user u_i has given to item t_j. We assume that r_{ij} take on a value in {0,1] if user u_i has rated t_j or empty otherwise, i.e.:

$$r_{ij} = \begin{cases} v & \textit{If the user i rated the item j with v level } (0 < v \leq 1) \\ \emptyset & \textit{If the user i haven't known the item x yet} \end{cases} \quad (1)$$

For datasets with rating values $r_{ij} \in \{1, 2, \ldots, g\}$, this can be achieved by normalization: $r_{ij}=r_{ij}/g$. Table 3 shows the result of this transformation applied to data in Table 2. Recall that we intentionally number new items so that rated items get lower indices.

The task now is to estimate the rating values for the empty elements of R. For a given user u_a, the top-N recommendations are then generated as N unseen items with highest estimated ratings.

We represent the data as a bipartite graph $G=\langle V, E\rangle$. The set of nodes V consists of the set of user nodes U so that each node of U corresponds to a user, and the set of item node T so that each node of T corresponds to an item ($V=U \cup T$). The set of edges E represent the interactions of users with items so that an edge $e_{ij} \in E$ connecting a user node u_i and item node t_j represents the rating given by u_i to t_j if such exists. There are no edges between two user nodes or two item nodes. Each edge e_{ij} is assigned a weight w_{ij} as follows:

$$E = \{e = (i, j) : i \in U,\ \ x \in T\ |r_{ij} \neq \emptyset\} \quad (2)$$

$$w_{ij} = \begin{cases} r_{ij} & If\ \ (i, j) \in E \\ 0 & Otherwise \end{cases} \quad (3)$$

Fig. 2 shows the graph representation for the data from Table 3. The graph is sparse: there are no links to item nodes t_5 ... t_{24} because those new items have not been rated.

3.3. Graph-based similarity calculation

K-Nearest Neighbor (k-NN) algorithms are among the most popular collaborative filtering methods. Algorithms of this class can be user-based or item-based. User-based k-NN algorithms rely on similarity between users to select users that are most similar to the active one and calculate recommendations from this neighborhood. On the other side, item-based k-NN rely on item-item similarities

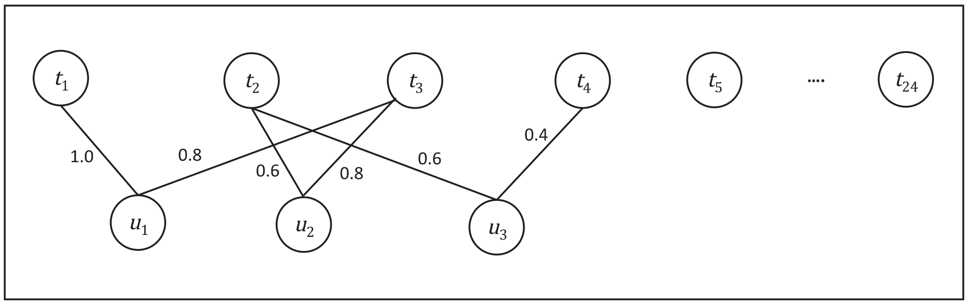


Fig. 2. Example of graph representation.

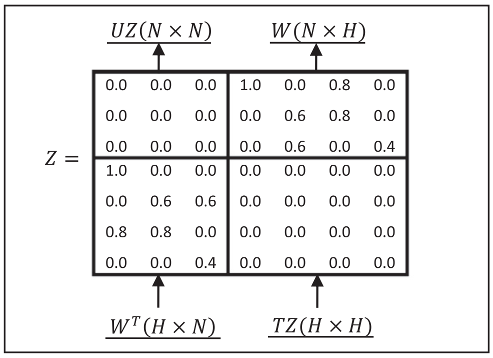
to find neighbors for a given item. Accurate calculation of similarities (or distance) is critical for the success of the methods. In traditional user-based k-NN methods, the similarity between user u and user v is calculated as the Pearson correlation or cosine between two vectors, elements of which are ratings given by u and v for co-rated items. The similarity between two items in item-based k-NN is calculated similarly based on users having co-rated the two items. When rating data are sparse, which is often the case, the number of items co-rated by two users (or number of users having co-rated two item in item-based k-NN) is small or equal to zero, making it unreliable or impossible to calculate the similarity. This problem is more severe after item splitting, which makes the user-item data sparser. To illustrate this problem, let us consider the following example from Fig. 2. As shown in Fig. 2, users u_1 and u_3 have not co-rated any item and thus it is impossible to calculate the similarity between them if using only direct user-item connections. However, if we consider also indirect connections by transitivity, we can see that u_1 and u_2 have co-rated item t_3, u_2 and u_3 have co-rated item t_2. Based on transitive relations between u_1 and u_2, u_2 and u_3, it is possible to infer the relation between u_1 and u_3.

The idea of using transitive relations on graphs has been introduced by Huang et al. (2004), who calculate the associations between a user i and an item j as the sum of (weighted) paths connecting node i and node j on the graph. Items with the strongest associations to the active user are then selected for recommendations. A drawback of this method is that the direct calculation of user-item association may result in spurious relationships, which are harmful for recommendation accuracy, especially when using long paths. In this work, we propose using the graph to calculate the similarities between users or between items and then using the calculated similarity scores for user-based or item-based k-NN. As we will show experimentally, the combination of graph-based similarity and nearest neighbor methods yields better results compared to direct graph-based methods. In what follow we describe the graph-based similarity calculation and provide theoretical insight of associations on graphs.

Given a pair of user nodes u_i and u_j, we define the similarity between them as the sum of weights of all distinctive paths connecting u_i and u_j. Note that the length of a path connecting two user nodes is an even number (2,4,6,...). The weight of a path is the product of the weights of the path's edges. For example, in Fig. 2, the path $u_1 - t_3 - u_2 - t_2 - u_3$ between u_1 and u_3 has the length of 4 and weight $0.8 \times 0.8 \times 0.6 \times 0.6 = 0.2304$. Because an edge's weight is less than 1, longer paths tend to have lower weights. In practice we do not consider paths whose length exceeds L, where L is a predefined parameter.

Similarly, we can calculate the similarity between two items as the sum of weights of the paths connecting the corresponding item nodes.

Formally, let us define $Z = \{z_{ij}\}$ as the adjacency matrix of the graph $G(i = 1, 2, \ldots(N+H);\ j = 1, 2, \ldots(N+H))$. The square matrix Z



$UZ(N \times N)$
$W(N \times H)$
$Z =$
$W^T(H \times N)$
$TZ(H \times H)$

$$Z = \begin{array}{ccc|cccc}
0.0 & 0.0 & 0.0 & 1.0 & 0.0 & 0.8 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 & 0.6 & 0.8 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 & 0.6 & 0.0 & 0.4 \\
\hline
1.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.6 & 0.6 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.8 & 0.8 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.4 & 0.0 & 0.0 & 0.0 & 0.0
\end{array}$$

Fig. 3. The matrix representation for bipartite graph G.

can be decomposed into four parts as follows:

$$Z = \begin{pmatrix} UZ(N \times N) & W(N \times H) \\ W^T(H \times N) & TZ(H \times H) \end{pmatrix} \tag{4}$$

Note that the UZ and TZ components are zero matrices.

For path length L, the transitive associations or similarity between two users are given in matrix UZ^L of size $N \times N$ computed as follows:

$$UZ^L = \begin{cases} W.W^T, & L = 2 \\ W.W^T.UZ^{L-2}, & L = 4, 6, 8, \ldots \end{cases} \tag{5}$$

Where uz_{ij}^L is the similarity between user node i and user node j over the paths of lengths up to L.

Similarly, we can calculate the similarity score between any two items, which are elements of sub-matrix TZ.

As an example, for the graph G in Fig. 2, the adjacency matrix Z is given in Fig. 3.

The similarity scores between two users for $L = 2, 4, 6$ are given below:

$$UZ^2 = \begin{bmatrix} 1.64 & 0.64 & 0.00 \\ 0.64 & 1.00 & 0.36 \\ 0.00 & 0.36 & 0.52 \end{bmatrix},$$

$$UZ^4 = \begin{bmatrix} 3.0992 & 1.6896 & 0.2304 \\ 1.6896 & 1.5392 & 0.5472 \\ 0.2304 & 0.5472 & 0.4000 \end{bmatrix},$$

$$UZ^6 = \begin{bmatrix} 6.164032 & 3.756032 & 0.728064 \\ 3.756032 & 2.817536 & 0.838656 \\ 0.728064 & 0.838656 & 0.404992 \end{bmatrix}$$

```
Input:
- Multi-dimensional matrix $R_o$ (including contextual information).
- $u_a \in U$ is the active user that needs recommendations.
- $c \in (C_1 \times C_2 \times \ldots \times C_n)$ is the current contextual situation of the active user.
- $L_{max}$ is the maximum path length.
- $K_1$ is the number of neighbors for the active user.
- $K_2$ is the number of items that will be recommended to the active user.

Output:
- Top $K_2$ items recommended for $u_a$ in the context situation $c$.

Steps:

Step 1. Transform the original multi-dimensional matrix into 2D $user \times item$ rating matrix
    Perform the "item splitting" procedure described in section 3.1

Step 2. Calculate similarities between users based on graph
        Construct graph as described in section 3.2
        $L \Leftarrow 2$;// Set the initial path length
        Repeat
          $UZ^L = \begin{cases} W.W^T, & L = 2 \\ W.W^T.UZ^{L-2}, & L = 4,6,8,\ldots \end{cases}$
          $L \Leftarrow L + 2$; // Increasing path length.
        Until ($L = L_{max}$ or $uz_{ij}^L \neq 0$ with every $u_j \in (U \backslash u_i)$);

Step 3. Generate recommendations for user $u_a$
    • For $u_a$, select $K_1$ users with the highest similarity scores to $u_a$. Let $U_a$ denote the set of these $K_1$ users.
    • Estimate each missing rating $r_{aj}$ of $u_a$ for item $t_j$:
          $r_{aj} = \frac{\sum_{r_{ij} \in R_j} r_{ij}}{|R_j|}, R_j = \{r_{ij} | r_{ij} \neq \emptyset, t_i \in U_a\}$
    • Map the 2D rating matrix with fictitious items (from $T$) to muti-dimensional matrix with real items (from $P$)
      and contextual situations (from $C$).
    • Return $K_2$ real items with the highest ratings as recommendations for user $u_a$ in contextual situation $c$.
```

Fig. 4. IS-UserBased-Graph algorithm.

3.4. Recommendation generation

Once graph-based similarity scores have been computed we generate recommendations using the standard k-NN approach. Specifically, the user-based k-NN recommendation is performed as follows. For the active user u_a, we select K users with the highest similarity scores to u_a. Let U_a denote the set of these K users. Each missing rating r_{aj} of u_a for item t_j is then estimated by averaging over existing ratings for t_j given by the users from U_a:

$$r_{aj} = \frac{\sum_{r_{ij} \in R_j} r_{ij}}{|R_j|}, \quad R_j = \{r_{ij} | r_{ij} \neq \emptyset, t_i \in U_a\} \tag{6}$$

Unrated items with the highest ratings are then recommended for user u_a.

The item-based k-NN is performed as follows. For each item t_j not rated by the active user u_a we find K items that are: 1) already rated by u_a; and 2) the most similar to t_j based on the graph-based item-item similarity scores. Let T_a denote the set of such items. The rating of u_a for t_j is then estimated by averaging over ratings given by u_a to items from T_a.

Fig. 4 shows the algorithm with all steps described above when using user-based k-NN. Item-based k-NN is similar and is not shown here to save the space.

3.5. Spreading activation on graphs

A key step of the proposed algorithm is computing similarity scores. Although this step can be done by matrix multiplication as described above, it is computationally expensive. Since graph G is sparse in the number of edges, matrix multiplication can be avoided by using spreading activation algorithms on graph G. In this work, we adopt the spreading activation algorithm from Weston, Elisseeff, Zhou, Leslie, and Noble (2004) to compute the similarity between pairs of users and pairs items. The spreading activation algorithm for the active user u_a and maximum path length L is shown in Fig. 5. Note that this algorithm considers only existing edges, which correspond to existing ratings.

Let V_a denote the set of nodes of G except u_a, $n_i \in V_a$ can be a user node or an item node and z_{ij} is a linking weight between n_i and n_j. Let $s_i(t)$ be the weight sum of all paths of lengths up to t connecting u_a with n_i. The algorithm performs L iterations.

Note that we need to perform the update inside the "for" loop in line 5 only for non-zero z_{ji}. This "for" loop with condition can be implemented efficiently using sparse representation: instead of storing full matrix Z, it is enough to store, for each n_j, indices i and values of non-zero z_{ji}. Since non-zero elements of Z correspond to existing ratings, the complexity of "for" loop in line 5 is $O(|R|)$, with $|R|$ denoting the number of existing ratings. The complexity of the spreading activation algorithm with sparse representation, therefore, reduces to $O(L|R||V|)$.

```
Input:
- Graph G with nodes V and adjacency matrix Z (described in section 3.2)
- $\alpha$ $(0 \le \alpha \le 1)$ is the path weight decay parameter.
- $u_a \in U$ is the active user.
- $V_a = V \backslash u_a$
- L is the maximum path length from $u_a$ to other users $u_k \in U$.

Output:
- Similarity scores between $u_a$ and other users $u_k \in U$.

Steps:
1.  Initialize $s_i(0) = 0$ if $n_i$ is a user node, $s_i(0) = z_{ai}$ if $n_i$ is an item node ($n_i \in G_a$);
2.  for $t = 1,2,\dots,L$
3.      for $n_i \in V_a$ do
4.          $s_i(t) = z_{ai}$;
5.          for $n_j \in V_a$ and $z_{ji} \ne 0$ do
6.              $s_i(t) \leftarrow s_i(t) + \alpha z_{ji} s_j(t-1)$;
7.          end for
8.      end for
9.  end for
10. return ($s_i(L)$; // The similarity score between $u_a$ and user $u_i \in U$ for max path length $L$.
```

Fig. 5. The spreading activation algorithm for calculating similarities between users.

4. Experiments and results

In this section we describe the experiments to evaluate the effectiveness of the proposed method and compared with several baselines and state-of-the-art context-aware methods on real-world datasets. We provide results and analysis.

4.1. Data sets

We used three data sets: DepaulMovie, MovieLens, InCarMusic in our experiments.

- The DepaulMovie data set (Zheng, Mobasher, & Burke, 2015) comprises 5043 ratings from 97 users for 79 movies over three contextual dimensions. The contextual dimensions are *Time*, *Location*, and *Companion*. Each contextual dimension has multiple discrete values: *Time* has two contextual conditions (Weekend, Weekday), *Location* has two values (Home, Cinema), and *Companion* has three values (Alone, Family, Partner). Rating value are from 1 to 5, which are converted to {0,1] range as described above. Sparse level of rating data is 94,516%.
- The MovieLens 100 K data set[1] contains 100,000 ratings given by 973 users for 1682 movies. Based on timestamp, two context dimensions were extracted, namely *TimeOfDay* and *DayOfWeek*. *TimeOfDay* has five values corresponding to five context conditions (Morning, Noon, Afternoon, Evening, Night). *DayOfWeek* can take on one of two values (Weekday, Weekend). Rating values are 1 to 5, which are converted to {0,1]. Sparse level of the dataset is 93,89%.
- The InCarMusic data set[2] comprises 3938 ratings given by 1042 users for 139 albums over eight contextual dimensions. The contextual dimensions are *Driving style, Road type, Landscape, Sleepiness, Traffic conditions, Mood, Weather, Natural Phenomena.* The context conditions of each context factors are as follows: *Driving style* (Relaxed driving, Sport driving), *Road type* (City, Highway, Serpentine), *Landscape* (Coast line, country side, mountains/hills, Urban), *Sleepiness* (Awake, Sleepy), *Traffic* (Free road, Many Cars, Traffic jam), *Mood* (Active, Happy, Lazy, Sad), *Weather* (Cloudy, Snowing, Sunny, Rainy), *Natural Phenomena* (Day time, Morning, Night, Afternoon). Rating values from 1 to 5, sparse level of rating data is 99.99%.

4.2. Experimental setup

Metrics. There are two types of recommendation tasks: *rating prediction* and *item recommendation* (or *top-N recommendation*). For the rating prediction, the most popular evaluation metrics are *Mean Average Error* (*MAE*), *Root Mean Square Error* (*RMSE*), and *Mean Prediction Error* (*MPE*). The item prediction can be evaluated by using ranking metrics such as *Precision@N, Recall@N*, and *MAP@N*. In this work, we focus on top-N recommendation, in which *N* top scored items are recommended to the user. We also compare our method with other top-N recommendation algorithms. Therefore, we used *Precision@N* (or *P@N*) and *Mean Average Precision* (*MAP@N*), which are common for the top-N recommendation settings. Given a ranked list of items, *Precision@N* is the fraction of items that are relevant among the top-N positions:

$$Precision@N = \frac{|\{relevant\ items\} \cap \{top - N\ items\}|}{N} \tag{7}$$

Mean Average Precision (*MAP@N*) is defined through *Average Precision* (*AP@N*) as follows. Let *N* be the number of items to recommend, *m* be the number of relevant items in the full space of items. We compute the precision at each relevant position and average them.

$$AP@N = \frac{1}{m}\sum_{k=1}^{N}\left(P(k)\ if\ k^{th}\ item\ was\ relevant\right) = \frac{1}{m}\sum_{k=1}^{N} P(k).rel(k) \tag{8}$$

where $rel(k) = 1$ if the kth item is relevant and $rel(k) = 0$ otherwise.

[1] http://www.grouplens.org/

[2] https://github.com/irecsys/CARSKit/blob/master/context-aware_data_sets/Music_InCarMusic.zip

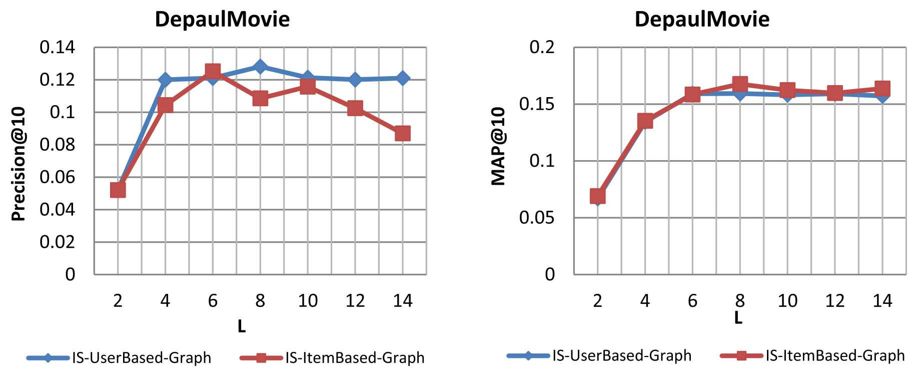


Fig. 6. The effect of maximum path length L for DepaulMovie data set.

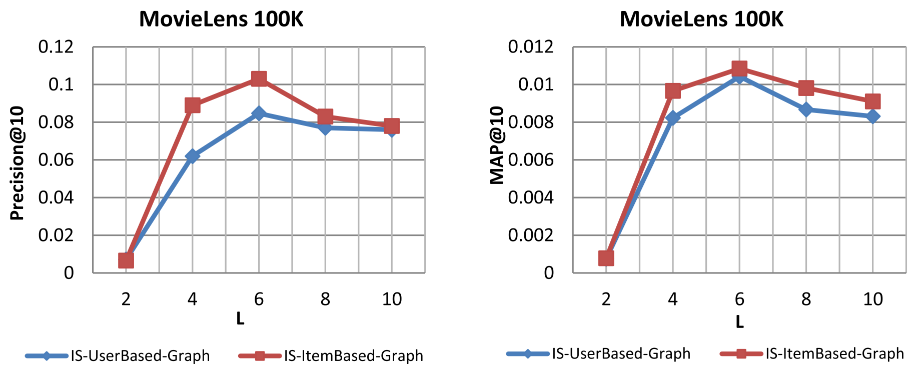


Fig. 7. The effect of maximum path length L for MovieLens 100 K data set.

Since $AP@N$ is defined for a single user, we average the $AP@N$ over all $|U|$ users.

$$MAP@N = \frac{1}{|U|} \sum_{i=1}^{|U|} (AP@N)_{u_i} \quad (9)$$

We used $N = 10$ in all experiments.

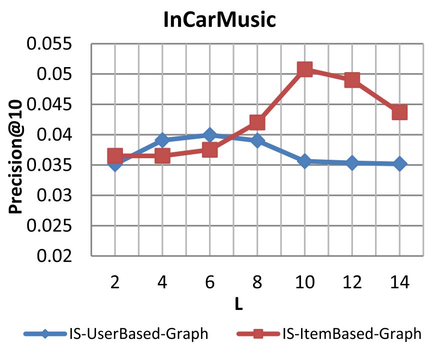
Cross-validation. We used 10-fold cross validation on the set of users. Specifically, the set of users is divided into 10 subsets of similar sizes. We withheld one subset as the test set and the remaining subsets as the training set. We repeated this procedure ten times for different test sets and report the averaged metrics over ten folds. For each user in the test set, we withheld 20% of rated items (the last rated items based on timestamps). The predictions made by the system are then compared with the withheld data to calculate the evaluation metrics.

Baselines and methods to compare. We evaluated several variants of the proposed method and compared with several baselines and state-of-the-art context-aware recommendation methods:

- *UserSplitting-BiasedMF.* This method splits one user into two different users depending on contextual situation associated with users (Zheng, Burke et al., 2014). After splitting, BiasedMF (Biased-Matrix Factorization) (Koren et al., 2009) is used to train the recommendation model and generate predictions.
- *ItemSplitting-BiasedMF.* This method splits one item into two different items depending on the context (Baltrunas & Ricci, 2014) and applies BiasedMF to the resulting matrix to make predictions.
- *UISplitting-BasedMF.* This method combines both UserSplitting and ItemSplitting, followed by BiasedMF to train the recommendation model and make predictions.
- *CSLIM* (Zheng, Burke et al., 2014). This method extends the SLIM algorithm (Ning & Karypis, 2011) to incorporate contextual information. CSLIM is chosen for comparison due to its reported superiority over other context-aware MF methods. We used the CSLIM_MCS variant of CSLIM provided within CARSKIT software (Zheng et al., 2015).
- *ItemSplitting-SLIM.* This method combines Item Splitting procedure with SLIM to train the recommendation model and make predictions.
- *ItemSplitting-Graph.* This method combined Item Splitting procedure with pure graph-based recommendations as described by Huang (Huang et al., 2004). We set the maximum path length to 5 as this has been reported to give the best results [8].
- *IS-UserBased-Graph.* This is our proposed method, which combines graph-based user-user similarity with user-based kNN recommendations.
- *IS-ItemBased-Graph.* This is our proposed method, which combines graph-based item-item similarity with item-based kNN recommendations.

4.3. Results

The first experiment was designed to evaluate the effect of transitivity relations on recommendation accuracy. The transitivity range is specified by parameter L, which is the maximum path length used when calculating graph-based similarity scores. We gradually increased L from 2 by a step of 2 and calculate precisions for the resulting models. Figs. 6–8 show *Precision*@10 and *MAP*@10 scores for different values of L on DepaulMovie, MovieLens, and InCarMusic data sets respectively. On DepaulMovie and MovieLens, both *Precision*@10 and *MAP*@10 values increase fast for the first several values of L, then they stabilize of decrease as L continues to increase. The only variation from this pattern is *Precision*@10 of IS-ItemBased-Graph for InCarMusic data set, which does not change much as L increases from 2 to 6 then improves fast until L reaches 10. The observed patterns of precision values show that using



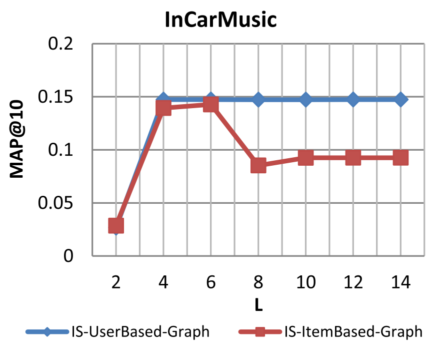


Fig. 8. The effect of maximum path length L for InCarMusic data set.

Table 4
Performance comparison results for DepaulMovie data set.

Methods	Precision@10	MAP@10
UserSplitting-BiasedMF	0.089	0.162
ItemSplitting-BiasedMF	0.086	0.147
UISplitting-BiasedMF	0.084	0.144
CSLIM	0.085	0.121
ItemSplitting-SLIM	0.092	0.158
ItemSplitting-Graph	0.117	0.148
IS-UserBased-Graph	0.121	0.159
IS-ItemBased-Graph	**0.125**	0.158

Table 5
Performance comparison results for MovieLens 100 K data set.

Methods	Precision@10	MAP@10
UserSplitting-BiasedMF	0.030	0.0076
ItemSplitting-BiasedMF	0.029	0.0065
UISplitting-BiasedMF	0.028	0.0066
CSLIM	0.004	0.0005
ItemSplitting-SLIM	0.023	0.0061
ItemSplitting-Graph	0.081	0.0089
IS-UserBased-Graph	0.085	0.0104
IS-ItemBased-Graph	**0.103**	**0.0108**

Table 6
Performance comparison results for InCarMusic data set.

Methods	Precision@10	MAP@10
UserSplitting-BiasedMF	0.033	0.125
ItemSplitting-BiasedMF	0.034	0.127
UISplitting-BiasedMF	0.033	0.117
CSLIM	0.018	0.038
ItemSplitting-SLIM	0.023	0.065
ItemSplitting-Graph	0.014	0.115
IS-UserBased-Graph	0.034	**0.147**
IS-ItemBased-Graph	**0.037**	0.142

transitivity relations in forms of indirect connections between nodes improves recommendation accuracy but using too long transitivity relations does not lead to further improvements and may decrease the performance. A possible explanation for this phenomenon is that long-range connections may contain spurious associations that do not reflect the real relations between users or items. In general, item-based models are more sensitive to the change of L than user-based models, but the former also achieved better precisions in four out of six cases. Except for the *Precision*@10 of the item-based model on InCarMusic, the best performance was achieved with L value of 6 or 8, showing that using $L = 6$ is appropriate for most cases. In the next experiments we fixed the maximum path length L at 6.

In the next experiment, we compared our methods with other baselines. Our methods were run with $L = 6$ and 30 neighbours. We ran other methods with different parameters' values and report the best performance. Tables 4–6 summarize the *Precision*@10 and *MAP*@10 for DepaulMovie, MovieLens and InCarMusic data sets, respectively. The results show that splitting methods followed by matrix factorization performs better than CSLIM, which is an representative of context modeling approaches. In particular, CSLIM consistently achieved lower precision scores than ItemSplitting-SLIM that is a simple combination of item splitting with original SLIM. This result shows the competitiveness of splitting approach, which motivates its use in our proposed method. Although UserSplitting-BiasedMF achieved higher *MAP*@10 on DepaulMovie, there is no clear winner between user splitting or item splitting in general. This observation is consistent with previous findings (Zheng, Burke et al., 2014).

The performance of ItemSplitting-Graph changes from data set to data set. This method achieved relatively high *Precision*@10 on DepaulMovie and MovieLens, beating MF based and SLIM based methods but achieved lower precision on InCarMusic. Such unstable performance of ItemSplitting-Graph suggests that making recommendations based on direct user-item associations may be not the best way to utilize graph-based connectivity, and other methods like the one we propose in this work should be considered.

In general, both user-based and item-based variants of our method achieved the best performance in terms of precision scores. Specifically, the item-based variant of our method, namely IS-ItemBased-Graph, achieved the highest *Precision*@10 scores for all experimented data sets and the highest *MAP*@10 on MovieLens. The user-based variant (IS-UserBased-Graph) achieved the highest *MAP*@10 on MovieLens and InCarMusic. The only exception is *MAP*@10 on DepaulMovie data set, for which the highest value was achieved by UserSplitting-BiasedMF. Since DepaulMovie is the least sparse data set, these results provide evidence that our method is less sensitive to data sparseness, despite the fact that we allowed all contextual segments. In general, the item-based variant achieved better precision than the user-based variant, which is not surprising since we split items, thus providing finer grained information over the item dimension. The superiority of our methods over original graph-based method shows the usefulness of combining graph-based transitivity with k-NN.

To compare the computation times of the experimented methods, we recorded training and prediction times of each method on different data sets. All models were implemented in Java and run on a PC with Intel Core i7-3770 CPU and 8GB RAM. Figs. 9–11 show the average computation times of each fold on three data sets. As can be seen, the training times of graph-based methods are shorter than those of factorization and SLIM-based methods in all cases. This is not surprising since the training of the graph-based models consists only of item splitting and sparse matrix construction.

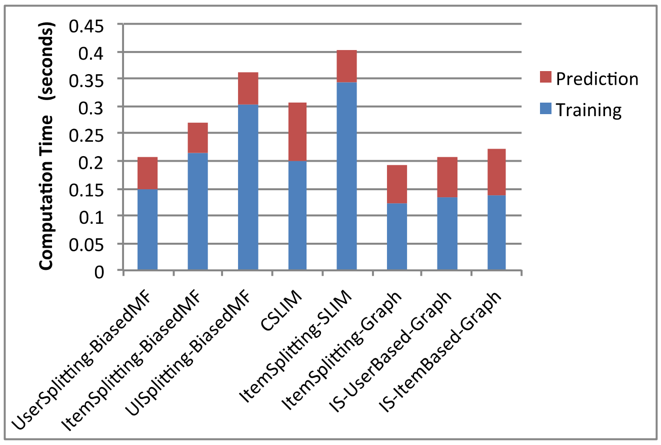


Fig. 9. Computation time on DepauMovie data set.

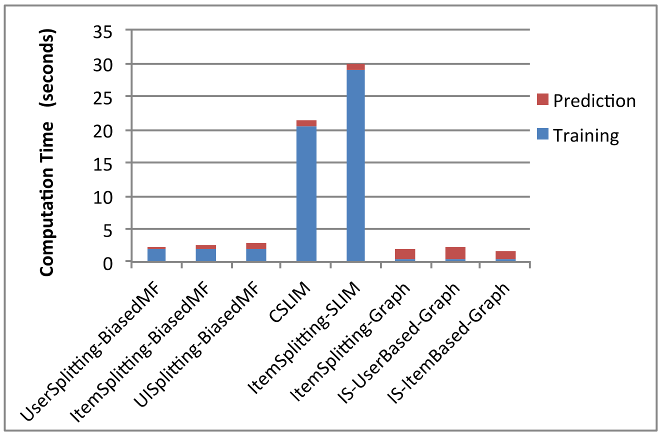


Fig. 10. Computation time on MovieLens 100 K data set.

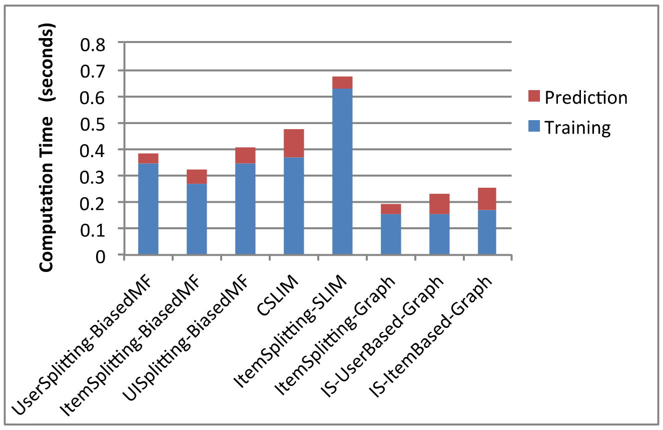


Fig. 11. Computation time on InCarMusic data set.

At test time, graph-based methods required slightly more time than factorization methods to generate recommendations. There are no differences in prediction time between graph-based and SLIM based methods. Among graph-based methods, variants of our method were slightly slower than the pure graph algorithm by Huang, mostly due to the k-NN step. Note that in experiments, we considered all graph-based similarity computation as a prediction step. In practice, the similarity scores between users can be pre-computed offline and stored to make prediction time even shorter.

The experimental results also show that IS-UserBased-Graph and IS-IserBased-Graph are very efficient at making predictions. On MovieLens - the largest data set - both methods take less than 1.2 s to perform context-filtering and generate recommendations for one fold of 97 users, which means it requires only about 0.012 s to generate recommendations for a single active user. The methods take even less time to make predictions on the remaining data sets. The average time required to generate recommendations, for a single user, by either IS-UserBased-Graph or IS-IserBased-Graph are less than 0.007 s and 0.004 s on DepauMovie and InCarMusic, respectively. These results clearly show that the proposed methods are efficient enough to perform context filtering and generate recommendations in real time.

5. Conclusion

We have presented a new context-aware recommendation method. The proposed method exploits all available contextual information by transforming the original multi-dimensional matrix of ratings into two-dimensional matrix of ratings using a modified item splitting procedure. We then represent the resulting matrix as a bipartite graph and compute the similarity between two users or two items as the connections between the corresponding nodes. We use the computed similarity scores as input to item-based or user-based k-NN. The use of graph connections allows catching indirect relations among users and items, which complement the direct ones and are very useful when the matrix is sparse. This brings the main advantage of the proposed method: It allows considering more contextual segments at lower risk of being affected by data sparseness. The method also has computational complexity linear in the number of users + items and ratings, which is important for applicability in domains with many items resulted from splitting on many contextual factors. We present experimental results on three context-containing data sets, exploring different depths of graph-based transitivity, and comparing with several baselines and powerful context-aware recommendation approaches. The results demonstrate the superiority of the proposed method in terms of recommendation accuracy, as well as the appropriateness of individual design choices.

Credit authorship contribution statement

Tu Minh Phuong: Conceptualization, Methodology, Investigation, Writing - original draft, Writing - review & editing, Supervision. **Do Thi Lien:** Methodology, Software, Investigation, Writing - original draft, Writing - review & editing, Visualization. **Nguyen Duy Phuong:** Conceptualization, Writing - review & editing, Supervision.

References

Adomavicius, G., Mobasher, B., Ricci, F., & Tuzhilin, A. (2011). Context-aware recommender systems. *AI Magazine, 32*(3), 67–80.
Adomavicius, G., Sankaranarayanan, R., Sen, S., & Tuzhilin, A. (2005). Incorporating contextual information in recommender systems using a multidimensional approach. *ACM Transactions on Information Systems, 23*(1), 103–145. https://doi.org/10.1145/1055709.1055714.
Adomavicius, G., & Tuzhilin, A. (2001). Multidimensional recommender systems: A data warehousing approach. In *Proceedings of the second international workshop on electronic commerce* (pp. 180–192). Springer-Verlag.
Afzal, M., Ali, S. I., Ali, R., Hussain, M., Ali, T., Khan, W. A., et al. (2018). Personalization of wellness recommendations using contextual interpretation. *Expert Systems with Applications, 96*, 506–521. https://doi.org/10.1016/j.eswa.2017.11.006.
Bach, N. X., Hai, N. D., & Phuong, T. M. (2016). Personalized recommendation of stories for commenting in forum-based social media. *Information Sciences, 352-353*, 48–60. Available at http://dx.doi.org/10.1016/j.ins.2016.03.006.
Balabanović, M., & Shoham, Y. (1997). Fab: Content-based, collaborative recommendation. *Communication of the ACM, 40*(3), 66–72. https://doi.org/10.1145/245108.245124.

Baltrunas, L., Ludwig, B., & Ricci, F. (2011). Matrix factorization techniques for context aware recommendation. In *Proceedings of the fifth ACM conference on recommender systems* (pp. 301–304). ACM. https://doi.org/10.1145/2043932.2043988.

Baltrunas, L., & Ricci, F. (2009). Context-based splitting of item ratings in collaborative filtering. In *Proceedings of the third ACM conference on recommender systems - RecSys '09* (May), 245. https://doi.org/10.1145/1639714.1639759.

Baltrunas, L., & Ricci, F. (2014). Experimental evaluation of context-dependent collaborative filtering using item splitting. *User Modeling and User-Adapted Interaction, 24*(1–2), 7–34. https://doi.org/10.1007/s11257-012-9137-9.

Burke, R. (2002). Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction, 12*(4), 331–370. https://doi.org/10.1023/A:1021240730564.

Cai, L., Xu, J., Liu, J., & Pei, T. (2018). Integrating spatial and temporal contexts into a factorization model for POI recommendation. *International Journal of Geographical Information Science, 32*(3), 524–546. https://doi.org/10.1080/13658816.2017.1400550.

Deladiennee, L., & Naudet, Y. (2017). A graph-based semantic recommender system for a reflective and personalised museum visit: Extended abstract. In *Proceedings - 12th international workshop on semantic and social media adaptation and personalization, SMAP 2017* (pp. 88–89). https://doi.org/10.1109/SMAP.2017.8022674.

Fan, X., Hu, Y., Zheng, Z., Wang, Y., Brezillon, P., & Chen, W. (2017). CASR-TSE: context-aware web services recommendation for modeling weighted temporal-spatial effectiveness. *IEEE Transactions on Services Computing, 1*. 1–1. https://doi.org/10.1109/TSC.2017.2782793.

Fouss, F., Pirotte, A., Renders, J.-M., & Saerens, M. (2007). Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *IEEE Transactions on Knowledge and Data Engineering, 19*(3), 355–369. https://doi.org/10.1109/TKDE.2007.46.

Garcia-de-Prado, A., Ortiz, G., & Boubeta-Puig, J. (2017). COLLECT: COLLaborativE ConText-aware service oriented architecture for intelligent decision-making in the Internet of Things. *Expert Systems with Applications, 85*, 231–248. https://doi.org/10.1016/j.eswa.2017.05.034.

Goldberg, D., Nichols, D., Oki, B. M., & Terry, D. (1992). Using collaborative filtering to weave an information tapestry. *Communications of the ACM, 35*(12), 61–70. https://doi.org/10.1145/138859.138867.

Hidasi, B., Karatzoglou, A., Baltrunas, L., & Tikk, D. (2016). Session-based recommendations with recurrent neural networks. *International conference on learning representations*.

Hidasi, B., & Tikk, D. (2016). General factorization framework for context-aware recommendations. *Data Mining Knowledge Discovery, 30*(2), 342–371. https://doi.org/10.1007/s10618-015-0417-y.

Huang, Z., Chen, H., & Zeng, D. (2004). Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering. *ACM Transactions on Information Systems, 22*(1), 116–142. https://doi.org/10.1145/963770.963775.

Karatzoglou, A., Amatriain, X., Baltrunas, L., & Oliver, N. (2010). Multiverse recommendation: N-dimensional tensor factorization for context-aware collaborative filtering. In *Proceedings of the fourth ACM conference on recommender systems* (pp. 79–86). ACM. https://doi.org/10.1145/1864708.1864727.

Kefalas, P., & Manolopoulos, Y. (2017). A time-aware spatio-textual recommender system. *Expert Systems with Applications, 78*, 396–406. https://doi.org/https://doi.org/10.1016/j.eswa.2017.01.060.

Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer, 42*(8), 30–37. https://doi.org/10.1109/MC.2009.263.

Li, X., & Chen, H. (2013). Recommendation as link prediction in bipartite graphs: A graph kernel-based machine learning approach. *Decision Support Systems, 54*(2), 880–890. https://doi.org/https://doi.org/10.1016/j.dss.2012.09.019.

Macedo, A. Q., Marinho, L. B., & Santos, R. L. T. (2015). Context-aware event recommendation in event-based social networks. In *Proceedings of the 9th ACM conference on recommender systems* (pp. 123–130). ACM. https://doi.org/10.1145/2792838.2800187.

Ning, X., & Karypis, G. (2011). SLIM: Sparse linear methods for top-N recommender systems. In *Proceedings of the 2011 IEEE 11th international conference on data mining* (pp. 497–506). IEEE Computer Society. https://doi.org/10.1109/ICDM.2011.134.

Odic, A., Tkalcic, M., Tasic, J. F., & Kosir, A. (2013). Predicting and detecting the relevant contextual information in a movie-recommender system. *Interacting with Computers, 25*, 74–90.

Panniello, U., Gorgoglione, M., & Tuzhilin, A. (2016). Research note—In CARSs we trust: how context-aware recommendations affect customers' trust and other business performance measures of recommender systems. *Information Systems Research, 27*(1), 182–196. https://doi.org/10.1287/isre.2015.0610.

Panniello, U., Tuzhilin, A., & Gorgoglione, M. (2014). Comparing context-aware recommender systems in terms of accuracy and diversity. *User Modeling and User-Adapted Interaction, 24*(1–2), 35–65. https://doi.org/10.1007/s11257-012-9135-y.

Pham, C., Diep, N. N., & Phuong, T. M. (2013). A wearable sensor based approach to real-time fall detection and fine-grained activity recognition. *Journal of Mobile Multimedia, 9*(1&2), 15–26. http://www.rintonpress.com/journals/jmm/abstractsJmm9-12.html.

Phuong, N. D., Thang, L. Q., & Phuong, T. M. (2008). A graph-based method for combining collaborative and content-based filtering. In T. B. Ho, & Z.-H. Zhou (Eds.). In *Proceedings of PRICAI 2008: 5351* (pp. 859–869). Springer. https://doi.org/10.1007/978-3-540-89197-0_80.

Phuong, T. M., Thanh, T. C., & Bach, N. X. (2018). Combining user-based and session-based recommendations with recurrent neural networks. In L. Cheng, A. C.-S. Leung, & S. Ozawa (Eds.), *Proceedings of neural information processing - 25th international conference, {ICONIP} 2018, Part {I}* (pp. 487–498). Springer. https://doi.org/10.1007/978-3-030-04167-0_44.

Rendle, S., Gantner, Z., Freudenthaler, C., & Schmidt-Thieme, L. (2011). Fast context-aware recommendations with factorization machines. In *Proceedings of the 34th international ACM SIGIR conference on research and development in information retrieval* (pp. 635–644). ACM. https://doi.org/10.1145/2009916.2010002.

Sulthana, A. R., & Ramasamy, S. (2018). Ontology and context based recommendation system using neuro-fuzzy classification. *Computers & Electrical Engineering*. https://doi.org/https://doi.org/10.1016/j.compeleceng.2018.01.034.

Sundermann, C. V., Domingues, M. A., Da Silva Conrado, M., & Rezende, S. O. (2016). Privileged contextual information for context-aware recommender systems. *Expert Systems with Applications, 57*, 139–158. https://doi.org/10.1016/j.eswa.2016.03.036.

Tang, K., Chen, S., & Khattak, A. J. (2018). Personalized travel time estimation for urban road networks: A tensor-based context-aware approach. *Expert Systems with Applications, 103*, 118–132. https://doi.org/10.1016/j.eswa.2018.02.033.

Tuan, T. X., & Phuong, T. M. (2017). 3D convolutional networks for session-based recommendation with content features. In *Proceedings of the eleventh ACM conference on recommender systems* (pp. 138–146). ACM. https://doi.org/10.1145/3109859.3109900.

Unger, M., Bar, A., Shapira, B., & Rokach, L. (2016). Towards latent context-aware recommendation systems. *Knowledge-Based Systems, 104*, 165–178. https://doi.org/https://doi.org/10.1016/j.knosys.2016.04.020.

Uslu, G., & Baydere, S. (2015). RAM: Real time activity monitoring with feature extractive training. *Expert Systems with Applications, 42*(21), 8052–8063. https://doi.org/10.1016/j.eswa.2015.06.017.

Wang, S. L., & Wu, C. Y. (2011). Application of context-aware and personalized recommendation to implement an adaptive ubiquitous learning system. *Expert Systems with Applications, 38*(9), 10831–10838. https://doi.org/10.1016/j.eswa.2011.02.083.

Weston, J., Elisseeff, A., Zhou, D., Leslie, C. S., & Noble, W. S. (2004). Protein ranking: From local to global structure in the protein similarity network. *Proceedings of the National Academy of Sciences, 101*(17), 6559–6563. https://doi.org/10.1073/pnas.0308067101.

Yang, K., & Toni, L. *Graph-based recommendation system* Retrieved from http://arxiv.org/abs/1808.00004.

Yin, H., & Cui, B. (2016). *Spatio-temporal recommendation in social media* (1st ed). Springer Publishing Company, Incorporated.

Yu, C.-A., Chan, T.-S., & Yang, Y.-H. (2017). Low-rank matrix completion over finite Abelian group algebras for context-aware recommendation. In *Proceedings of the 2017 ACM on conference on information and knowledge management* (pp. 2415–2418). ACM. https://doi.org/10.1145/3132847.3133057.

Zheng, Y., Burke, R., & Mobasher, B. (2014). Splitting approaches for context-aware recommendation: An empirical study. In *Proceedings of the 29th annual ACM symposium on applied computing* (pp. 274–279). ACM. https://doi.org/10.1145/2554850.2554989.

Zheng, Y., Mobasher, B., & Burke, R. (2014). CSLIM: Contextual SLIM recommendation algorithms. In *Proceedings of the 8th ACM conference on recommender systems* (pp. 301–304). ACM. https://doi.org/10.1145/2645710.2645756.

Zheng, Y., Mobasher, B., & Burke, R. (2015). CARSKit: A java-based context-aware recommendation engine. In *Proceedings of the 2015 IEEE international conference on data mining workshop (ICDMW)* (pp. 1668–1671). IEEE Computer Society. https://doi.org/10.1109/ICDMW.2015.222.

Zou, B., Li, C., Tan, L., & Chen, H. (2015). GPUTENSOR: Efficient tensor factorization for context-aware recommendations. *Information Sciences, 299*, 159–177. https://doi.org/https://doi.org/10.1016/j.ins.2014.12.004.