Personalized Recommendation on Multi-Layer Context Graph

Weilong Yao[1], Jing He[2], Guangyan Huang[2], Jie Cao[3], and Yanchun Zhang[1,2]

[1] University of Chinese Academy of Science, Beijing, China
yaoweilong12@mails.ucas.ac.cn
[2] Centre for Applied Informatics, Victoria University, Melbourne, Australia
{Jing.He,Guangyan.Huang,Yanchun.Zhang}@vu.edu.au
[3] Nanjing University of Finance and Economics, Nanjing, China
caojie690929@163.com

Abstract. Recommender systems have been successfully dealing with the problem of information overload. A considerable amount of research has been conducted on recommender systems, but most existing approaches only focus on user and item dimensions and neglect any additional contextual information, such as time and location. In this paper, we propose a Multi-Layer Context Graph (MLCG) model which incorporates a variety of contextual information into a recommendation process and models the interactions between users and items for better recommendation. Moreover, we provide a new ranking algorithm based on Personalized PageRank for recommendation in MLCG, which captures users' preferences and current situations. The experiments on two real-world datasets demonstrate the effectiveness of our approach.

Keywords: recommender system, context, random walk, graph.

1 Introduction

Recommender systems have been successfully dealing with the problem of information overload. While a substantial amount of research has already been conducted by both industry and academia in the area of recommender systems, most approaches only recommend items to users without taking into account any additional contextual information, such as time, location, age, or genre. Thus, the conventional recommender systems operate in the two-dimensional $USER \times ITEM$ space [1].

Incorporating contextual information into a recommendation process can achieve better recommendation accuracy than considering only user and item information. For example, a vacation recommendation for a given user may depend on season, age and interest. More specifically, in summer, an elderly user would probably prefer to enjoy her/his vacation on a peaceful beach rather than a ski resort.

Generally speaking, we consider three types of context in recommender systems: 1) User context describes user's meta attributes, such as gender, age, education and social information. Users sharing more common user context tend to

X. Lin et al. (Eds.): WISE 2013, Part I, LNCS 8180, pp. 135–148, 2013.

have similar tastes or preferences. 2) item context enables measuring relativity between two items. 3) The third type of contextual information we call decision context, involves context where the decision is made, such as time, location or mood. The same user with different decision context shows different preferences. For instance, the style of songs that a user listens at home on weekends may be different from the style of songs he listens in office on weekdays. Therefore, putting users and items in context can be beneficial to higher accuracy. However, there have been few methods which can incorporate all the three types of context into a recommendation process, and most approaches focus on only partial context (see Section 2).

In this paper, we propose a Multi-Layer Context Graph (MLCG) model which considers all of the three types of context into recommender systems, and models the interactions between users and items in the corresponding decision context. A personalized random walk-based ranking method on a MLCG is further presented, which captures users' preferences and decision context.

The most related work that also incorporates contextual information on graph by utilizing nodes to represent multidimensional data is proposed in [4]. This work strengthens the connections between users and items by merging different dimensions of context. The difference between MLCG and that work lies in that our method focuses on the instant situation where users interacts with items. That is, MLCG emphasizes the fact that interactions between users and items occur in a certain context.

To summarize, our main contributions are as follows:

1. We propose a Multi-Layer Context Graph (MLCG) model that incorporates all the three types of context to model the decision making by users. In particular, MLCG emphasizes the instant situation of the interactions between users and items.
2. Based on MLCG, we provide a new ranking algorithm, which extends Personalized PageRank for increasing accuracy of top-k recommendation through running on a MLCG that models the influence flow between/in layers.
3. Our experiments based on two real-world datasets demonstrate the effectiveness of our proposed method.

The rest of this paper is organized as follows. In Section 2, we cover related work on context-aware recommendation and graph-based recommendation. In Section 3, we define the problem of context-based recommendation. In Section 4, we present a MLCG construction algorithm and a recommendation method based on MLCG. In Section 5, we compare our method with counterparts on real-world datasets. Section 6 concludes this paper.

2 Related Work

In this section, we present a survey of work on context-aware recommendation and graph-based recommendation.

2.1 Context-Aware Recommendation

Early work in context-aware recommender utilized contextual information for pre-processing or post-processing. Recent work has focused on integrating contextual information with the user-item relations.

In [2], they use contextual information about the user's task to improve the recommendation accuracy. However, this method operates only within the traditional 2D $USER \times ITEM$ space. In [3], a reduction-based pre-filtering approach is proposed, which uses the user's prior item preferences to help match the current context for recommending items.

In [5], a regression-based latent factor model is proposed to incorporate features and past interactions. In [6], a context-aware factorization machine is provided to simultaneously take context into account to enhance predictions. The approach in [7] is based on tensor factorization. However, these methods are infeasible for scenarios with only implicit feedback data.

2.2 Graph-Based Recommendation

Recommendation on a graph is comprised of two steps: constructing a graph from training data and ranking item nodes for a given user.

In [8], a two-layer graph model is proposed for book recommendation based on similarity among user nodes or item nodes. In [9], a graph is constructed by connecting two item nodes rated by at least one user, then the node scoring algorithm, ItemRank, is executed to rank item nodes according to the active user's preference records. In [10], several Markov-chain model based quantities are considered, which provide similarities between any pair of nodes on a bipartite graph for recommendation. However, all the above approaches consider only $USER$ and $ITEM$ dimensions without additional contextual information.

In [11], ContextWalk algorithm is provided for movie recommendation, which uses original random walk on a graph by considering the user and item context (e.g., actor, director, genre), but it ignores the decision context where a user chooses to watch a movie. As an example in [1], a user may have significantly different preferences on the genre of movies she/he wants to see when she/he is going out to a theater with her/his boyfriend/girlfriend as opposed to going with her/his parents. In [12], a graph-based method is presented that aims to improve recommendation quality by modeling user's long-term and short-term preferences. This method can deal with time information, but can not incorporate other types of contextual information, such as location and mood. In [4], a bipartite graph is proposed to model the interactions between users and items based on a recommendation factor set, F, where each recommendation factor $f \in F$ is defined as a combination of multidimensional data. Nodes corresponding to recommendation factors are connected with item nodes. That is, the method utilizes duplicable dimensions to enhance the interactions, which may bring noises into recommendation. In addition, they do not provide principles to define recommendation factor set F, which is crucial to recommendation performance.

3 Problem Formulation

We define the context-based recommendation as follows. In recommender systems, we have a set of users $U = \{u_k|k = 1, 2, ..., |U|\}$ and a set of items $I = \{i_k|k = 1, 2, ..., |I|\}$. Formally, let $C_U = \{C_{Uk}|k = 1, 2, ..., |C_U|\}$ be the context of users, where C_{Uk} is a domain of user context (e.g., AGE, GENDER), let $C_I = \{C_{Ik}|k = 1, 2, ..., |C_I|\}$ be the context of items, where C_{Ik} is a domain of item context (e.g., GENRE, ARTIST), let $C_D = \{C_{Dk}|k = 1, 2, ..., |C_D|\}$ be the context where decisions are made, where C_{Dk} is a domain of decision context (e.g., TIME, LOCATION). A recommendation task can be expressed as: given a user $u \in U$ with user context $c_u = \{c_{uk}|c_{uk} \in C_{Uk}, k = 1, 2, ..., |C_U|\}$ in a particular decision context $c_d = \{c_{dk}|c_{dk} \in C_{Dk}, k = 1, 2, ..., |C_D|\}$, a ranked list of items $R(u, c_d) \subseteq I$ is provided as the potential items ranked by relevance scoring function *utility*, where *utility* is defined as $utility(u, c_u, c_d, i)$ to measure the relevance between tuple $< u, c_u, c_d >$ and item $i \in I$. For example, recommending songs for a user, Ted, when he is at *home* on *Saturday night* can be interpreted as finding the songs with top-K relevance with tuple $< u = Ted, c_u : \{Gen. = M\},\ c_d : \{DayofWeek = Sat., Time = Night, Loc. = Home\} >$. For simplicity, we assume that the contextual information is denoted by categorical values.

4 Proposed Method

In this section, we start by presenting the MLCG construction algorithm, then provide a novel random walk-based ranking method based on MLCG.

4.1 Graph Construction

Construction Algorithm. We first describe the data format. We consider users' attributes as user context, and attributes of items as item context. Hence in Table 1, C_U is $\{GEN. : \{M\}, AGE : \{[18 - 30]\}\}$, and in Table 2, C_I is $\{ARTIST : \{M.J.\}, GENRE : \{Rock\}\}$. Meanwhile, log data in Table 3 describes the song a user listened to in a certain situation. In graph-based methods, each item or user is represented as a node. Most graph-based methods describe the interaction between a user and an item by directly creating an edge between the two nodes [4, 8, 10–13]. However, these methods ignore the effect of decision context, while we argue that, in a recommender system, it is in a certain decision context where users interact with items. That is, users' preferences on items should flow though particular decision context before reaching the item nodes.

Therefore, we design a new type of node, decision node $node_{decision} =< u, c_{d1}, c_{d2}, ..., c_{d|C_D|} >$, where $u \in U$ and $c_{dk} \in C_{Dk}$, to characterize the decision context and model the interactions between users and items. Note that $node_{decision}$ is a combined node with a user and a decision context. The underlying intuition of $node_{decision}$ is that decision context is a local effect and should not be shared by all users as a global effect. That is, the same decision context for

Table 1. Example of User Data

USER	GENDER	AGE
Ted	M	[18-30]
Mike	M	[18-30]

Table 2. Example of Item Data

SONG	ARTIST	GENRE
Beat It	Michael Jackson	Rock
Rock with you	Michael Jackson	Rock

Table 3. Example of Log Data

USER	DayofWeek	LOCATION	SONG
Ted	Saturday	Home	Beat it
Mike	Sunday	Office	Rock with it
Mike	Sunday	Office	Beat it
Ted	Sunday	Office	Beat it

different user have different impacts on decision making. The proposed MLCG construction algorithm is outlined in Algorithm 1.

As the Algorithm 1 shows, a MLCG is a three-layer graph that consists of a user context layer, an item context layer and a decision context layer. Figure 1 illustrates an example of MLCG constructed from Tables 1−3. The number on the edge represents the co-occurrence of two end-nodes.

Only one type of entity node is denoted as a square node in Figure 1, on each layer, such as $node_{decision}$ on decision context layer. Entity nodes are characterized by their own context nodes on the same layer. For every context/attribute of an entity, there is a corresponding edge between the entity node and the context node. As Figure 1 shows, node M describes the gender of the user nodes connected with it. Furthermore, the nodes of the same entity type interact through their context nodes. That is, influence of an entity node propagates to another entity through sharing context nodes. In the case of Figure 1, the more rock songs a user listened to, the more of the user's preferences flow to other rock songs through the node $ROCK$.

In addition to the intra-layer edges, inter-layer edges are also available to represent the interactions between layers. In our model, user nodes do not directly interact with items because we emphasize that interactions should occur in a certain situation. In this sense, the interaction is expressed as: an edge from a user node on a user layer to the corresponding $node_{decision}$ and the other edge from the $node_{decision}$ to a song node on an item layer. For an active user, songs which were listened to in the same context are connected to the same corresponding

Algorithm 1. Construct a Multi-Layer Context Graph

Input: Set of users U and context C_U; Set of Items I and context C_I; Decision Context C_D; Log record table $LogTable$, where each log is in the form of $< u, c_{d1}, c_{d2}, ..., c_{d|C_D|}, i >$, $u \in U$, $i \in I$ and $c_{dk} \in C_{Dk}$

Output: Multi-Layer Context Graph $\mathcal{G}$

Initialize a graph $\mathcal{G}$ with $USER\text{-}Layer$, $ITEM\text{-}Layer$ and $Decision\text{-}Layer$
$CreateLayer(C_U, U, USER\text{-}Layer)$
$CreateLayer(I_U, I, ITEM\text{-}Layer)$
for each context domain $c_d \in C_D$ **do**
    for each context value $v \in c_d$ **do**
        Create a decision context node for v on $Decision\text{-}Layer$
    end for
end for
for each log record $log \in LogTable$ **do**
    Create a $node_{decision}$ $v =< u, c_{d1}, c_{d2}, ..., c_{d|C_D|} >$ on Decision Layer
    Connect user node u and v
    Connect $c_{d1}, c_{d2}, ..., c_{d|C_D|}$ nodes and v
    Connect item node i node and v
end for
Return $\mathcal{G}$

Subroutine CreateLayer$(C, T, Layer)$
for each context domain $c \in C$ **do**
    for each context value $v \in c$ **do**
        Create a context node for v on Layer
    end for
end for
for each entity $t \in T$ **do**
    Create a node for t on $Layer$
    Connect node t and its corresponding context nodes
end for

$node_{decision}$ node. In this way, the effect of current context is distributed precisely over these songs listened to in that context.

Weight Assignment. Most graph-based recommendation methods consider a recommendation process as a node ranking task on a graph, hence several random walk-based ranking measures are proposed [4, 9, 12, 13]. However, these ranking methods are performed on a homogeneous graph, ignoring the different types of edges, so they will not work for MLCG. By fusing different edge types, we transform the heterogeneous multi-layer graph to a homogeneous graph.

We denote $N(j)$ as a set of nodes connected with node j, denote $N_s(j) \subseteq N(j)$ as a set of nodes on the same layer with node j, and $N_d(j) = N(j) - N_s(j)$. Given node j and $k \in N_s(j)$ on any layer, the edge weight $w(j, k)$ is defined as follows:

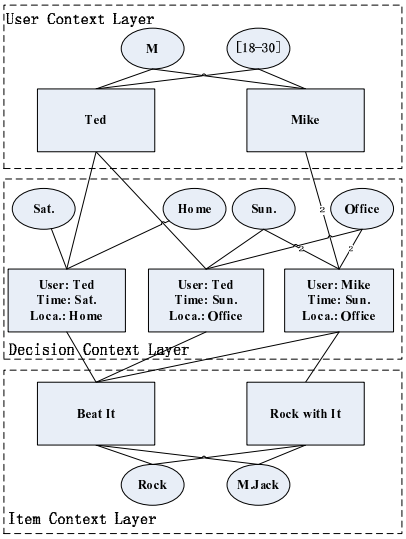


Fig. 1. An example: the MLCG constructed based on Tables 1−3

$$w(j,k) = \begin{cases} \frac{\alpha}{|T(N_s(j))|} \frac{f(j,k)}{\sum_{t \in N_s(j)} f(j,t)} & \text{if } N_d(j) > 0 \\ \frac{co\text{-}occu(j,k)}{\sum_{t \in N_s(j)} co\text{-}occu(j,t)} & \text{if } N_d(j) = 0 \\ 0 & otherwise, \end{cases} \tag{1}$$

where function $f(j,k)$ denotes the importance score for node k with regard to node j, and $co\text{-}occu(j,k)$ is the co-occurrence of node j and node k. As an example in Figure 1, the user, Ted, completed two interactions in the same decision context where it is Sunday, hence the co-occurrence of node $Sunday$ and $node_{decision}$ is 2. $T(N_s(j))$ is a set of node types of nodes in $N_s(j)$.

Meanwhile, $f(j,k)$ should satisfy the following intuition-based criteria: 1) rare context/attributes are likely to be more important, whereas common context/attributes are less important. For example, thousands of people like football but only Ted and $Mike$ like PingPong, in which case node $PingPong$ carries more benefit for looking for similar user than node $Football$. 2) the more co-occurrence of two nodes, the more related they are. For example, for a user who is inclined to listen to songs at home, his preferences propagate mainly through node $Home$ to other songs. Considering the two intuitive rules, we borrow the idea of TF-IDF from information retrieval area and define $f(j,k)$ as follows.

$$f(j,k) = co\text{-}occu(j,k) \log \frac{\sum_{v \in \Omega(k)} \sum_{t \in N_s(v)} co\text{-}occu(v,t)}{\sum_{t \in N_s(k)} co\text{-}occu(k,t)}, \tag{2}$$

where $\Omega(k)$ is a set of nodes having the same node type with node k. For example, $\Omega(Saturday) = \{Saturday, Sunday\}$ since they share a node type $DayofWeek$.

Given node j and $k \in N_d(j)$, the edge weight $w(j,k)$ is calculated as:

$$w(j,k) = \begin{cases} \frac{(1-\alpha)co\text{-}occu(j,k)}{\sum_{t \in N_d(j)} co\text{-}occu(j,t)} & \text{if } N_s(j) > 0 \\ \frac{co\text{-}occu(j,k)}{\sum_{t \in N_d(j)} co\text{-}occu(j,t)} & \text{if } N_s(j) = 0 \\ 0 & otherwise. \end{cases} \tag{3}$$

Here, α controls the trade-off between intra-layer and inter-layer interactions. The larger α is, the more influence flow to nodes on the same layer, and the more effect intra-layer interactions have.

4.2 Recommendation on MLCG

Here, we consider the recommendation task as a ranking task on a graph. We then extend the Personalized PageRank [15], which is a variation of the PageRank algorithm[14]. The original PageRank score of a node is calculated as follows:

$$PR(k+1) = \alpha \cdot M \cdot PR(k) + (1-\alpha) \cdot d, \tag{4}$$

where $PR(k)$ denotes the rank value at the k-th iteration, M is a transition probability matrix, α is a damping factor that is normally given as 0.85 and d is a vector defined as $d_k = \frac{1}{n}, k = 1, 2, ..., n$, where n is the number of nodes.

As Equation 4 shows, PageRank can be considered as a Markov process with restart, and the probability of a random walker will jump to a node after a restart is equal to others. That is, the PageRank algorithm considers all nodes equally without biasing any important nodes. In [15], a topic-sensitive PageRank is proposed to introduce a personalized vector to bias users' preferences. More specifically, vector d is built as a user-specific personalized vector, where $d_k = 1$ if k-th node represents the active user, otherwise $d_k = 0$. Then the Personalized PageRank is calculated in Equation 4.

By using a hint from [9] and [12], we extend the personalized PageRank which captures both users' preferences and current decision context. For a given user $u \in U$ and current decision context $c_d = \{c_{dk} | c_{dk} \in C_{Dk}, k = 1, 2, ..., |C_D|\}$, we define $\varepsilon = \{i_k | i_k \in I, k = 1, 2, ..., |\varepsilon|\}$ as a set of items that u accessed before, then we construct $\tilde{d}$ as follows:

$$\tilde{d_j} = \begin{cases} \frac{\lambda}{|\varepsilon|} & \text{if node } j \in \varepsilon \\ \frac{1-\lambda}{|c_d|+1} & \text{if node } j \in c_d \text{ or node } j = u \\ 0 & otherwise, \end{cases} \tag{5}$$

where λ adjusts the radio of bias between users' preferences and current decision context. d is $\tilde{d}$ after L-1 normalization. Meanwhile, we consider the recommendation process as a multi-path random walk process with multiple starting points. Hence $PR(0)$ is defined as follows.

$$PR(0)_j = \begin{cases} 1 & \text{if node } j \in c_d \text{ or node } j = u \\ 0 & otherwise. \end{cases} \quad (6)$$

Then, we normalize vector $PR(0)$ to ensure that the sum of its non-negative elements is 1.

So, the extended personalized PageRank is summarized as: construct d for the active user based on Eq. 5, then run Eq. 4 until convergence with initializing $PR(0)$ based on Eq. 6. In order to make recommendations, we rank item nodes based on their PR value and keep only the top-K item nodes as recommendations.

5 Experiment

In this section, we present an experimental study which is conducted on two real-world datasets to demonstrate the effectiveness of our approach.

5.1 Dataset

We use two implicit datasets for our experiments instead of explicit rating data such as MovieLens[1], since we aim at improving the top-K recommendation other than explicit rating estimation.

The first dataset is Last.fm[2] which contains 19,150,868 music listening logs of 992 users (till May, 4th 2009). We extract the logs from April to May and remove these songs which were listened to less than 10 times, then the final dataset contains 992 users, 12,286 songs and 264,446 logs. In this case, the user context only includes domains of COUNTRY and AGE, while item context includes a domain of ARTIST. We notice that the original log tuples only consist of USER, SONG and TIMESTAMP domains. By transforming TIMESTAMP into different temporal features, we obtain several decision context domains, such as *Day of Week* and *Time Slice* (i.e., each time slice lasts for 6 hours). Finally, we use logs in April as a training set, and randomly choose 1000 logs from 1st to 4th May 2009 as a test set.

The second dataset is CiteULike[3] which provides logs on *who* posted *what* and *when* the posting occurred. By removing users who posted less than 5 papers and papers which were posted by less than 5 users, we obtain a subset of original dataset which contains 1,299 users, 5,856 items and 40,067 user-items pairs from January to May 2007. In this case, there is no user context because of the lack of user information. Similarly, we transform TIMESTAMP into several temporal features, and TAGs of papers are considered as item context. Finally, we use logs in the two final weeks as a test set, and others as a training set.

[1] `http://www.movielen.umn.edu`
[2] `http://www.last.fm.com`
[3] `http://www.citeulike.org`

5.2 Evaluation Metrics

We use HitRatio@K [16], Mean Reciprocal Rank (MRR@K) [17] and Recall@K to evaluate the performance of our top-K recommendation.

Given a test case $< u, c_d, i >$ in test set $TEST$, where a user u accessed an item i in decision context c_d, the recommendation method generates a ranked list of items $R(u, c_d)$, where $|R(u, c_d)| = K$. Then HitRatio@K is defined as follows:

$$HitRatio@K = \frac{1}{|TEST|} \sum_{<u,c_d,i>} I(i \in R(u, c_d)), \tag{7}$$

where $I(\cdot)$ is an indicator function.

In addition, we also measured the MRR@K for evaluating the rank of the target item i:

$$MRR@K = \frac{1}{|TEST|} \sum_{<u,c_d,i>} \frac{1}{rank(i)}, \tag{8}$$

where $rank(i)$ refers to the rank of target item i in $R(u, c_d)$.

Furthermore, we use Recall@K to evaluate the overall relevancy performance of recommendation methods:

$$Recall@K = \frac{1}{|TEST|} \sum_{<u,c_d>} \frac{|T(u, c_d) \cap R(u, c_d)|}{|T(u, c_d)|}, \tag{9}$$

where $T(u, c_d)$ is the set of items the user u accessed in context c_d.

5.3 Baseline Methods

We evaluate the effectiveness of our method through comparing it with other existing methods:

Frequency based (FreMax): A user independent method, which ranks the items by the times they were accessed. That is, FreMax generates a same list of items for any user.

User-based CF (UserCF): A N-neighbor user-based collaborative filtering method, which uses Pearson Correlation as the user similarity measurement. The optimal value of N is 10 in experiments on CiteULike dataset, and N is 1 on Last.fm dataset.

ItemRank [9]: A random walk-based item scoring algorithm, which is performed on an item graph. The item graph is constructed by connecting two items if they were rated by at least one user.

GFREC [4]: A contextual bipartite graph-based method, which defines a recommendation factor set F, to transform a given log table into a bipartite graph. In our experiments, we use one of the best settings of F according to [4].

We do not consider item-based CF, since the number of items is much greater than users, and user-based CF methods can provide more accurate recommendation. In addition, as [18] indicates, SVD methods only achieve slight improvement on implicit feedback datasets, thus, we do not compare our method with

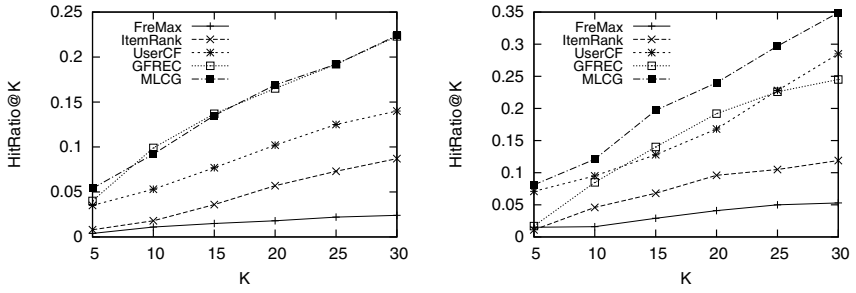
SVD-based methods. Since UserCF and ItemRank operate in a $USER \times ITEM$ space, we transform our training data into a pseudo rating matrix by considering the normalized access count of a user on an item as the user's pseudo rating on the item.

5.4 Experimental Results

In this subsection, we present our experimental results in two scenarios. Note that for Last.fm dataset, our experiments provide a ranked list of songs, which might have been listened to before by the given user, since users generally listen to the same song more than once. For the experiments on CiteULike dataset, we recommend papers that have not been posted by the active user.

Here, λ in each experiment is set to 0.5. α in each layer is a tunable parameter. For Last.fm dataset, $\alpha_{user} = 0.005$, $\alpha_{decision} = 0.2$ and $\alpha_{item} = 0.01$. For CiteULike, $\alpha_{user} = 0.0$ (user context is unavailable), $\alpha_{decision} = 0.05$ and $\alpha_{item} = 0.01$.

HitRatio@K Analysis. Figure 2 illustrates the HitRatio@K of our experiments on the two datasets. For Last.fm, the methods with contextual information (i.e., GFREC and MLCG) show substantial improvement over FreMax, ItemRank and UserCF, which confirms the importance of context for recommendation. FreMax shows the lowest HitRatio, revealing the fact that users have their own preferences on songs, and would not be affected by popularity. The HitRatio of MLCG is 54.3%(K=5)-75.3%(K=15) greater than that of UserCF. Moreover, there is a gain of 35.0% over GFREC from MLCG in top-5 recommendation. Generally, on Last.fm MLCG shows close/comparable performance with GFREC over the metrics. For CiteULike, it is clear that our method significantly outperforms counterpart methods. Similar to the case on Last.fm, contextual information contributes to better performance. Among the context-based methods, MLCG achieves perceptible improvement, where HitRatio of MLCG is 3.8 times (K=5) and 42.4% (K=15) higher than that of GFREC.

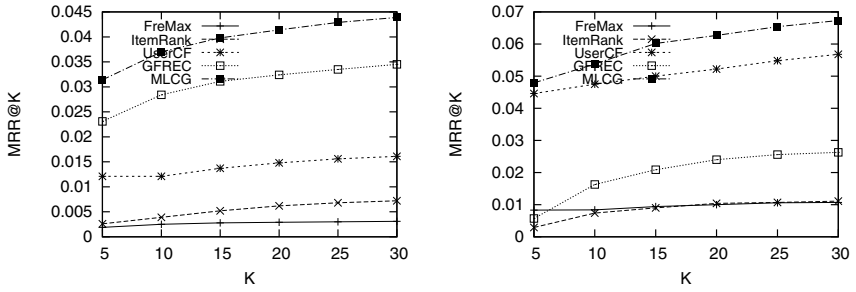


(a) Last.fm (b) CiteULike

Fig. 2. Comparing the HitRatio@K of MLCG against Baseline Approaches

It should be noticed that Top-K recommender systems benefit more from higher accuracy when K is small, such as user experience.

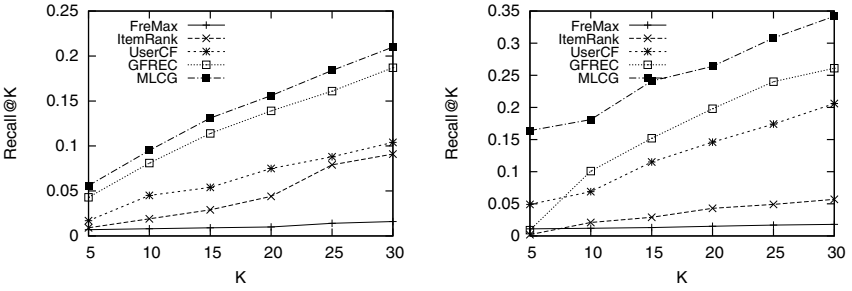
MRR Analysis. As shown in Figure 3, MLCG significantly outperforms other methods in MRR, that is, MLCG generates more accurate recommendations. For Last.fm, MLCG achieves a performance gain over GFREC by 35.9% (MRR@5). We notice that in CiteULike, the MRR of GFREC is lower than that of UserCF. The reason is that superfluous combinations of multidimensional data bring connections, as well as noises. MLCG addresses this issue by grouping context into the three layers to explore their relationships, obtaining an improvement of MRR by up to 20.4% (K=15) over UserCF.



(a) Last.fm (b) CiteULike

Fig. 3. Comparing the MRR@K of MLCG against Baseline Approaches

Recall Analysis. We give the results of recall in Figure 4. It can be seen that by incorporating contextual information, MLCG and GFREC have higher recall than other methods. And the improvements of MLCG over the 4 comparison algorithms on both datasets are still clear. More specifically, while GFREC in general performs the best of the baseline methods, MLCG outperforms it with recall of up to 32.3% (K=5) and 17.2 times (K=5) greater. Higher recall indicates that the algorithm returns most of the relevant results based on the instant situation and the active user. In this sense, having a better understanding of the context of the active user, MLCG provides more personalized recommendations than other methods. In summary, our methods significantly outperforms several counterparts in terms of MRR@K and Recall@K, and it is comparable to or better than GFREC in HitRatio@K. In particular, MLCG achieves significant improvement of HitRatio over GFREC in the scenario of recommending previously unseen items (i.e., CiteULike), which is a more typical application. These observations demonstrate that context plays a vital role in improving recommendation performance, and the proposed MLCG is effective in blending various types of context for recommendation.



(a) Last.fm (b) CiteULike

Fig. 4. Comparing the Recall@K of MLCG against Baseline Approaches

6 Conclusion

In this paper, a Multi-Layer Context Graph (MLCG) model is proposed. MLCG utilizes contextual information to construct a layer for each type of context respectively, and models the decision making by users. In particular, our model emphasizes that users interact with items in an instant context. Furthermore, we take different edge types into consideration, distinguishing not only the intra-layer from inter-layer interactions but also various contextual domains, such as *Day of Week*. Based on MLCG, we extend Personalized PageRank to rank items in MLCG which captures users' preferences and current decision context. Finally, experiments based on two real-world datasets demonstrate that the effectiveness of the proposed method exceeds other existing ones in all evaluation metrics. This work can form a basis to introduce social relationships into the user context to improve accuracy further. For example, the similarity of interest in groups/communities can be used to improve recommendation diversity and accuracy.

Acknowledgement. This work is partially supported by the National Natural Science Foundation of China (Grant No. 61272480).

References

1. Adomavicius, G., Tuzhilin, A.: Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. IEEE Trans. on Knowl. and Data Eng. 17(6), 734–749 (2005)
2. Herlocker, J.L., Konstan, J.A.: Content-Independent Task-Focused Recommendation. IEEE Internet Computing 5(6), 40–47 (2001)
3. Adomavicius, G., Sankaranarayanan, R., Sen, S., Tuzhilin, A.: Incorporating contextual information in recommender systems using a multidimensional approach. ACM Transactions on Information Systems (TOIS) 23(1), 103–145 (2005)
4. Lee, S., Song, S.-I., Kahng, M., Lee, D., Lee, S.-G.: Random walk based entity ranking on graph for multidimensional recommendation. In: Proceedings of the Fifth ACM Conference on Recommender Systems, pp. 93–100 (2011)

5. Agrawal, D., Chen, B.: Regression-based latent factor models. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 19–28 (2009)
6. Rendle, S., Gantner, Z., Freudenthaler, C., Schmidt-Thieme, L.: Fast context-aware recommendations with factorization machines. In: Proceedings of the 34th International Conference on Reseach and Development in Information, pp. 635–644 (2011)
7. Xiong, L., Chen, X., Huang, T.-Y., Schneider, J., Carbonell, J.: Temporal collaborative filtering with bayesian probabilistic tensor factorization. In: Proceedings of SIAM International Conference on Data Mining, pp. 211–222 (2010)
8. Huang, Z., Chung, W., Ong, T.-H., Chen, H.: A graph-based recommender system for digital library. In: Proceedings of the 2nd ACM/IEEE-CS Joint Conference on Digital Libraries, pp. 65–73 (2002)
9. Gori, M., Pucci, A., Roma, V., Siena, I.: Itemrank: A random-walk based scoring algorithm for recommender engines. In: Proceedings of the 20th International Joint Conference on Artifical Intelligence, pp. 2766–2771 (2007)
10. Fouss, F., Pirotte, A., Renders, J.-M., Saerens, M.: 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 (2007)
11. Bogers, T.: Movie recommendation using random walks over the contextual graph. In: Proc. of the 2nd Intl. Workshop on Context-Aware Recommender Systems (2010)
12. Xiang, L., Yuan, Q., Zhao, S., Chen, L., Zhang, X., Yang, Q., Sun, J.: Temporal recommendation on graphs via long-and short-term preference fusion. In: Proceedings of the 16th International Conference on Knowledge Discovery and Data Mining, pp. 723–732 (2010)
13. Gori, M., Pucci, A.: Research paper recommender systems: A random-walk based approach. In: IEEE/WIC/ACM International Conference on Web Intelligence, pp. 778–781 (2006)
14. Page, L., Brin, S., Motwani, R., Winograd, T.: The PageRank Citation Ranking: Bringing Order to the Web. Technical Report (1999)
15. Haveliwala, T.H.: Topic-sensitive pagerank. In: Proceedings of the 11th International Conference on World Wide Web, pp. 517–526 (2002)
16. Karypis, G.: Evaluation of item-based top-n recommendation algorithms. In: Proceedings of the Tenth International Conference on Information and Knowledge Management, pp. 247–254 (2001)
17. Vallet, D., Cantador, I., Jose, J.M.: Personalizing web search with folksonomy-based user and document profiles. In: Gurrin, C., He, Y., Kazai, G., Kruschwitz, U., Little, S., Roelleke, T., Rüger, S., van Rijsbergen, K. (eds.) ECIR 2010. LNCS, vol. 5993, pp. 420–431. Springer, Heidelberg (2010)
18. Hu, Y., Koren, Y., Volinsky, C.: Collaborative filtering for implicit feedback datasets. In: Eighth IEEE International Conference on Data Mining, pp. 263–272 (2008)