Similarity-Based Context-Aware Recommendation

Yong Zheng(✉), Bamshad Mobasher, and Robin Burke

Center for Web Intelligence, School of Computing, DePaul University, 243 South Wabash Ave, Chicago, IL 60604, USA
{yzheng8,mobasher,rburke}@cs.depaul.edu

Abstract. Context-aware recommender systems (CARS) take context into consideration when modeling user preferences. There are two general ways to integrate context with recommendation: contextual filtering and contextual modeling. Currently, the most effective context-aware recommendation algorithms are based on a contextual modeling approach that estimate deviations in ratings across different contexts. In this paper, we propose context similarity as an alternative contextual modeling approach and examine different ways to represent context similarity and incorporate it into recommendation. More specifically, we show how context similarity can be integrated into the sparse linear method and matrix factorization algorithms. Our experimental results demonstrate that learning context similarity is a more effective approach to context-aware recommendation than modeling contextual rating deviations.

Keywords: Recommender system · Context · Context-aware · Matrix factorization

1 Introduction

Introducing contexts to recommender systems (RS) contributes to building new types of applications, such as context-aware recommendation [2] and context suggestions [21]. Context-aware recommender systems (CARS) have been a topic of considerable recent research interest in RS. Considering context is shown to be useful in a variety of recommendation domains including tourism [4,17], music [3], and restaurants [13]. In many application domains, including these, it is reasonable to assume that users' preferences may change from context to context, even for the same item. For example, a user may choose a different movie if he or she is going to see the movie with kids, rather than on a romantic date. In this case, the *companion* is the influential contextual variable.

The standard formulation of the recommendation problem begins with a two dimensional (2D) matrix of ratings, organized by user and item: *Users* × *Items* → *Ratings*. The key insight of context-aware recommender systems is that users' preferences for items may be a function of the context in which those items are

© Springer International Publishing Switzerland 2015
J. Wang et al. (Eds.): WISE 2015, Part I, LNCS 9418, pp. 431–447, 2015.
DOI: 10.1007/978-3-319-26190-4_29

encountered. Incorporating contexts requires that we estimate user preferences using a multidimensional rating function – R: *Users* $\times$ *Items* $\times$ *Contexts* $\rightarrow$ *Ratings* [2].

Contextual modeling is a popular general architecture for developing context-aware recommender systems. The most effective contextual modeling approaches, such as context-aware matrix factorization [5], try to adapt to contextual preferences by modeling contextual rating deviations: how do ratings for one context differ generally from those in another context. However, researchers ignore the reason for such deviations, namely that contexts may be inherently similar, and contextual similarity may be a better framework for understanding and representing contextual effects.

In this paper, we propose the notion of similarity-based contextual recommendation. The basic assumption behind this paradigm is that recommendation lists should be similar if their contextual situations are similar. Furthermore, we provide specific ways to model the context similarity and incorporate it into two recommendation models respectively: sparse linear method (SLIM) [12] and matrix factorization (MF) [10]. Our experimental results over multiple context-aware data sets demonstrate that similarity-based context-aware recommendation algorithms are able to outperform the ones based on modeling contextual rating deviations, offering a promising and novel way to further develop context-aware recommendation algorithms.

2 Related Work

Context is usually defined as any information that can be used to characterize the situation of an entity [1]. In CARS, the contextual variables are usually from the attributes of the activity itself [16], e.g., *time* and *companion* are two variables which can be considered as the attribute of the activity "watching a movie". Other attributes from users or items, such as user gender and movie genre, are usually deemed as contents in RS.

There are generally two ways to incorporate contexts into recommender systems – contexts can be used either as filters in the algorithms (i.e., *contextual filtering*), such as differential context modeling [17,18] and context-aware splitting [6,20], or as one part of the predictive functions in the recommendation process (i.e., *context modeling*).

There are usually two categories in context modeling: independent models and dependent models. For example, tensor factorization (TF) [9] directly considers contexts as additional dimensions in the multidimensional rating space, which is an independent model assuming contextual effects are not dependent with users or items. On the other hand, context-aware matrix factorization (CAMF) [5] and contextual sparse linear method (CSLIM) [22,23] are two dependent models which try to adapt to contextual preferences by modeling contextual rating deviations, where those deviations are usually dependent with either user or item dimension in the rating data set. Previous work [5,23] has demonstrated that dependent models usually outperform the independent models in most cases. Thus, much of the research in contextual modeling explores various ways to model context dependencies.

Current dependent models rely on the dependencies between contexts and user (or item) dimensions by modeling rating deviations in different contexts. We call these algorithms as *Deviation-Based* context-aware recommendation algorithms; However, such algorithms ignore the relationships between context themselves. Context similarity was explored in some previous work – it is calculated based on either semantics [8,11] or co-ratings in the same contexts [7]. Semantic context similarity is derived from the applications in natural language processing and information retrieval. It is usually useful in hierarchical semantic structure and not general enough to any cases, which limits its application, e.g., it is difficult to measure the similarity between *at home* and *at cinema* from semantics. Calculations based on co-ratings are usually unreliable, since the number of co-ratings in the same context is limited.

2.1 Matrix Factorization and Deviation-Based CAMF

Matrix factorization (MF) [10] is one of the most effective recommendation algorithms in the traditional, non-contextual, recommender systems. Simply, both users and items are represented by vectors, e.g., $\overrightarrow{p_u}$ is used to denote a user vector, and $\overrightarrow{q_i}$ as an item vector. As a result, the rating prediction can be described by Eq. 1.

$$\hat{r}_{ui} = \overrightarrow{p_u} \cdot \overrightarrow{q_i} \tag{1}$$

$$\hat{r}_{ui} = \mu + b_u + b_i + \overrightarrow{p_u} \cdot \overrightarrow{q_i} \tag{2}$$

More specifically, the values in the user or item vectors indicate the weights on K (e.g., $K = 5$) latent factors. The weights in $\overrightarrow{p_u}$ can be viewed as the degree to which those latent factors represent user's interests, and the weights in $\overrightarrow{q_i}$ represent how the specific item is associated with the latent factors. Therefore, the dot product of those two vectors can be used to indicate how much the user likes this item, where users' preferences on items are captured by the latent factors. In addition, user and item rating biases can be added to the prediction function, as shown in Eq. 2, where μ denotes the global average rating in the data, b_u and b_i represent the user and item bias respectively (Table 1).

Table 1. Contextual ratings on movies

User	Item	Rating	Time	Location	Companion
U1	T1	3	Weekend	Home	Alone
U1	T1	5	Weekend	Cinema	Girlfriend
U1	T1	?	Weekday	Home	Family

Assume there are one user $U1$, one item $T1$, and three contextual dimensions – Time (weekend or weekday), Location (at home or cinema) and Companion (alone, girlfriend, family) as shown in the table above. In the following discussion, we use *contextual dimension* to denote the contextual variable, e.g. "Location". The term *contextual condition* refers to a specific value in a dimension, e.g. "home" and "cinema" are two contextual conditions for "Location". A *context* or *contextual situation* is, therefore, a set of contextual conditions, e.g. {*weekend, home, family*}. More specifically, we use c_k and c_m to denote two different contextual situations. In other words, c_k is composed by a set of contextual conditions. Let $c_{k,l}$ denote the l^{th} contextual condition in the context c_k. For example, if $c_k = \{\text{weekend, home, alone}\}$, then $c_{k,2}$ is "home".

The rating prediction function in CAMF [5] can be shown in Eq. 3.

$$\hat{r}_{uic_{k,1}c_{k,2}\ldots c_{k,L}} = \mu + b_u + \sum_{j=1}^{L} B_{ijc_{k,j}} + \overrightarrow{p_u} \cdot \overrightarrow{q_i} \tag{3}$$

Assume there are L contextual dimensions in total, $c_k = \{c_{k,1}c_{k,2}...c_{k,L}\}$ is used to describe the contextual situation, where $c_{k,j}$ denotes the contextual condition in the j^{th} context dimension. Therefore, $B_{ijc_{k,j}}$ indicates the contextual rating deviation associated with item i and the contextual condition in the j^{th} dimension.

A comparison between Eqs. 2 and 3 reveals that CAMF simply replaces the item bias b_i by a contextual rating deviation term $\sum_{j=1}^{L} B_{ijc_{k,j}}$, and it assumes that the contextual rating deviation is dependent on items. Therefore, this approach is denoted as CAMF_CI. In a similar manner, the deviation can also be viewed as being dependent on users, which replaces b_u by the contextual rating deviation term and helps formulate the CAMF_CU. Finally, the CAMF_C variant of the algorithm assumes that the contextual rating deviation is independent of users and items.

As the typical optimization in matrix factorization, the parameters, such as the user and item vectors, user biases and rating deviations, can be learned by the stochastic gradient descent (SGD) method to minimize the squared rating prediction errors. In early work [5], CAMF was demonstrated to outperform the independent contextual modeling approaches, such as the tensor factorization [9].

2.2 Sparse Linear Method and Deviation-Based CSLIM

Sparse linear method (SLIM) shown in Fig. 1 is an approach designed for top-N recommendations in traditional RS. It improves upon the traditional item-based K-nearest neighbor collaborative filtering by learning a sparse matrix of aggregation coefficients between items that are analogous to the traditional item-item similarities [12].

The ranking score in SLIM for user u_i on item t_j is represented by $\widehat{S}_{i,j}$, which can be estimated by Eq. 4 – where N is the total number of items, W is a matrix estimating the coefficients between items, and t_h corresponds to the items rated by u_i rather than item t_j. $R_{i,h}$ therefore refers to the known rating given by u_i on other items. In other words, $\widehat{S}_{i,j}$ is calculated by a sparse aggregation

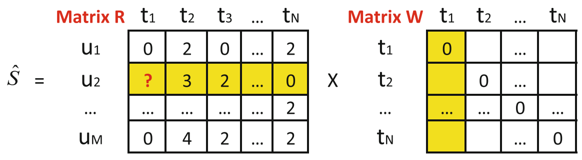

Fig. 1. Example of SLIM

(i.e., aggregated by the coefficient $W_{h,j}$) of the ratings on the other items (i.e., $R_{i,h}$) that have been rated by u_i.

$$\widehat{S}_{i,j} = R_{i,:} W_{:,j} = \sum_{\substack{h=1 \\ h \neq j}}^{N} R_{i,h} W_{h,j} \tag{4}$$

From another perspective, this function is analogous to the rating prediction function in ItemKNN, where the normalization term is removed. In addition, the coefficients between items are learnt based on the loss function shown by Eq. 5, instead of calculations in ItemKNN (e.g., using Pearson correlations or cosine similarity). β_1 and β_2 are the regularization parameters. Both ℓ_F terms (e.g. $\|W\|_F^2$) and ℓ_1 terms (e.g. $\|W\|_1$) are included, where the ℓ_1 regularization term is usually applied for sparse models.

$$\underset{W}{Minimize} \ \frac{1}{2} \left\| R_{i,j} - \widehat{S}_{i,j} \right\|_F^2 + \frac{\beta_2}{2} \|W\|_F^2 + \beta_1 \|W\|_1 \tag{5}$$

Previously, SLIM model has been extended to incorporate contextual information. These extensions include a deviation-based general contextual SLIM (GCSLIM) [23] approach which learns and estimates the contextual rating deviations from a context to another. More specifically, in GCSLIM, the goal is to create a prediction function shown in Eq. 6 to estimate the ranking score, $\widehat{S}_{i,j,c_k}$, for u_i on item t_j in contexts c_k. Again, t_h belongs to the set of items rated by u_i ($h \neq i$), and c_m is the contextual situation where u_i placed rating on t_h (note: it is allowed that $c_m = c_k$). Assume there are L contextual dimensions in total. Then $c_{m,l}$ denotes the contextual condition in the l^{th} dimension in context c_m. The function Dev measures the contextual rating deviation between two contextual conditions – it is zero if $c_{m,l} = c_{k,l}$. The matrix W continues to measure the coefficient between two items. In contrast to the SLIM approach, GCSLIM additionally learns the deviations between two contextual conditions in the algorithm, where the optimization goal is to minimize the ranking score prediction error and the loss function can be generated accordingly by adding the deviation terms to Eq. 5.

$$\widehat{S}_{i,j,c_k} = \sum_{\substack{h=1 \\ h \neq i}}^{N} (R_{i,h,c_m} + \sum_{l=1}^{L} Dev(c_{m,l}, c_{k,l})) W_{h,j} \tag{6}$$

In a summary, both CAMF and CSLIM introduce a contextual rating deviation term as part of the prediction function, extending the original recommendation algorithm. In the next section, we introduce the idea of similarity-based context-aware recommendation which replaces the rating deviation term by a context similarity term.

3 Similarity-Based Contextual Modeling

Given a particular user and multiple contexts in which recommendations can be provided, we assume that the greater the similarity between two contexts, the greater the similarity between recommendation lists that should be delivered. Therefore, contextual similarity becomes a constraint of the delivery of recommendations.

3.1 Similarity-Based CSLIM Approach

Inspired by the prediction function in Eq. 6, we can derive a similarity-based CSLIM approach which was first proposed in [14] and initially explored in [24]. The corresponding ranking score prediction can be described as follows.

$$\widehat{S}_{i,j,c_k} = \sum_{\substack{h=1 \\ h \neq j}}^{N} R_{i,h,c_m} \times W_{h,j} \times Sim(c_k, c_m) \tag{7}$$

Specifically, we aggregate the ranking score by the contextual rating score R_{i,h,c_m} with the coefficients between item t_j and t_h multiplying by the similarity between c_k and c_m. We set $h \neq j$ to avoid bias by using u_i's other contextual ratings on t_j. This strategy will ensure that we learn the coefficients between as many different items as possible. Using this approach, it is possible to learn the coefficients between items and also the similarity between different contexts by SGD accordingly.

3.2 Similarity-Based CAMF Approach

The idea above can also be applied to matrix factorization:

$$\hat{r}_{uic_k} = \overrightarrow{p_u} \cdot \overrightarrow{q_i} \cdot Sim(c_k, c_E) \tag{8}$$

Due to that $\overrightarrow{p_u} \cdot \overrightarrow{q_i}$ represents a non-contextual rating, the context similarity turns to measure the similarity between a contextual situation and a non-contextual situation. We use c_E to denote the empty context (i.e. non-contextual) situation – the value in each contextual dimension is empty or "N/A"; that is, $c_{E,1} = c_{E,2} = ... = c_{E,L} = \text{N/A}$. Therefore, the function $Sim(c_k, c_E)$ actually estimates the correlation between the c_E and the contextual situation c_k where at least one contextual condition is not empty or "N/A". Note that in Eq. 3, the contextual rating deviation can be viewed as the deviation from the empty contextual situation to a non-empty contextual situation.

$$\underset{p,q,Sim}{Minimize}\,\frac{1}{2}\,\|(r_{uic_k} - \hat{r}_{uic_k})\|_F^2 + \frac{\alpha}{2}(\|\overrightarrow{p_u}\|^2 + \|\overrightarrow{q_i}\|^2 + Sim^2) \tag{9}$$

Accordingly, the user and item vectors, as well as the contextual similarity can be learned by SGD, where the loss function is shown in Eq. 9 and α denotes the regularization parameter. Note that this is the general form for the loss function, where the term "Sim^2" should be specified and adjusted accordingly when it is modeled in different ways. Here, we will introduce three context similarity models as follows.

3.3 Modeling Context Similarity

Apparently, the form of the context similarity will vary if it is represented in different ways. The remaining challenge is how to represent or model the similarity of contexts in the prediction functions. This decision will directly influence the performance of the learning algorithm, thus it is necessary to discuss and explore different options. In this paper, we introduce three ways to represent similarity of contexts as follows. Note that we introduce those notions in general – the context similarity or correlation is assumed to be measured between any two contextual situations c_k and c_m. However, in order to prevent having to compute an quadratic number of between-context similaritiers, we measure similarity between c_k and c_E, where c_E is the empty context.

Independent Context Similarity (ICS). An example of a similarity matrix can be seen in Table 2. With Independent Context Similarity, we only measure the similarity between two contextual conditions when those they lie on the same contextual dimension, e.g., we never measure the similarity between "Time = Weekend" and "Location = Home", since they are from two different dimensions. Each pair of contextual dimensions are assumed to be independent. In this case, the similarity between two contexts can be represented by the product of the similarities among different dimensions. For example, assume c_k is {Time = Weekend, Location = Home}, and c_m is {Time = Weekday, Location = Cinema}, the similarity between c_k and c_m can be represented by the

Table 2. Example of a similarity matrix

	Time = Weekend	Time = Weekday	Location = Home	Location = Cinema
Time = Weekend	1	0.54	N/A	N/A
Time = Weekday	0.54	1	N/A	N/A
Location = Home	N/A	N/A	1	0.82
Location = Cinema	N/A	N/A	0.82	1

similarity of <Time = Weekend, Time = Weekday> multiplied by the similarity of <Location = Home, Location = Cinema>, since the two dimensions, "Time" and "Location" are assumed as independent.

Assuming there are L contextual dimensions in total, the similarities can be depicted by Eq. 10, where $c_{k,l}$ is used to denote the value of contextual condition in the l^{th} dimension in context c_k, and the "similarity" function is used to represent the similarity between two contextual conditions, which is also what to be learnt in the optimization. In other words, the similarity between two contexts is represented by the multiplication of the individual similarities between contextual conditions on each dimension.

$$Sim(c_k, c_m) = \prod_{l=1}^{L} similarity(c_{k,l}, c_{m,l}) \tag{10}$$

These similarity values (i.e., $similarity(c_{k,l}, c_{m,l})$) can be learned by the optimization process via the base algorithm (either SLIM or MF). The risk of this representation is that some information may be lost if similarities are not in fact independent in different dimensions. For example, if users usually go to cinema to see romantic movies with their partners, the "Location" (e.g. at cinema) and "Companion" (e.g. partners) may have significant correlations as a result.

Latent Context Similarity (LCS). As noted earlier, contextual rating data is often sparse, since it is somewhat unusual to have users rate items repeatedly within multiple contextual situations. This poses a difficulty when new contexts are encountered. For example, the similarity between a *new pair* of contexts <"Time = Weekend", "Time = Holiday"> may be required in the testing set, but it may not have been learned from the training data due to the sparsity problem. But, the similarity for two *existing pairs*, <"Time = Weekend", "Time = Weekday"> and <"Time = Weekday", "Time = Holiday">, may have been learned. In this case, this representation suffers from the contextual rating sparsity problem. Treating each dimension independently prevents the algorithm from taking advantage of comparisons that might be made across dimensions.

To alleviate this situation, we represent each contextual condition by a vector of weights over a set of latent factors (we use 5 latent factors in our experiments), where the weights are initialized at the beginning and learnt by the optimization process. The dot product between two vectors can be used to denote the similarity between each pair of contextual conditions. As a result, even if the newly observed pair does not exist in the training data, the weights in the vectors representing the two conditions (i.e., "Time = Weekend" and "Time = Holiday") will be learned and updated by the learning process over *existing pairs*, and the similarity for the *new pair* can be easily computed using the dot product. The similarity is given by

$$similarity(c_{k,l}, c_{m,l}) = V_{c_{k,l}} \bullet V_{c_{m,l}} \tag{11}$$

where $V_{c_{k,l}}$ and $V_{c_{m,l}}$ denote the vector representation for the contextual condition $c_{k,l}$ and $c_{m,l}$, respectively, over the space of latent factors. We then use the same similarity product calculation as in Eq. 10. We call this approach the *Latent Context Similarity* (LCS) model. This approach was able to improve the performance of deviation-based CSLIM algorithms [15]. In contrast to the independent context similarity approach, LCS provides more flexibility, but it also has the added computational costs associated with learning the latent factors. In LCS, what to be learnt in the optimization process are the vectors of weights representing each contextual condition.

3.4 Multidimensional Context Similarity (MCS)

In the multidimensional context similarity model, we assume that contextual dimensions form a multidimensional coordinate system. An example is depicted in Fig. 2.

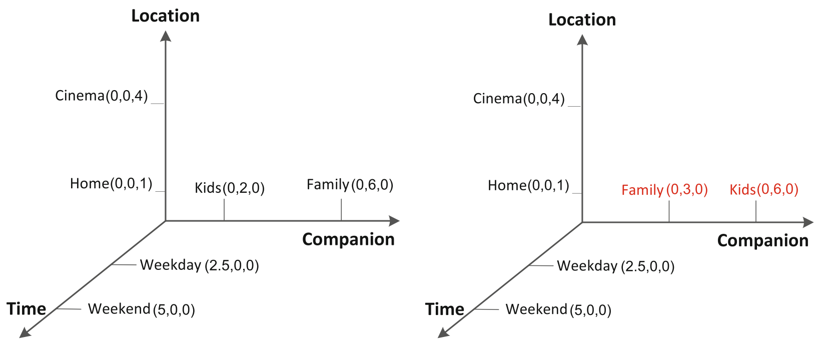

Fig. 2. Example of multidimensional coordinate system

Let us assume that there are three contextual dimensions: time, location and companion. We assign a real value to each contextual condition in those dimensions, so that each condition can locate a position in the corresponding axis. In this case, a context (as a set of contextual conditions) can be viewed as a point in the multidimensional space. Accordingly, the distance between two such points can be used as the basis for a similarity measure. In this approach, the real values for each contextual condition are the parameters to be learned in the optimization process. For example, the values for "family" and "kids" are updated in the right-hand side of the figure. Thus, the position of the data points associated to those two contextual conditions will be changed as well as the distance between the corresponding two contexts.

To avoid unconstrained distance measures, the values assigned to the contextual conditions can be normalized to be within the range [0, 1]. As a result, the data points can be represented as a cube where the length of each side equals 1. The similarity can be measured as the inverse of the distance between two data points. In our experiments, we use Euclidean distance to measure the distances, though other distance measures can also be used. The computational cost is directly associated with the number of contextual dimensions and conditions, which may make this approach the highest-cost model. Again, the number of contextual conditions can be reduced by context selection.

3.5 An Example: Constructing Algorithms

We can incorporate those three representations for context similarity (i.e., ICS, LCS and MCS) to the prediction functions in similarity-based CSLIM and CAMF respectively. Here, we give examples of how to build similarity-based CAMF. Other algorithms, using SLIM as the base, can be built in a similar manner.

As mentioned before, the prediction function in similarity-based CAMF can be shown in Eq. 8, and the loss function is described by Eq. 9. Assume, here c_k equations to <"Time = Weekday", "Location = Home">, and c_E is the empty contexts <"Time = N/A", "Location = N/A">. In ICS, contextual dimensions are assumed as independent, so the context similarity between c_k and c_E can be represented by follows.

$$sim_1 = similarity(\text{“}Time = Weekday\text{”}, \text{“}Time = N/A\text{”}) \tag{12}$$

$$sim_2 = similarity(\text{“}Location = Home\text{”}, \text{“}Location = N/A\text{”}) \tag{13}$$

$$Sim(c_k, c_E) = sim_1 \times sim_2 \tag{14}$$

Given the loss function in Eq. 9, what to be learnt are user and item vectors, sim1 and sim2. Based on stochastic gradient descent, those parameters can be updated as follows, where α is the regularization parameter and β is the learning rate.

$$err = r_{uic_k} - \hat{r}_{uic_k} \tag{15}$$

$$\overrightarrow{p_u} = \overrightarrow{p_u} + \beta \cdot (err \cdot \overrightarrow{q_i} \cdot Sim(c_k, c_E) - \alpha \cdot \overrightarrow{p_u}) \tag{16}$$

$$\overrightarrow{q_i} = \overrightarrow{q_i} + \beta \cdot (err \cdot \overrightarrow{p_u} \cdot Sim(c_k, c_E) - \alpha \cdot \overrightarrow{q_i}) \tag{17}$$

$$sim_1 = sim_1 + \beta(err \cdot (\overrightarrow{p_u} \cdot \overrightarrow{q_i}) \cdot sim_2 - \alpha \cdot sim_1) \tag{18}$$

$$sim_2 = sim_2 + \beta(err \cdot (\overrightarrow{p_u} \cdot \overrightarrow{q_i}) \cdot sim_1 - \alpha \cdot sim_2) \tag{19}$$

In LCS, each context condition is represented by a vector. Assume the vectors for "Time = Weekday", "Time = N/A", "Location = Home", "Location = N/A" can be denoted by $\overrightarrow{V_{Tw}}$, $\overrightarrow{V_{Tna}}$, $\overrightarrow{V_{Lh}}$ and $\overrightarrow{V_{Lna}}$ respectively. The context similarity is still calculated by Eq. 27, but the individual similarity in each context dimension is switched to be:

$$sim_1 = \overrightarrow{V_{Tw}} \cdot \overrightarrow{V_{Tna}} \tag{20}$$

$$sim_2 = \overrightarrow{V_{Lh}} \cdot \overrightarrow{V_{Lna}} \tag{21}$$

What is being learned in LCS are the user and item vectors, as well as those four vectors in this example. Accordingly, the updating functions for user and item vectors are the same as shown in Eqs. 22–24. The four vectors can be updated as follows:

$$\overrightarrow{V_{Tw}} = \overrightarrow{V_{Tw}} + \beta(err \cdot (\overrightarrow{p_u} \cdot \overrightarrow{q_i}) \cdot sim_2 \cdot \overrightarrow{V_{Tna}} - \alpha \cdot \overrightarrow{V_{Tw}}) \tag{22}$$

$$\overrightarrow{V_{Tna}} = \overrightarrow{V_{Tna}} + \beta(err \cdot (\overrightarrow{p_u} \cdot \overrightarrow{q_i}) \cdot sim_2 \cdot \overrightarrow{V_{Tw}} - \alpha \cdot \overrightarrow{V_{Tna}}) \tag{23}$$

$$\overrightarrow{V_{Lh}} = \overrightarrow{V_{Lh}} + \beta(err \cdot (\overrightarrow{p_u} \cdot \overrightarrow{q_i}) \cdot sim_1 \cdot \overrightarrow{V_{Lna}} - \alpha \cdot \overrightarrow{V_{Lh}}) \tag{24}$$

$$\overrightarrow{V_{Lna}} = \overrightarrow{V_{Lna}} + \beta(err \cdot (\overrightarrow{p_u} \cdot \overrightarrow{q_i}) \cdot sim_1 \cdot \overrightarrow{V_{Lh}} - \alpha \cdot \overrightarrow{V_{Lna}}) \tag{25}$$

In MCS, we use a real value to represent the position of each context condition, e.g., T_1 positions "Time = Weekday", T_0 denotes "Time = N/A", L_1 positions "Location = Home" and L_0 indicates "Location = N/A". The context similarity is described as:

$$Dist = \sqrt{(T_1 - T_0)^2 + (L_1 - L_0)^2} \tag{26}$$

$$Sim(c_k, c_E) = 1 - Dist \tag{27}$$

To make sure the similarity values are in the range [0, 1], the position of each context condition should be limited to $[0, \frac{1}{\sqrt{D}}]$, where D is the number of context dimensions. Or, a better solution is to use bounded (or projected) gradient method. Therefore, what to be learnt in MCS are user and item vectors and those real values representing the positions of each context condition in the multidimensional context space. The updating function for those positions can be described as follows:

$$T_1 = T_1 + \beta(err \cdot (\overrightarrow{p_u} \cdot \overrightarrow{q_i}) \cdot \frac{T_1 - T_0}{Dist} - \alpha \cdot T_1) \tag{28}$$

$$T_0 = T_0 - \beta(err \cdot (\overrightarrow{p_u} \cdot \overrightarrow{q_i}) \cdot \frac{T_1 - T_0}{Dist} + \alpha \cdot T_0) \tag{29}$$

$$L_1 = L_1 + \beta(err \cdot (\overrightarrow{p_u} \cdot \overrightarrow{q_i}) \cdot \frac{L_1 - L_0}{Dist} - \alpha \cdot L_1) \tag{30}$$

$$L_0 = L_0 - \beta(err \cdot (\overrightarrow{p_u} \cdot \overrightarrow{q_i}) \cdot \frac{L_1 - L_0}{Dist} + \alpha \cdot L_0) \tag{31}$$

4 Experimental Evaluation

The number of context-aware data sets is quite limited because ratings in multiple contexts are difficult to collect and user privacy is often a concern. In CARS research, most publicly available data sets are collected from surveys, resulting in small and sparse data sets. In this paper, we select three context-aware data sets with different numbers of contextual dimensions and conditions. *Restaurant* data [13] is comprised of users' ratings on restaurants in city of Tijuana, Mexico. *Music* data [3] captures users' ratings on music tracks in different driving and traffic conditions. The *Tourism* data [4] collects users' places of interest (POIs) from mobile applications. The characteristics of these data sets are summarized in Table 3. For more specific information about the contextual dimensions and conditions, please refer to the original papers using those data sets.

Table 3. Context-aware data sets

	Restaurant	Music	Tourism
Rating profiles	50 users, 40 items 2314 ratings	40 users, 139 items 3940 ratings	25 users, 20 items 1678 ratings
# of contextual dimensions	2	8	14
# of contextual conditions	7	34	67
Rating scale	1–5	1–5	1–5

4.1 Evaluation Protocols

We use five-fold cross validation on our data sets, performing top 10 recommendation task and using precision and mean average precision (MAP) as the evaluation metrics. Precision is defined as the ratio of relevant items selected to number of items recommended in a specific context. MAP is another popular ranking metric which additional takes the ranks of the recommended items into consideration. It is calculated by Eq. 32, where M denotes the number of the users, and N is the size of the recommendation list, where $P(k)$ means the precision at cut-off k in the item recommendation list, i.e., the ratio of number of users followed up to the position k over the number k, where m in $ap@N$ denotes the number of relevant items.

$$MAP@N = \sum_{i=1}^{M} ap@N/M, \text{ where } ap@N = \frac{\sum_{k=1}^{N} P(k)}{\min(m, N)} \tag{32}$$

For comparison purposes, tensor factorization (TF) [9] is chosen as a baseline since it is an independent contextual modeling method. The proposed similarity-based CSLIM and CAMF were built upon the SLIM and MF approaches, therefore we would like to compare them with the deviation-based algorithms. More specifically, CAMF [5] is selected as the baseline. We tried all three variants: CAMF_CI, CAMF_CU and CAMF_C, and only present the best performing one, denoted as "CAMF-Dev", in the following sections. Similarly, we choose the best performing GCSLIM [23], denoted as "CSLIM-Dev". For the similarity-based context-aware recommendation algorithms, we simply use "Algorithm-SimilarityModel" to present the algorithm, e.g., "CAMF-ICS" denotes the similarity-based CAMF using independent context similarity modeling. "CSLIM-MCS" indicates the similarity-based CSLIM using multidimensional context similarity.

4.2 Analysis and Findings

The experimental results by similarity-based CSLIM approaches can be described by the Fig. 3, and the results based on similarity-based CAMF can be shown in Fig. 4.

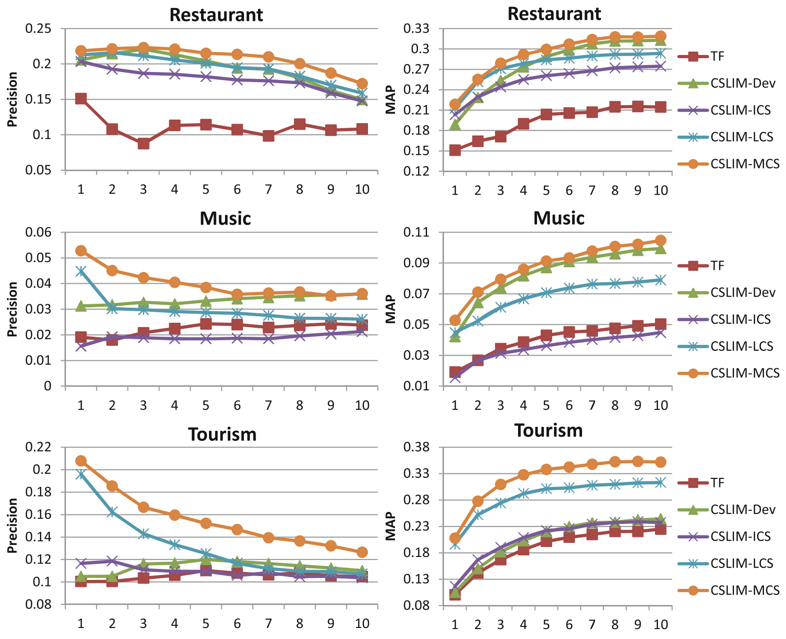

Fig. 3. Similarity-based CSLIM (x-axis denotes the number of recommendations.)

Overall Comparisons. The best performing algorithm is "CSLIM-MCS" which denotes the similarity-based CSLIM using multidimensional context similarity model. In Fig. 3, we can see that CSLIM-Dev always outperforms TF. Not all the similarity-based CSLIM approaches are able to beat TF, e.g., in the music data, CSLIM using the independent context similarity works worse than TF. However, we can always find one similarity-based CSLIM that outperforms both the TF and deviation-based CSLIM, suggesting that the context similarity works well if appropriately designed. The similar pattern can be found in the similarity-based CAMF approaches shown in Fig. 4.

In general, recommendation models using latent context similarity always outperform the ones using independent context similarity, which reveals that the LCS is able to successfully alleviate the sparsity problem mentioned previously. In addition, multidimensional context similarity is the best representation for context similarity at the price of increased computational effort, although this can be alleviated through careful context selection or merging.

Comparisons Between Deviation-Based and Similarity-Based Models. Similarity-based context-aware recommendation models are able to outperform the best performing deviation-based algorithms when the context similarity is appropriately represented. Generally, CSLIM-MCS is able to beat CSLIM-Dev, and CAMF-MCS always outperforms CAMF-Dev in our three data sets. This confirms that learning context similarity is a better way than modeling the con-

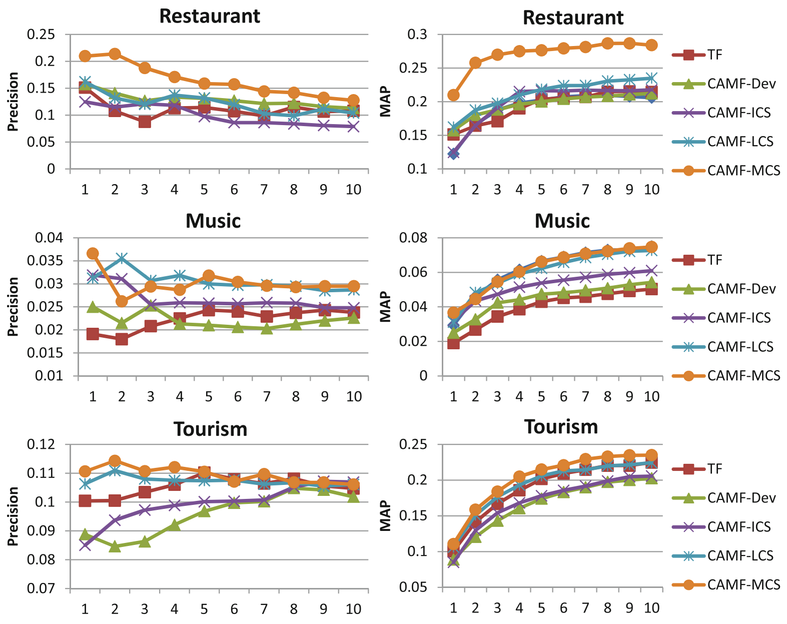

Fig. 4. Similarity-based CAMF (x-axis denotes the number of recommendations.)

textual rating deviations in developing dependent context modeling approaches. We expect similar benefits if the approach were applied to other recommendation algorithms, e.g. the slope-one recommendation model.

Table 4. Comparison between CAMF and CSLIM

		Restaurant				Music				Tourism			
		CAMF-Dev	CAMF-MCS	CSLIM-Dev	CSLIM-MCS	CAMF-Dev	CAMF-MCS	CSLIM-Dev	CSLIM-MCS	CAMF-Dev	CAMF-MCS	CSLIM-Dev	CSLIM-MCS
Precision	@5	0.1309	0.1586	**0.2044**	**0.2151**	0.0210	0.0385	**0.0332**	**0.0385**	0.0968	0.1105	**0.1200**	**0.1522**
	@10	0.1130	0.1276	**0.1496**	**0.1723**	0.0226	0.0361	**0.0359**	**0.0361**	0.1018	0.1061	**0.1101**	**0.1265**
MAP	@5	0.2001	0.2765	**0.2889**	**0.2993**	0.0474	0.0661	**0.0871**	**0.0913**	0.1740	0.2148	**0.2199**	**0.3379**
	@10	0.2122	0.2840	**0.3128**	**0.3187**	0.0542	0.0747	**0.0995**	**0.1047**	0.2026	0.2350	**0.2442**	**0.3518**

Comparisons Between CSLIM and CAMF. To compare CSLIM and CAMF in both their deviation-based and similarity-based formulations, we extracted the top-5 and top-10 precision and MAP values shown in Table 4. (Only the best performing models are shown.) From the table, we can see that CSLIM outperforms CAMF significantly when the same strategy (either deviation or correlation modeling) is applied. CSLIM-MCS works the best in general. It is not surprising, since earlier work [12] has demonstrated that SLIM is able to outperform matrix factorization in many settings. The result that CSLIM outperforms CAMF further confirms this pattern.

Summary. Similarity-based context-aware recommendation algorithms are able to outperform the deviation-based algorithms and the independent contextual modeling approach (i.e., TF). CSLIM always outperform the CAMF when the same strategy (either deviation-based or similarity-based modeling) is applied. Choosing the appropriate representation for context similarity is important, since it directly influences the performance of the similarity-based context-aware recommendation. In general, the multidimensional context similarity is the best way to model the similarity of contexts at the price of computational costs, but these costs can be alleviated by context selection.

5 Conclusions and Future Work

In this paper, we proposed the idea of similarity-based contextual recommendation where context similarity is incorporated into the recommendation algorithm. Specifically, we chose sparse linear method and matrix factorization as two base algorithms. We then developed similarity-based contextual recommendation algorithms by modeling the similarity of contexts in three different ways. Our experimental results demonstrate that a multidimensional approach is the best representation for context similarity, and that CSLIM-MCS works the best among all compared recommendation algorithms.

One direction that we will explore in future work is to explore pre-selection or other pre-processing approaches to reduce noise and improve the efficiency of the multidimensional similarity model. Also, we believe the context similarity can be incorporated into more other recommendation algorithms, such as slope one recommender, which we will explore in the future. Furthermore, the correlations or similarities between contexts could be used for interpretations, such as interpreting emotional effects [19] in RS.

References

1. Abowd, G.D., Dey, A.K.: Towards a better understanding of context and context-awareness. In: Gellersen, H.-W. (ed.) HUC 1999. LNCS, vol. 1707, pp. 304–307. Springer, Heidelberg (1999)
2. Adomavicius, G., Mobasher, B., Ricci, F., Tuzhilin, A.: Context-aware recommender systems. AI Mag. **32**(3), 67–80 (2011)
3. Baltrunas, L., Kaminskas, M., Ludwig, B., Moling, O., Ricci, F., Aydin, A., Lüke, K.-H., Schwaiger, R.: InCarMusic: context-aware music recommendations in a car. In: Huemer, C., Setzer, T. (eds.) EC-Web 2011. LNBIP, vol. 85, pp. 89–100. Springer, Heidelberg (2011)
4. Baltrunas, L., Ludwig, B., Peer, S., Ricci, F.: Context-aware places of interest recommendations for mobile users. In: Marcus, A. (ed.) HCII 2011 and DUXU 2011, Part I. LNCS, vol. 6769, pp. 531–540. Springer, Heidelberg (2011)
5. Baltrunas, L., Ludwig, B., Ricci, F.: Matrix factorization techniques for context aware recommendation. In: Proceedings of the Fifth ACM Conference on Recommender Systems, pp. 301–304. ACM (2011)

6. Baltrunas, L., Ricci, F.: Experimental evaluation of context-dependent collaborative filtering using item splitting. User Model. User-Adap. Inter. **24**(1–2), 7–34 (2014)
7. Chen, A.: Context-aware collaborative filtering system: predicting the user's preference in the ubiquitous computing environment. In: Strang, T., Linnhoff-Popien, C. (eds.) LoCA 2005. LNCS, vol. 3479, pp. 244–253. Springer, Heidelberg (2005)
8. Codina, V., Ricci, F., Ceccaroni, L.: Local context modeling with semantic pre-filtering. In: Proceedings of the 7th ACM Conference on Recommender Systems, pp. 363–366. ACM (2013)
9. Karatzoglou, A., Amatriain, X., Baltrunas, L., Oliver, N.: 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 (2010)
10. Koren, Y., Bell, R., Volinsky, C.: Matrix factorization techniques for recommender systems. IEEE Comput. **42**(8), 30–37 (2009)
11. Liu, L., Lecue, F., Mehandjiev, N., Xu, L.: Using context similarity for service recommendation. In: 2010 IEEE Fourth International Conference on Semantic Computing (ICSC), pp. 277–284. IEEE (2010)
12. Ning, X., Karypis, G.: SLIM: sparse linear methods for top-n recommender systems. In: 2011 IEEE 11th International Conference on Data Mining, pp. 497–506. IEEE (2011)
13. Ramirez-Garcia, X., Garca-Valdez, M.: Post-filtering for a restaurant context-aware recommender system. In: Castillo, O., Melin, P., Pedrycz, W., Kacprzyk, J. (eds.) Recent Advances on Hybrid Approaches for Designing Intelligent Systems, vol. 547, pp. 695–707. Springer, Heidelberg (2014)
14. Zheng, Y.: Deviation-based and similarity-based contextual SLIM recommendation algorithms. In: Proceedings of the 8th ACM Conference on Recommender Systems, pp. 437–440. ACM (2014)
15. Zheng, Y.: Improve general contextual SLIM recommendation algorithms by factorizing contexts. In: Proceedings of the 30th Annual ACM Symposium on Applied Computing, pp. 929–930. ACM (2015)
16. Zheng, Y.: A revisit to the identification of contexts in recommender systems. In: Proceedings of the 20th ACM Conference on Intelligent User Interfaces Companion, pp. 133–136. ACM (2015)
17. Zheng, Y., Burke, R., Mobasher, B.: Differential context relaxation for context-aware travel recommendation. In: Huemer, C., Lops, P. (eds.) EC-Web 2012. LNBIP, vol. 123, pp. 88–99. Springer, Heidelberg (2012)
18. Zheng, Y., Burke, R., Mobasher, B.: Recommendation with differential context weighting. In: Carberry, S., Weibelzahl, S., Micarelli, A., Semeraro, G. (eds.) UMAP 2013. LNCS, vol. 7899, pp. 152–164. Springer, Heidelberg (2013)
19. Zheng, Y., Burke, R., Mobasher, B.: The role of emotions in context-aware recommendation. In: ACM RecSys 2013, Proceedings of the 3rd International Workshop on Human Decision Making in Recommender Systems, pp. 21–28. ACM (2013)
20. Zheng, Y., Burke, R., Mobasher, B.: Splitting approaches for context-aware recommendation: an empirical study. In: Proceedings of the 29th Annual ACM Symposium on Applied Computing, pp. 274–279. ACM (2014)
21. Zheng, Y., Mobasher, B., Burke, R.: Context recommendation using multi-label classification. In: Proceedings of the 13th IEEE/WIC/ACM International Conference on Web Intelligence, pp. 288–295. IEEE/WIC/ACM (2014)

22. Zheng, Y., Mobasher, B., Burke, R.: CSLIM: contextual SLIM recommendation algorithms. In: Proceedings of the 8th ACM Conference on Recommender Systems, pp. 301–304. ACM (2014)
23. Zheng, Y., Mobasher, B., Burke, R.: Deviation-based contextual SLIM recommenders. In: Proceedings of the 23rd ACM Conference on Information and Knowledge Management, pp. 271–280. ACM (2014)
24. Zheng, Y., Mobasher, B., Burke, R.: Integrating context similarity with sparse linear recommendation model. In: Ricci, F., Bontcheva, K., Conlan, O., Lawless, S. (eds.) UMAP 2015. LNCS, vol. 9146, pp. 370–376. Springer, Heidelberg (2015)