Noname manuscript No.
(will be inserted by the editor)

Non-Dominated Differential Context Modeling for Context-Aware Recommendations

Yong Zheng

Received: date / Accepted: date

Abstract Context plays an important role in the process of decision making. A user's preferences on the items may vary from contexts to contexts, e.g., a user may prefer to watch a different type of the movies, if he or she is going to enjoy the movie with *partner* rather than with *children*. Context-aware recommender systems, therefore, were developed to adapt the recommendations to different contextual situations, such as time, location, companion, etc. Differential context modeling is a series of recommendation models which incorporate contextual hybrid filtering into the neighborhood based collaborative filtering approaches. In this paper, we propose to enhance differential context modeling by utilizing a non-dominated user neighborhood. The notion of *dominance relation* was originally proposed in multi-objective optimization, and it was reused to definite *non-dominated user neighborhood* in collaborative filtering recently. These non-dominated user neighbors refer to the neighbors that dominate others from different perspectives of the user similarities, such as the user-user similarities based on ratings, demographic information, social relationships, and so forth. In this paper, we propose to identify the non-dominated user neighborhood by exploiting user-user similarities over multiple contextual preferences. Our experimental results can demonstrate the effectiveness of the proposed approaches in comparison with popular context-aware collaborative filtering models over five real-world contextual rating data sets.

Keywords recommender systems · context · context-aware · collaborative filtering · non-dominated · dominance relation

Yong Zheng
College of Computing
Illinois Institute of Technology
Chicago, Illinois, 60616, USA
E-mail: yzheng66@iit.edu

1 Introduction

Recommender systems have been demonstrated as one of the successful technologies in alleviating the problem of information overload, by delivering item recommendations tailored to user preferences. Different recommendation algorithms, such as the collaborative filtering [32,20], content-based recommendation models [30,23], and hybrid approaches [9], were developed in the past decades to serve the applications in various domains, such as e-commerce, social media, online streaming, tourism, and so forth.

In the real-world practice, users may make different choices (e.g., purchasing items, watching movies, listening to songs, etc.) or like different items, if the context of the activity is different. For example, a user may choose to have lunch at a fast-food store, if there are no lunch partners. However, he or she may dine in a formal restaurant if it is a business dinner. A user may like rock music regularly, but he or she may prefer blue songs when the user is feeling sad. Context-aware recommender systems (CARS) [3,2], therefore, were proposed and developed to adapt the recommendations to different contextual situations (e.g., time, location, companion, occasion, emotional states, etc).

Context-aware collaborative filtering are the major approaches in CARS. The ideal solution is context matching in which we only use the ratings with matched contexts to build the recommendation models. For example, to predict a user's rating on a movie in contexts, "watching movie *in a cinema at weekend with family*", it's better to use rating profiles in the matched contexts. However, it results in the sparsity issue, since there may be limited ratings within the exact contexts. The issue could be more serious, especially when there are many context dimensions or variables. Differential context modeling (DCM) [38,39] is a series of models which alleviate the sparsity issue by utilizing partial matching or selecting rating profiles by weighted context similarities. DCM were built upon the user-based collaborative filtering (UBCF) which relies on the quality of user neighborhoods in the recommendation process. UBCF-based models are still popular due to its capability in interpretability and explainability, as well as the ease in implementation and deployment, even if the latent-factor models and deep learning models for recommendations were well-developed.

One of the interesting approaches to improve UBCF is utilizing non-dominated neighbors [27,28]. The notion of *dominance relation* was proposed in the area of multi-objective optimization [15], in order to determine that a solution is better than others in the multi-objective problem. Researchers reused this notion to propose *non-dominated user neighbors* which refer to the neighbors that dominate others from different perspectives of user similarities [28], such as the user-user similarities based on ratings, demographic information, social relationships, and so forth. Researchers have demonstrated that the non-dominated user or item neighborhoods are more powerful than the regular neighborhood in collaborative filtering [27,28]. In this paper, we propose to enhance the DCM by introducing the non-dominated user neighborhood. Particularly, we propose to identify these neighbors by using users' contextual preferences on the items, rather than demographic information or social relationships. The contributions in this article can be summarized as follows.

- Our work made the first attempt to introduce the notion of non-dominated user neighborhood into context-aware recommender systems. More specifically, we propose to identify the non-dominated user neighborhood based on user-user similarities over contextual preferences, and incorporate it into the existing DCM methods (i.e., differential context relation and differential context weighting).
- We propose and examine different similarity measures in the process of identifying the non-dominated user neighborhood, though there are no unique winners among multiple data sets.
- Our proposed non-dominated DCM approaches can beat the existing DCM methods, and even outperform the recommendation models based on latent-factors which represent another way to alleviate the sparsity issues in context-aware recommendations.

The remainder of the article is organized as follows: Section 2 briefly reviews context-aware recommender systems and the notion of non-dominated user neighborhood; Section 3 introduces the existing DCM approaches as a preliminary; Section 4 illustrates the proposed non-dominated DCM recommendation methods; Section 5 presents our experiments and results, followed by the conclusion and future work in Section 6.

2 Related Work

In this section, we provide the related work in context-aware recommendations, as well as the development of non-dominated user neighborhood in collaborative filtering.

2.1 Context-Aware Recommender Systems

As mentioned before, CARS additionally take context information as inputs to adapt the recommendations to different situations. Context is usually defined as "any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves" [1]. G. Adomavicius et al. [2] believed that CARS can be classified into CARS with *represnentational contexts* and CARS with *interactional contexts* (e.g., session-based recommender systems [24, 16]). The context information is interactive if they are partially observed or even not observed, and their impacts on user preferences may change dynamically. In this paper, we only focus on the CARS with represnentational contexts in which the contexts (e.g., time, location, companion, emotions, etc.) are fully observed, and the contextual effects are static or do not change frequently.

The context-aware recommendation models can be built by incorporating contexts in three ways [2]:

- *Contextual pre-filtering.* Context information can be used to filter out irrelevant rating profiles, such as ratings with non-matched or dissimilar contexts. Any traditional recommendation approaches can be applied to the remaining rating profiles to build the recommendation model.

- *Contextual post-filtering*. We can use traditional recommendation models to produce predicted rating or the recommendation list. Afterwards, context information will be considered to adjust these predicted ratings or the ranking of the items in the recommendation list.
- *Contextual modeling*. Contexts can also be used as parts of the predictive or ranking function in the recommendation model.

The pre-filtering and post-filtering approaches define a strategy to filter out irrelevant information while any traditional recommendation models can be utilized to produce the context-aware recommendations. By contrast, the contextual modeling approaches usually need to develop new recommendation models based on machine learning techniques, such as the approaches based on the factorization models [6,36] or deep learning [10,34,35].

2.2 Context-Aware Collaborative Filtering

Researchers tried to incorporate the contexts into the collaborative filtering techniques, and developed multiple context-aware collaborative filtering methods in the past years.

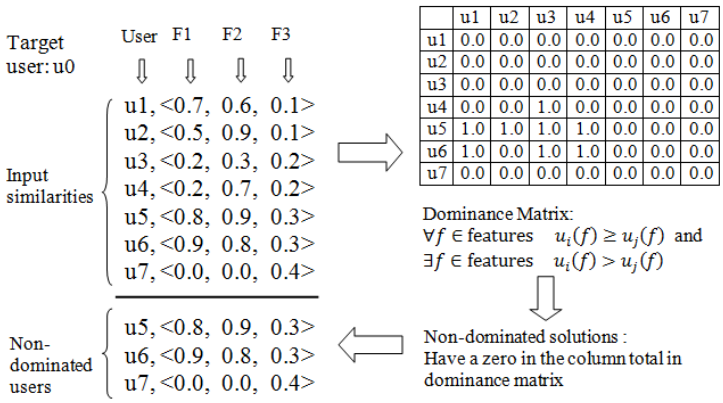
The context-aware collaborative filtering models can be classified by pre-filtering, post-filtering, and contextual modeling, as mentioned above. The exact-filtering [3] is the first contextual pre-filtering technique which uses exact contexts to filter out irrelevant rating profiles. The semantic pre-filtering (SPF) [12] can measure context similarity and filter out rating profiles with dissimilar contexts. UISplitting [40] is a combination of the user splitting [5] and item splitting [7] which fuse contexts into users and items in the recommendation model. After the pre-filtering operations, any traditional recommendation techniques can be applied to produce recommendations. By contrast, context modeling was usually built upon the latent-factor models, such as the context-aware matrix factorization (CAMF) [6], tensor factorization (TF) [17,36] and similarity-based contextual models [42]. The advanced development on deep learning-based models [10,34,35] can also be considered as contextual modeling methods. Post-filtering models [31,29,37] are less investigated and usually under-performed in comparison with the other two categories. Alternatively, context-aware collaborative filtering (CACF) models can be classified by using different types of the collaborative filtering models. The traditional collaborative filtering is composed of the memory-based (e.g., UBCF) and model-based (e.g., matrix factorization) approaches. The memory-based collaborative filtering is easy to be interpreted and the recommendations generated can be well explained. However, these models usually suffer from the sparsity issue. The model-based collaborative filtering (e.g., CAMF and TF) can perform well on sparse data, since using latent factors is one of the ways to alleviate the sparsity issue in these model-based approaches. However, it is difficult to interpret the model or the recommendations, in comparison to the memory-based recommendation models.

Since the memory-based CACF models may suffer from the sparsity issue seriously, different solutions were proposed to alleviate the sparsity for these memory-based CACF models. Chen [11] first proposed to utilize context similarity in UBCF, but these methods require items to be rated in each context condition frequently. As a result, the context similarities may not be reliable if items were

not rated in different contexts for several times. Later, DCM [38, 39] was proposed, and it is a series of context-aware collaborative filtering models based on UBCF. It was demonstrated to outperform the memory-based context-aware collaborative filtering, and even some model-based approaches, since it can avoid exact matching and alleviate the sparsity issue by introducing partial contextual matching or filtering in UBCF. Linda et al. [22] recently proposed to improve the DCM by filling in missing ratings and using a genetic algorithm as the optimizer. Filling in missing ratings was one of the methods to alleviate sparsity in traditional UBCF approaches. However, the underlying errors of these filled ratings may introduce additional risk or inaccuracy in the recommendation models. Ali et al. [4] proposed the notion of context scoring in recommender systems, but their approach relies on extra content information (e.g., move genre or year).

2.3 Non-Dominated User Neighborhood

The performance of UBCF usually relies on the quality of the user neighborhood. One of the interesting approaches to identify better user neighborhood is the proposal of using non-dominated user neighborhood. The "dominance" and "non-dominance" relations are the notions originally proposed in multi-objective optimization [14, 13]. Ortega et al. [27] and Ozsoy et al. [28] borrowed these notions and proposed to identify non-dominated users as the optimal user neighborhood for UBCF.



	u1	u2	u3	u4	u5	u6	u7
u1	0.0	0.0	0.0	0.0	0.0	0.0	0.0
u2	0.0	0.0	0.0	0.0	0.0	0.0	0.0
u3	0.0	0.0	0.0	0.0	0.0	0.0	0.0
u4	0.0	0.0	1.0	0.0	0.0	0.0	0.0
u5	1.0	1.0	1.0	1.0	0.0	0.0	0.0
u6	1.0	0.0	1.0	1.0	0.0	0.0	0.0
u7	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Fig. 1 Steps to find non-dominated neighborhood [28]

Take Ozsoy's approach [28] as shown by Figure 1 for example, given a target user u, the user-user similarities can be computed from three views or features – the ratings over items, users' hometown locations, and social relationships (i.e., as shown by F_1, F_2, F_3 in Figure 1). As a result, the similarities between a neighbor candidate and target user can be represented by a similarity vector with three dimensions, i.e., similarities based on ratings, locations, and social relationships respectively, as shown by the "input similarities" in Figure 1. A user-user dominance matrix thereby can be produced by filling in the dominance value $dom(p, q)$

as shown by Equation 1. We use $Sim^f_{p,u}$ to denote the similarity value between users p and u based on feature f (e.g., rating, location, or social relationship). Candidates whose column sum is zero in the dominance matrix will finally be considered as the non-dominated neighbors for the target user u in UBCF.

$$dom(p,q) = \begin{cases} 1 & \forall f : Sim^f_{p,u} \geq Sim_{q,u}, \exists f : Sim^f_{p,u} > Sim^f_{q,u} \\ 0 & Otherwise \end{cases} \quad (1)$$

In the traditional collaborative filtering, a user neighborhood is found based on the user-user similarities computed based on a single dimension – ratings on co-rated items. By contrast, the non-dominated user neighborhood can be identified by user-user similarities computed from multiple dimensions, such as ratings, demographic information, social relationships, etc. The selected neighbors in the non-dominated user neighborhood thereby have higher similarities with the target user than others from multiple perspectives (e.g., ratings, demographic information, social relationships, etc.). That's the major reason that the non-dominated user neighborhood may be better than the regular neighborhood in UBCF.

By contrast, the dominance relationship defined by Ortega et al. [27] relies on the absolute rating difference on co-rated items between two candidates, while the list of these items are the ones rated by the target user u. However, this method may suffer from the sparsity issue, since the number of co-rated items may be limited.

Researchers have demonstrated that the UBCF can be improved by utilizing these non-dominated neighbors. In our work, we extend the approach by Ozsoy et al., and propose our approach to identify the non-dominated user neighborhood based on contextual preferences for DCM.

3 Preliminary: Differential Context Modeling

We introduce the original DCM approaches in this section, followed by the discussion about our non-dominated DCM in Section 4.

3.1 Terminologies and Notations

To better describe the CARS algorithms, we introduce the terminology and notations used in this paper first. Take the sample data in Table 1 for example, there is one user u, one movie i, and three context variables – Time (weekend or weekday), Location (at home or cinema) and Companion (alone, partner, family). In the following discussion, we use *context dimension* to denote the contextual variable, e.g., "Location". The term *context condition* refers to a specific value in a dimension, e.g., "home" and "cinema" are two contextual conditions in "Location". The *contexts* or *context situation* is, therefore, a set of contextual conditions, e.g., {*weekend, home, family*}.

The symbols or notations used in the following discussions can be listed in Table 2.

Table 1 Contextual Ratings on Movies

User	Item	Rating	Time	Location	Companion
u	i	3	weekend	home	alone
u	i	4	weekday	home	children
u	i	5	weekend	cinema	partner
u	i	?	weekday	home	family

Table 2 Notations

Notation	Explanations
M, Z	the number of users and context dimensions, respectively
c, x, y	context situations
$r_{u,i,c}$	rating given by user u on item i in context situation c
$\hat{r}_{u,i,c}$	predicted rating for user u on item i in context situation c
$r_{u,i}$	rating given by user u on item i without considering contexts
$\hat{r}_{u,i}$	predicted rating for user u on item i without considering contexts
N_u	neighborhood for user u
T_r, T_e	training and testing set, respectively

3.2 User-Based Collaborative Filtering

UBCF assumes a user's preference on one item is close to a group of users' tastes on the same item. This group of users should share similar preferences with the target user, and is usually named as user neighborhood.

$$\hat{r}_{u,i} = \bar{r}_u + \frac{\sum\limits_{a \in N_u} (r_{a,i} - \bar{r}_a) \times sim(a,u)}{\sum\limits_{a \in N_u} sim(a,u)} \tag{2}$$

The standard prediction function in UBCF can be described by Equation 2, where u is a user, i is an item, and N_u is the user neighborhood for user u. The algorithm calculates $\hat{r}_{u,i}$ which is the predicted rating by user u on item i. The similarity between users u and neighbor a can be computed from the ratings of their co-rated items based on popular similarity measure (e.g., Pearson correlation, cosine similarity, etc.). N_u can be formed by the top-K similar neighbors based on the user-user similarities. It usually suffers from the rating sparsity issue.

DCM delivers two context-aware recommendation models based on UBCF – differential context relaxation (DCR) and differential context weighting (DCW) which can be described in Section 3.3 and Section 3.4 respectively.

3.3 Differential Context Relaxation

As mentioned before, the ideal solution for CARS is applying a context matching on the rating profiles. However, it may suffer from the sparsity issue. Take Table 1 for example, to predict the user's rating on the movie in the situation "at home with family at weekday", we cannot find the ratings with exact-matched contexts. By contrast, DCR [38] can alleviate this sparsity issue by introducing two solutions.

First of all, DCR utilizes the optimal context matching which could be exact matching or partial matching. In Table 1, we are not able to find matched contexts by using all three dimensions (i.e., time, location, companion), but we can match

them if we use relaxed context dimensions. We can find one rating entry (i.e., the 2^{nd} row in Table 1) matched if we use "time" and "location" only, and there are two ratings (i.e., the 1^{st} and 2^{nd} row in Table 1) matched if we use "location" only.

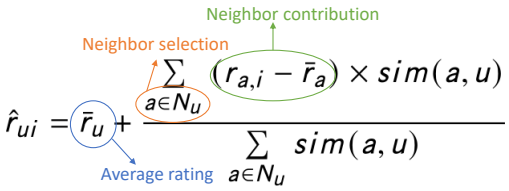


Fig. 2 Three components in UBCF

In addition, UBCF can be decomposed into different components. Figure 2 described a UBCF with three components – neighbor selection, neighbor contribution and computation of user u's average rating. We believe that each component may need different context relaxations in order to obtain the accurate computation for each component. For example, we may use "companion" to find neighbors (i.e., candidates must rate items in the same condition matched by the dimension "companion"), and utilize "time" and "location" to perform context matching and calculate u's average rating.

With more relaxation, we can find more rating profiles with matched contexts to be used in the computation for each component in UBCF. However, the relaxation operation may also bring noises if contexts are relaxed too much. We consider it as a feature selection process and use binary particle swarm optimizer (BPSO) [19] to find the optimal context relaxation. More specifically, the relaxed contexts for each component can be represented by a binary vector while the size of the vector is as same as the number of context dimensions. If a bit in the vector is 1, it indicates the corresponding context dimension is selected. Otherwise, that dimension is not selected in the context relaxation.

3.4 Differential Context Weighting

Context matching can alleviate the issue, but it still suffers from the sparsity problem. DCW [39] was proposed to further alleviate the sparsity issue by introducing context similarity. The underlying idea is straightforward – we can assign a weight to each context dimension, so that the context similarity can be calculated based on a weighted Jaccard similarity (wjacc), as shown by Equation 3, where x and y are two context situations, and w refers to the weight vector. f denotes the index of the matched context dimensions, and Z refers to the number of context dimensions in the data.

$$wjacc(x, y, w) = \frac{\sum_{f \in x \cap y} w_f}{\sum_{t=1}^{Z} w_t} \tag{3}$$

DCW assigns a weight vector to each component in UBCF, and uses a minimal similarity value as hyperparameter. Only the rating profiles with similar contexts

can be used for the computation in each component. Accordingly, we can use the particle swarm optimizer (PSO) [18] to learn these best weights.

Algorithm 1: DCW by PSO

```
1  initialization;
2  while t <= MaxIteration do
3  |  for each particle do
4  |  |  Iterate rating entries in training set, e.g. fetch a contextual rating entry;
5  |  |  Use particle's position to calculate context similarity by Equation 3;
6  |  |  Use similar threshold to filter out dissimilar ratings for each component;
7  |  |  Calculate each component in Equation 2;
8  |  |  Calculate predicted rating by Equation 2;
9  |  |  Calculate fitness value (i.e., sum of squared prediction errors);
10 |  |  if fitness is better than pBest then
11 |  |  |  update pBest and its position;
12 |  |  end
13 |  |  if fitness is better than gBest then
14 |  |  |  update gBest and its position;
15 |  |  end
16 |  end
17 |  for each particle do
18 |  |  update particle velocity [18];
19 |  |  update particle position [18];
20 |  end
21 |  t = t + 1;
22 end
```

The DCW by using PSO as optimizer can be briefly described by the Algorithm 1. At the beginning, we initialize a population (i.e., a set of particles) and the position for each particle (i.e., the weight vector). Each particle will produce predicted rating by using its position. The prediction process follows DCW-based UBCF (i.e., lines 5-8) in which we calculate context similarity and filter out dissimilar rating profiles for each component. To update the position of each particle for the next learning iteration, the velocity defines how much the position should be changed, and it is updated by referring to the $pBest$ and $gBest$ solution. $pBest$ refers to the best fitness value in the learning history of particle p, while $gBest$ denotes the global best fitness value among all particles. The velocity and position of each particle will be updated based on a combination of self-learning (i.e., $pBest$) and swarm-learning (i.e., $gBest$) [18].

The similarity function $sim(a, u)$ in Equation 2 can be considered as the 4^{th} component in DCW. By adding this component, it is able to provide finder-grained and improved performance, but it also increases the computational cost in there are many context dimensions. In this paper, we only discuss and extend the three-component UBCF models in the following sections.

4 Non-Dominated Differential Context Modeling

The dominance relationship proposed by Ozsoy et al. [28] relies on the user-user similarities from different perspectives, as shown by Equation 1. Ozsoy et al. considered user locations and social ties in addition to the users' ratings on the items.

In CARS, we may not have user information (e.g., demographic information, social ties, etc.) available, but we have users' contextual preferences on the items. By this way, we can find the non-dominated neighbors by using user-user similarities based on co-rated items and co-rated contexts. Note that we only change the neighbor selection component in Figure 2, while the operations of context relaxation or context weighting on other components remain the same. We discuss these non-dominated DCM (ND-DCM) approaches in details as follows.

4.1 Non-Dominated Differential Context Relaxation (ND-DCR)

First, we build a user-item (UI) rating matrix, and each user is represented by a vector of ratings on the list of items. We use the average rating if a user rated an item for several times in different contexts. As a result, we can calculate the user-user similarities based on the UI rating matrix.

Table 3 Example of User-Condition Rating Matrix

	Time: weekend	Time: weekday	Location: home	Location: cinema	Companion: family	Companion: partner	Companion: children
u_1	3.3	3.2	3.0	4.0	4.1	3.8	3.4
u_2	3.5	3.0	3.1	3.2	2.8	4.1	4.0
u_3	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
u_M	...	...	...	...	...	...	...

In addition, we can build a user-condition (UC) rating matrix, as shown by Table 3. The matrix is filled by a user's average rating on items in each specific context condition. Therefore, each user can be represented by a rating vector over different context conditions.

Recall that we also have the context relaxation for this neighbor selection component. The context relation can tell which context dimension will be selected for the process of context matching. In addition to this context matching process, we also utilize the context relaxation to shrink the UC rating matrix. More specifically, there could be two strategies to reduce the dimension of the UC rating matrix. Assume that we want to predict a user's rating on an item in the contexts "at weekend, at home, with family", and the dimensions "time" and "location" are selected in the context relaxation. We describe the two strategies as follows.

- *At condition level.* In this case, we will use the two columns "Time: weekend" and "Location: home" in Table 3 only, since these two columns denote the matched conditions by context relaxation. By this way, each user is represented by a rating vector with size two. However, the risk in this strategy is that the rating vector for a user may be too short (e.g., only one or two context dimensions are selected), which results in unreliable computations of the user-user similarities based on these vectors.
- *At dimension level.* Alternatively, we can use all the conditions associated with the selected context dimensions to represent the rating vector for each user. If "time" and "location" are selected in the context relaxation, all the columns (i.e., conditions) related to these two dimensions in Table 3 will be retained.

Table 4 UC Matrices by ND-DCR

	Time: weekend	Time: weekday
u_1	3.3	3.2
u_2	3.5	3.0
u_3	...	...
...	...	...
u_M	...	...

	Location: home	Location: cinema
u_1	3.0	4.0
u_2	3.1	3.2
u_3	...	...
...	...	...
u_M	...	...

In our work, we utilize the strategy at the dimension level. More specifically, we can have two UC rating matrices, if "Time" and "Location" are selected in the context relaxation for the neighbor selection component – one is the rating matrix which only retains the columns related to the conditions in the dimension "Time", and another one is the rating matrix with columns associated with the conditions in dimension "Location", as shown by Table 4. Finally, we can calculate user-user similarities from three perspectives – UI ratings, UC ratings associated with dimension "Time", and UC ratings associated with dimension "Location". The dominance relationships defined in Equation 1 can be reused to find the set of non-dominated neighbors. After that, the regular context matching will be applied to these non-dominated neighbors by using the relaxed contexts for neighbor selection component, in order to obtain the final list of neighbors for the DCR algorithm. Namely, we utilize context relaxation to find a non-dominated neighbors and use it again to filter out neighbors who did not rate items in the matched contexts. The operation on other components is as the same as the ones in DCR. We name this approach as non-dominated DCR, and use ND-DCR to denote it.

Table 5 UC Matrix by ND$_s$-DCR

	Time: weekend	Time: weekday	Location: home	Location: cinema
u_1	3.3	3.2	3.0	4.0
u_2	3.5	3.0	3.1	3.2
u_3	...	...	...	...
...	...	...	...	...
u_M	...	...	...	...

In ND-DCR, a UC rating matrix may still have a few of columns, if the number of conditions in a dimension is limited. Alternatively, we can merge the columns associated with the conditions in all selected dimensions, as shown by Table 5. By this way, we measure user-user similarities from two perspectives only – UI rating matrix and a single UC rating matrix. However, it may result in too many non-dominated neighbors. A common method is setting a minimal similarity threshold so that only the non-dominated neighbors with high similarities with the target user can be selected as the neighbors. We use the average rating over the similarities based on the UI and UC rating matrices to denote the similarity between two neighbor candidates, and filter out the neighbors by using the minimal similarity threshold. We name this method as simplified non-dominated DCR, and use ND$_s$-DCR to denote it.

Note that the identification of non-dominated neighbors depends on the similarity measures, as well as the user representations. We tried different similarity measures in our experiments, including Pearson correlation (pcc), constrained Pearson correlation (cpc), cosine similarity (cos) and extend Jaccard coefficient (jacc) which can be shown by Equations 4 - 7, respectively. In the equations below, we use m and n to denote two vectors, while L tells the length of the vector, $\bar{n}$ refers to the mean value in vector n, and $\tilde{r}$ denotes the median value of the rating scale. We use $m \cdot n$ to denote the dot product of two vectors, and $||n||$ refers to the mode of the vector n.

$$pcc(m,n) = \frac{\sum_{i=1}^{L}(m_i - \bar{m})(n_i - \bar{n})}{\sqrt{\sum_{i=1}^{L}(m_i - \bar{m})^2}\sqrt{\sum_{i=1}^{L}(n_i - \bar{n})^2}} \tag{4}$$

$$cpc(m,n) = \frac{\sum_{i=1}^{L}(m_i - \tilde{r})(n_i - \tilde{r})}{\sqrt{\sum_{i=1}^{L}(m_i - \tilde{r})^2}\sqrt{\sum_{i=1}^{L}(n_i - \tilde{r})^2}} \tag{5}$$

$$cos(m,n) = \frac{m \cdot n}{||m|| \times ||n||} \tag{6}$$

$$jacc(m,n) = \frac{m \cdot n}{||m||^2 + ||n||^2 - m \cdot n} \tag{7}$$

4.2 Non-Dominated Differential Context Weighting (ND-DCW)

The non-dominated neighbors can be found in the DCW approach too. Recall that we assign and learn a weight vector in which we have the weights for each context dimension. In ND-DCR, we can build the UC rating matrix by grouping the conditions by selected dimensions. However, different conditions share a same weight if they are from a same dimension. In this case, we can create two UC rating matrices – half of the conditions of a dimension will be set as columns in the first matrix, and remaining ones in the second matrix. One example can be shown by Table 6. Each condition will inherit the weight from the associated context dimension.

In addition to the weighted Jaccard similarity shown in Equation 3, we use weighted cosine similarity (wcos) and weighted correlation (wcorr) to calculate user-user similarities based on these UC matrices, as shown by Equation 8 - 11. The *wcov* and *wavg* are used to denote the weighted covariance and weighed average functions in the equations below. We name this method as "ND-DCW" accordingly.

$$wcos(m,n,w) = \frac{\sum_{i=1}^{L} w_i m_i n_i}{\sqrt{\sum_{i=1}^{L} w_i m_i^2}\sqrt{\sum_{i=1}^{L} w_i n_i^2}} \tag{8}$$

$$wcorr(m,n,w) = \frac{wcov(m,n,w)}{\sqrt{wcov(m,m,w)wcov(n,n,w)}} \tag{9}$$

$$wcov(m,n,w) = \frac{\sum_{i=1}^{L} w_i (m_i - wavg(m,w))(n_i - wavg(n,w))}{\sum_{i=1}^{L} w_i} \tag{10}$$

$$wavg(m,w) = \frac{\sum_{i=1}^{L} w_i m_i}{\sum_{i=1}^{L} w_i} \tag{11}$$

Table 6 UC Matrices by ND-DCW

	Time: weekend	Location: home
u_1	3.3	3.0
u_2	3.5	3.1
u_3	...	...
...	...	...
u_M	...	...

	Time: weekday	Location: cinema
u_1	3.2	4.0
u_2	3.0	3.2
u_3	...	...
...	...	...
u_M	...	...

Alternatively, we can merge all conditions together to have a single UC matrix as shown by Table 5. Non-dominated neighbors will be found by using the user-user similarities based on the UI and UC rating matrices only. Other components will be computed as the same as the ones in DCW. We use ND$_s$-DCW to denote this simplified method.

As a summary, we propose four DCM approaches in this paper. All these approaches identify non-dominated neighbors by using user-user similarities from different perspectives:

- ND-DCR in which we measure user-user similarities based on the UI and UC rating matrices. The number of UC rating matrices depends on the number of selected context dimensions in the process of context relaxation.
- ND$_s$-DCR is similar to ND-DCR, but there is only a single UC rating matrix with conditions associated with all the selected context dimensions.
- ND-DCW in which we computer weighted user similarities based on the UI and two UC rating matrices, where we mix the conditions from different context dimensions to build the UC matrices. Each condition in the UC matrix inherits the weight from the associated context dimension.
- ND$_s$-DCW is similar to ND-DCW, but we only have a single UC rating matrix.

5 Experiments and Results

In this section, we introduce the contextual data sets, evaluation protocols, and our experimental results.

5.1 Contextual Rating Data

Due to the difficulty in context collection, rating data sets with context information were usually collected from surveys. They are either small or sparse. We use five real-world data sets as shown in Table 7.

Table 7 List of Contextual Data Sets

	Food	**Restaurant**	**CoMoDa**	**STS**	**Frappe**
# of users	212	50	121	325	957
# of items	20	40	1232	249	4082
# of context dimensions	2	2	8	11	3
# of context conditions	8	7	37	53	14
# of ratings	6360	2309	2292	2354	87580
Rating scale	1-5	1-5	1-5	1-5	0-4.46
Density	9.4%	9.6%	1.4e-7	1.3e-9	9.4e-5

- The *Food* data [26] was collected from surveys in which the subjects were asked to give ratings on the Japan food menus in two contextual dimensions: degree of hungriness in real situations, and degree of hungriness in assumed or imagined situations. Typical context conditions in these two dimensions are full, hungry, normal. This is a good data set for exploring contextual preferences, since each users gave multiple ratings on a same item in different contexts.
- The *Restaurant* data [31] is also a data set collected from survey. Subjects gave ratings to the popular restaurants in Tijuana, Mexico by considering two contextual variables: time and location.
- The *CoMoDa* data [21] is a publicly available context-aware movie data collected from surveys. There are 12 context dimensions which captured users' various situations, including mood, weather, time, location, companion, etc.
- The *South Tyrol Suggests (STS)* data [8] was collected from a mobile app which provides context-aware suggestions for attractions, events, public services, restaurants, and much more for South Tyrol. There are 14 contextual dimensions in total, such as budget, companion, daytime, mood, season, weather, etc.
- The *Frappe* data [33] comes from the mobile usage in the app named as Frappe which is a context-aware app discovery tool that will recommend the right apps for the right moment. We used three context dimensions for experimental evaluations, including time of the day, day of the week and location. This data captures the frequencies of an app used by each user within two months.

5.2 Evaluation Protocols

We apply 5-fold cross validation on all the data sets. Due to the fact that we use sum of squared errors as the fitness in the optimizer (e.g., BPSO, PSO), we evaluate these recommendation models on the rating prediction task by using mean absolute error (MAE) and root mean squared error (RMSE) as shown below, while T_e refers to the test set.

$$MAE = \frac{1}{|T_e|} \sum_{(u,i,c) \in T_e} |r_{u,i,c} - \hat{r}_{u,i,c}| \tag{12}$$

$$RMSE = \sqrt{\frac{1}{|T_e|} \sum_{(u,i,c) \in T_e} (r_{u,i,c} - \hat{r}_{u,i,c})^2} \tag{13}$$

In addition to the DCM (i.e., DCR and DCW) and Non-dominated DCM approaches, we compare them with the following baseline recommendation algorithms:

- UBCF [32] and MF [20] which are the traditional collaborative filtering techniques without considering contexts.
- SPF [12] and UISplitting [40] which are two contextual pre-filtering models. After the pre-filtering, we use MF on the remaining rating matrix.
- CAMF [6] and TF [17] which are two contextual modeling based on the factorization technique.
- CAMF_ICS [42] which incorporates context similarity into the MF approach.
- DCW-GA [22] which is a CACF method proposed recently. It was demonstrated to be able to improve the DCW by filling in missing ratings and using a genetic algorithm as the optimizer.

We evaluated these recommendation models with the help of CARSKit [41] which is a Java-based open-source library for context-aware recommendations.

5.3 Experimental Results

The results based on MAE and RMSE can be shown by Table 8 and Table 9, respectively. In terms of the DCM and ND-DCM approaches, we only present the best results in the tables. The optimal choice of similarity measures will be discussed later (e.g., Figure 3 and Figure 4). The ND_s-DCM approaches perform the best among our proposed models, and we additionally present the improvement ratio over the base method (e.g., DCR or DCW) in the table. We use $*$ to indicate the significant results by ND_s-DCM in comparison with the base DCM method, and use $\circ$ to denote the significant better results by comparing the ND_s-DCM approaches and the best performing baseline methods (i.e., SPF for the restaurant data, and UISplitting for other data sets). The significance was examined by using the two-sample t-test at the 95% confidence level.

Among the baseline approaches, UISplitting is the overall winner for all the data. More specifically, it outperforms all other baselines, except the RMSE on the food data where CAMF works better, and prediction errors on the restaurant data where SPF is better. However, there are no significant differences among these errors. Therefore, we consider UISPlitting as the best performing model among these standard baseline approaches.

By comparing the results by the DCM and ND-DCM recommendation models, we can observe that the ND-DCM methods can outperform the original DCM approaches significantly, except the ND-DCR model on the STS data. More specifically, the ND_s-DCM methods are the overall winner. The ND_s-DCR can improve the DCR method by over 4% on the food, restaurant and CoMoDa data sets in

Table 8 MAE Results

Algorithms			Food	Restaurant	CoMoDa	STS	Frappe
Baselines	UBCF		0.947	0.991	0.870	0.985	0.404
	MF		0.923	0.978	0.821	0.912	0.381
	SPF		0.900	**0.808**	0.819	0.900	0.382
	UISplitting		**0.805**	0.813	**0.775**	**0.893** ∘	**0.378**
	CAMF		0.845	0.860	0.795	1.019	0.398
	CAMF_ICS		0.858	0.825	0.777	0.986	0.388
	TF		0.966	0.945	0.858	0.916	0.392
DCR	DCR		0.772	0.833	0.791	0.954	0.394
	ND-DCR		0.740	**0.784**	0.732	0.955	0.390
	ND_s-DCR	MAE	**0.740** ∗ ∘	0.787 ∗ ∘	**0.726** ∗ ∘	**0.934** ∗	**0.386** ∗
		Impr/DCR	4.1%	5.5%	8.2%	2.1%	2.0%
DCW	DCW		0.755	0.758	0.740	0.947	0.388
	DCW-GA		0.744	0.749	0.735	0.942	0.388
	ND-DCW		0.727	0.739	0.731	0.928	0.385
	ND_s-DCW	MAE	**0.725** ∗ ∘	**0.735** ∗ ∘	**0.714** ∗ ∘	**0.923** ∗	**0.379** ∗
		Impr/DCW	4.0%	3.0%	3.5%	2.5%	2.3%

Table 9 RMSE Results

Algorithms			Food	Restaurant	CoMoDa	STS	Frappe
Baselines	UBCF		1.183	1.230	1.126	1.243	0.508
	MF		1.153	1.185	1.034	1.171	0.480
	SPF		1.237	**1.111**	1.036	1.196	0.528
	UISplitting		1.072	1.124	**0.972**	**1.149** ∘	**0.478**
	CAMF		**1.060**	1.118	1.001	1.151	0.480
	CAMF_ICS		1.075	1.172	1.011	1.296	0.532
	TF		1.220	1.139	1.104	1.155	0.514
DCR	DCR		1.008	1.109	1.016	1.244	0.491
	ND-DCR		0.979	1.090	0.939	1.199	0.488
	ND_s-DCR	RMSE	**0.978** ∗ ∘	**1.058** ∗ ∘	**0.938** ∗ ∘	**1.181** ∗	**0.484** ∗
		Impr/DCR	3.0%	4.5%	7.6%	5.1%	1.5%
DCW	DCW		1.006	1.035	0.970	1.214	0.489
	DCW-GA		0.988	1.022	0.951	1.208	0.490
	ND-DCW		0.972	1.002	0.927	1.187	0.479
	ND_s-DCW	RMSE	**0.972** ∗ ∘	**0.997** ∗ ∘	**0.927** ∗ ∘	**1.170** ∗	**0.478** ∗
		Impr/DCW	3.3%	3.7%	4.4%	3.6%	2.2%

terms MAE and RMSE, except a small decline on the food data in RMSE (i.e., 3% improvement on RMSE for the food data). The improvement on the Frappe data is only around 2%, but our significance tests confirm the advantages by our ND_s-DCR model. For example, the p-value by the t-test for DCR and ND_s-DCR is smaller than $6.6e^{-9}$ on the Frappe data.

Similar patterns can be found by comparing DCW and ND_s-DCW methods. Particularly, the ND_s-DCW methods can improve the ND_s-DCR approaches by further reducing the prediction errors. It is not surprising since DCW can further alleviate the sparsity issue by filtering the rating profiles using context similarity. Most of the results between ND-DCM and ND_s-DCM methods are not significant, except the results by on ND_s-DCW the CoMoDa data and the results by ND_s-DCR on the STS data. In addition, the baseline DCW-GA which fills in missing ratings and utilizes genetic algorithm as the optimizer can beat our proposed DCR methods in some cases. For example, DCW-GA can achieve lower MAE and RMSE

on the restaurant data, and lower MAE on the STS data, in comparison with ND-DCR. It is not surprising, since the context weighting strategy is a finer-grained method and usually better than context relaxations. However, DCW-GA fails to outperform our proposed ND-DCW methods, though there are no significant differences between the MAE on the CoMoDa data and RMSE on the STS data, in comparison between DCW-GA and ND-DCW. The finer-grained ND_s-DCW method can eventually outperform DCW-GA in terms of both MAE and RMSE.

In comparison with the best performing baseline approach (i.e., UISplitting), ND_s-DCW is still the best model on the food, restaurant and CoMoDa data sets. ND_s-DCM failed to beat UISplitting on the STS data, while it obtains similar results with UISplitting on the Frappe data (i.e., no significance on the MAEs). As mentioned in Section 2.2, latent-factor models were proposed and designed to alleviate the sparsity issue in collaborative filtering. In our experiments above, we utilize matrix factorization model after the splitting operations by UISplitting, while our ND-DCM approaches are all built upon UBCF which is a memory-based collaborative filtering model. That is one possible reason why UISplitting outperforms our proposed ND-DCM approaches. To validate this assumption, we run UISplitting again, but use UBCF as the recommendation model. The resulting MAE and RMSE are 0.935 and 1.199 respectively. Our ND_s-DCW approaches, especially the ND_s-DCW method can outperform the UISplitting significantly, where all of them use the UBCF as the fundamental recommendation model. As a summary, the result demonstrates that our proposed model can still work better than UISplitting if the recommendation approaches were built on the same UBCF model. By using matrix factorization in UISplitting, the advantage of latent-factor models can further alleviate the sparsity issue and deliver better performance than our proposed model on the STS data.

It is worth mentioning that the performance by these ND-DCM methods is sensitive to the similarity measure used in the computation of user-user similarities. It is because the dominance relationship relies on these user-user similarities. The impact on MAE by these measures can be observed from Figure 3 and Figure 4 by using the ND_s-DCR and ND_s-DCW, respectively. In our experiments, we did not find a unique winner as the best similarity measure. The best choice may vary from data to data.

6 Conclusion and Future Work

The differential context modeling were developed to alleviate the sparsity issue by using partial matching or filtering based on weighted context similarity. In this paper, we propose to improve differential context modeling by identifying and using the non-dominated neighborhood by using user-user similarities based on contextual preferences. Our experimental results demonstrate the effectiveness of these non-dominated DCM approaches, and they can even work better than the context-aware collaborative filtering based on the matrix factorization technique in the rating prediction task.

The idea of non-dominated neighbors is interesting and promising in the memory-based collaborative filtering. In our future work, we will seek other methods to find better non-dominated neighborhood. Note that the number of identified non-dominated neighbors may vary from data to data. If there are several non-

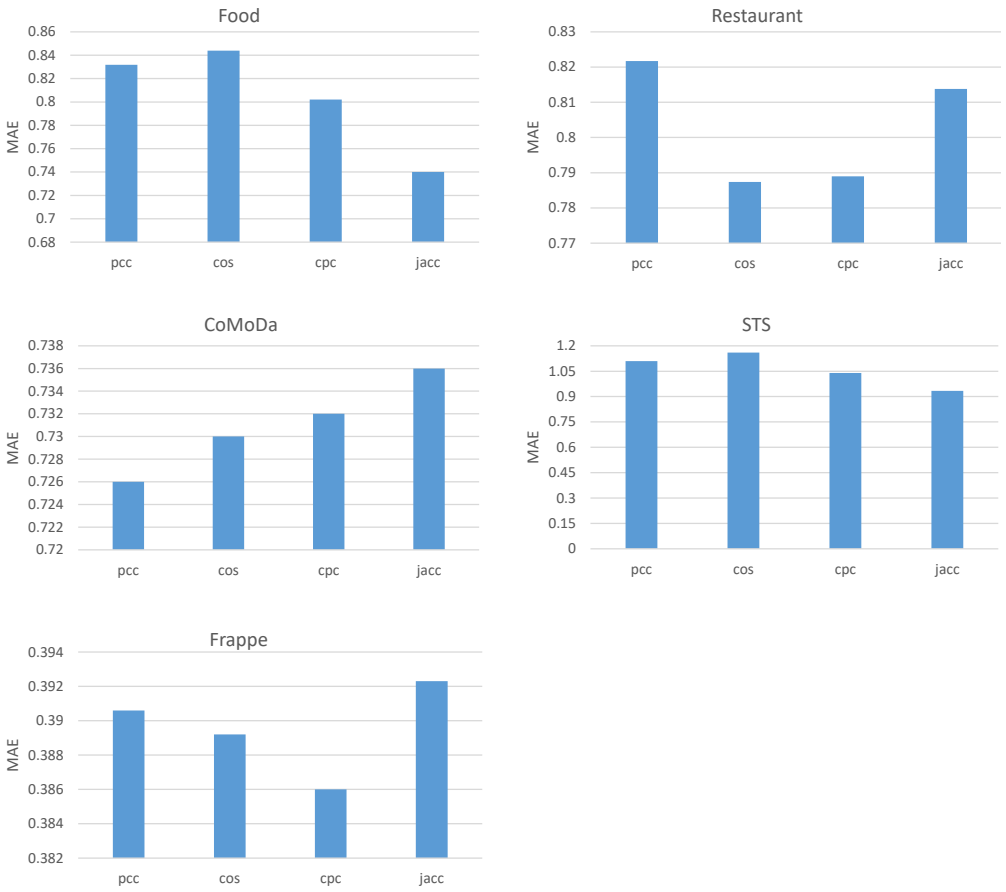


Fig. 3 MAE Results by ND_s-DCR Using Different Similarity Measures

dominated neighbors, we can setup a similarity threshold to select the top-K ones. Or, we can setup a threshold for the number of dominance, if there are limited number of non-dominated neighbors. We will exploit this option or tune up this parameter in our future work. Moreover, the quality of the non-dominated neighbors depends on the similarity measures. More powerful measures [25] may be helpful to further improve the quality of these non-dominated neighbors.

Author's contributions

The paper and the research on which the paper is based on are of individual effort. The author read and approved the final manuscript.

Conflict of interest

The authors declare that they have no conflict of interest.

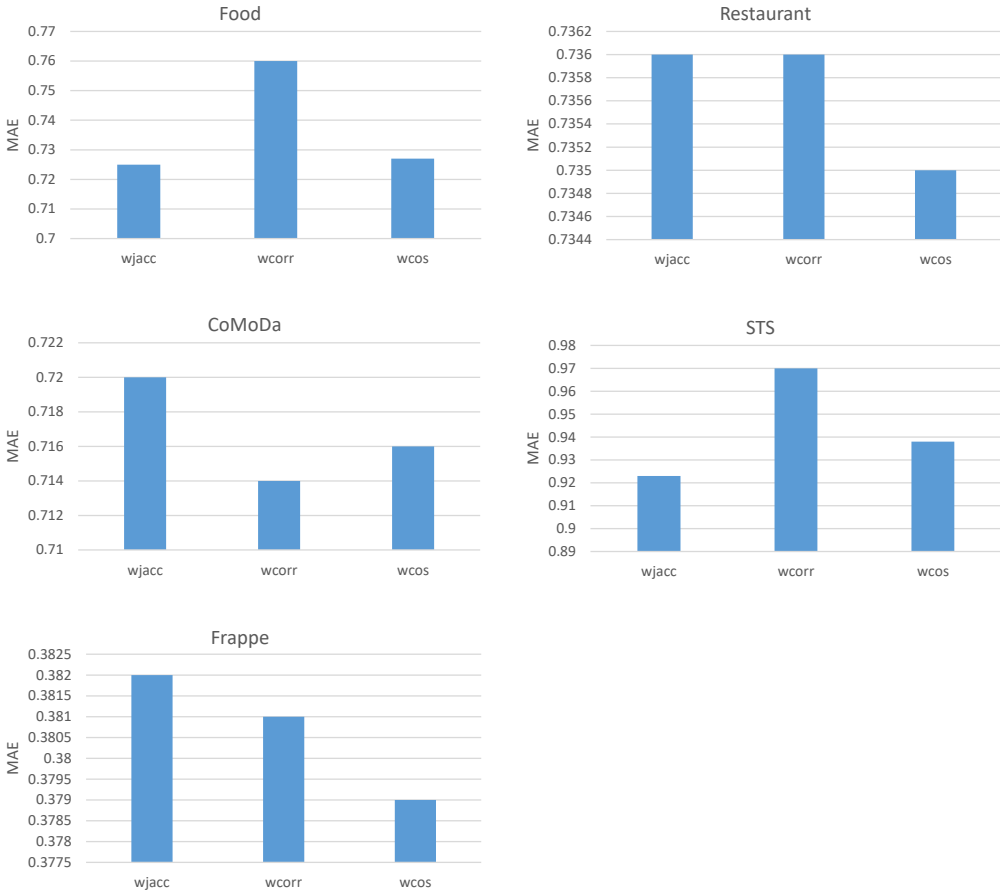


Fig. 4 MAE Results by ND_s-DCW Using Different Similarity Measures

Availability of data and materials

Most of the data are available at https://github.com/irecsys/CARSKit/

Funding

Not applicable.

Acknowledgements

Not applicable.

References

1. Abowd, G.D., Dey, A.K., Brown, P.J., Davies, N., Smith, M., Steggles, P.: Towards a better understanding of context and context-awareness. In: Handheld and ubiquitous computing, pp. 304–307. Springer (1999)
2. Adomavicius, G., Mobasher, B., Ricci, F., Tuzhilin, A.: Context-aware recommender systems. AI Magazine **32**(3), 67–80 (2011)
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. Ali, W., Din, S.U., Khan, A.A., Tumrani, S., Wang, X., Shao, J.: Context-aware collaborative filtering framework for rating prediction based on novel similarity estimation. Computers, Materials & Continua **63**(2), 1065–1078 (2020)
5. Baltrunas, L., Amatriain, X.: Towards time-dependant recommendation based on implicit feedback. In: ACM RecSys, the 4th Workshop on Context-Aware Recommender Systems (2009)
6. 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)
7. Baltrunas, L., Ricci, F.: Experimental evaluation of context-dependent collaborative filtering using item splitting. User Modeling and User-Adapted Interaction **24**(1-2), 7–34 (2014)
8. Braunhofer, M., Elahi, M., Ricci, F., Schievenin, T.: Context-aware points of interest suggestion with dynamic weather data management. In: Information and Communication Technologies in Tourism 2014, pp. 87–100. Springer (2013)
9. Burke, R.: Hybrid recommender systems: Survey and experiments. User Modeling and User-Adapted Interaction **12**(4), 331–370 (2002)
10. del Carmen Rodríguez-Hernández, M., Ilarri, S.: Ai-based mobile context-aware recommender systems from an information management perspective: Progress and directions. Knowledge-Based Systems **215**, 106740 (2021)
11. Chen, A.: Context-aware collaborative filtering system: predicting the user's preferences in ubiquitous computing. In: CHI'05 Extended Abstracts on Human Factors in Computing Systems, pp. 1110–1111 (2005)
12. Codina, V., Ricci, F., Ceccaroni, L.: Distributional semantic pre-filtering in context-aware recommender systems. User Modeling and User-Adapted Interaction **26**(1), 1–32 (2016)
13. Coello, C.C.: Evolutionary multi-objective optimization: a historical view of the field. IEEE computational intelligence magazine **1**(1), 28–36 (2006)
14. Deb, K.: Multi-objective optimization. In: Search methodologies, pp. 403–449. Springer (2014)
15. Giagkiozis, I., Fleming, P.J.: Methods for multi-objective optimization: An analysis. Information Sciences **293**, 338–350 (2015)
16. Jannach, D., Mobasher, B., Berkovsky, S.: Research directions in session-based and sequential recommendation. User Modeling and User-Adapted Interaction **30**(4), 609–616 (2020)
17. 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)
18. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Neural Networks, 1995. Proceedings., IEEE International Conference on, vol. 4, pp. 1942–1948. IEEE (1995)
19. Kennedy, J., Eberhart, R.: A discrete binary version of the particle swarm algorithm. In: Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation., 1997 IEEE International Conference on, vol. 5, pp. 4104–4108. IEEE (1997)
20. Koren, Y., Bell, R., Volinsky, C.: Matrix factorization techniques for recommender systems. IEEE Computer **42**(8), 30–37 (2009)
21. Košir, A., Odic, A., Kunaver, M., Tkalcic, M., Tasic, J.F.: Database for contextual personalization. ELEKTROTEHNISKI VESTNIK **78**(5), 270–274 (2011)
22. Linda, S., Minz, S., Bharadwaj, K.K.: Effective context-aware recommendations based on context weighting using genetic algorithm and alleviating data sparsity. Applied Artificial Intelligence **34**(10), 730–753 (2020)
23. Lops, P., De Gemmis, M., Semeraro, G.: Content-based recommender systems: State of the art and trends. Recommender systems handbook pp. 73–105 (2011)

24. Ludewig, M., Mauro, N., Latifi, S., Jannach, D.: Empirical analysis of session-based recommendation algorithms. User Modeling and User-Adapted Interaction **31**(1), 149–181 (2021)
25. Manochandar, S., Punniyamoorthy, M.: A new user similarity measure in a new prediction model for collaborative filtering. Applied Intelligence **51**(1), 586–615 (2021)
26. Ono, C., Takishima, Y., Motomura, Y., Asoh, H.: Context-aware preference model based on a study of difference between real and supposed situation data pp. 102–113 (2009)
27. Ortega, F., Sanchez, J.L., Bobadilla, J., Gutierrez, A.: Improving collaborative filtering-based recommender systems results using pareto dominance. Information Sciences **239**, 50–61 (2013)
28. Özsoy, M.G., Polat, F., Alhajj, R.: Multi-objective optimization based location and social network aware recommendation. In: 10th IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing, pp. 233–242. IEEE (2014)
29. Panniello, U., Gorgoglione, M.: Context-aware recommender systems: A comparison of three approaches. In: DART@ AI* IA (2011)
30. Pazzani, M.J., Billsus, D.: Content-based recommendation systems. In: The adaptive web, pp. 325–341. Springer (2007)
31. Ramirez-Garcia, X., Garcia-Valdez, M.: Post-filtering for a restaurant context-aware recommender system. In: Recent Advances on Hybrid Approaches for Designing Intelligent Systems, pp. 695–707. Springer (2014)
32. Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., Riedl, J.: Grouplens: an open architecture for collaborative filtering of netnews. In: Proceedings of the 1994 ACM conference on Computer supported cooperative work, pp. 175–186. ACM (1994)
33. Shi, Y., Karatzoglou, A., Baltrunas, L., Larson, M., Hanjalic, A.: Cars2: Learning context-aware representations for context-aware recommendations. In: proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, pp. 291–300 (2014)
34. Unger, M., Tuzhilin, A., Livne, A.: Context-aware recommendations based on deep learning frameworks. ACM Transactions on Management Information Systems (TMIS) **11**(2), 1–15 (2020)
35. Wu, L., Quan, C., Li, C., Wang, Q., Zheng, B., Luo, X.: A context-aware user-item representation learning for item recommendation. ACM Transactions on Information Systems (TOIS) **37**(2), 1–29 (2019)
36. Zhao, J., Yang, S., Huo, H., Sun, Q., Geng, X.: Tbtf: an effective time-varying bias tensor factorization algorithm for recommender system. Applied Intelligence pp. 1–12 (2021)
37. Zheng, Y.: Context-aware mobile recommendation by a novel post-filtering approach. In: FLAIRS Conference, pp. 482–485 (2018)
38. Zheng, Y., Burke, R., Mobasher, B.: Differential context relaxation for context-aware travel recommendation. In: E-Commerce and Web Technologies, Lecture Notes in Business Information Processing, pp. 88–99. Springer Berlin Heidelberg (2012)
39. Zheng, Y., Burke, R., Mobasher, B.: Recommendation with differential context weighting. In: User Modeling, Adaptation, and Personalization, Lecture Notes in Computer Science, pp. 152–164. Springer Berlin Heidelberg (2013)
40. 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)
41. Zheng, Y., Mobasher, B., Burke, R.: Carskit: A java-based context-aware recommendation engine. In: Proceedings of the 15th IEEE International Conference on Data Mining Workshops. IEEE (2015)
42. Zheng, Y., Mobasher, B., Burke, R.: Similarity-based context-aware recommendation. In: International Conference on Web Information Systems Engineering, pp. 431–447. Springer (2015)