information

Article

Context-Aware Collaborative Filtering Using Context Similarity: An Empirical Comparison

Yong Zheng

Department of Information Technology and Management, College of Computing, Illinois Institute of Technology, Chicago, IL 60616, USA; yzheng66@iit.edu

Abstract: Recommender systems can assist with decision-making by delivering a list of item recommendations tailored to user preferences. Context-aware recommender systems additionally consider context information and adapt the recommendations to different situations. A process of context matching, therefore, enables the system to utilize rating profiles in the matched contexts to produce context-aware recommendations. However, it suffers from the sparsity problem since users may not rate items in various context situations. One of the major solutions to alleviate the sparsity issue is measuring the similarity of contexts and utilizing rating profiles with similar contexts to build the recommendation model. In this paper, we summarize the context-aware collaborative filtering methods using context similarity, and deliver an empirical comparison based on multiple context-aware data sets.

Keywords: recommender systems; context-aware; context similarity; collaborative filtering

check for updates

Citation: Zheng, Y. Context-Aware Collaborative Filtering Using Context Similarity: An Empirical Comparison. *Information* **2022**, *13*, 42. https://doi.org/10.3390/info13010042

Academic Editors: Marco Polignano and Giovanni Semeraro

Received: 21 November 2021
Accepted: 10 January 2022
Published: 17 January 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

1. Introduction

The problem of information overload [1,2] refers to the situation that a person can have difficulty in understanding an issue and making decisions that can be caused by the presence of too much information. Recommender systems have been demonstrated to alleviate information overload by producing recommendations tailored to user preferences. Several recommendation algorithms have been built for item recommendations by exploiting users' preference history, such as collaborative filtering [3,4], content-based recommenders [5], hybrid approaches [6], and so forth.

In contrast to the traditional recommender systems, context-aware recommender systems (CARS) [7,8] were proposed and developed in order to adapt the recommendations to users' preferences in different contextual situations. The underlying assumption is that a user may make distinct decisions in different situations. For example, a user may choose a different type of movie when he or she will watch the movie *with a partner* rather than *with kids*. Furthermore, a user may choose a formal restaurant for *a business dinner*, but a fast-food restaurant may be enough for having a *quick lunch* alone. *Companion* and *occasion* are two context variables in these examples that may affect a user's choice of the items.

Several recommendation algorithms were proposed for CARS, such as pre-filtering [7,9,10], post-filtering [11,12], and contextual modeling [13–15] approaches. Taking pre-filtering approaches for example, we can use contexts to filter out irrelevant rating profiles, so the traditional recommendation algorithms can be applied to the remaining ratings. To predict a user's rating on a movie in contexts "at a cinema on a weekend with family", the model expects to use the ratings on the same item in the same contexts. However, the items may not be rated in the exact contexts for multiple times, which results in the sparsity issue in CARS. Researchers proposed to build different latent-factor models [13,14] to alleviate this issue, but we may have difficulties to discover the insights or interpret the recommendations in these models. Another popular solution is measuring context similarities so that rating profiles with similar contexts can be utilized to build the recommendation model. The similarity of the contexts can also be used to interpret the model or the contextual effects [16].

In this paper, we deliver a mini review about context-aware collaborative filtering (CACF) using context similarity. The major contributions can be summarized as follows.

- Using context similarity is one of the major solutions to alleviate the sparsity issue in CARS. In this paper, we summarize different approaches to measure the context similarity, and discuss existing CACF approaches using context similarity.
- We deliver an empirical comparison among these recommendation algorithms, including some classical CACF approaches that were proposed at the early stage, but not compared with any existing research, such as the Chen's method [17] in 2005.

The remainder of the article is organized as follows: Section 2 briefly reviews context-aware recommender systems and collaborative filtering. Section 3 introduces the classical collaborative filtering as a preliminary. Section 4 illustrates and summarizes different CACF approaches. Section 5 presents our empirical comparison among these CACF methods, followed by the conclusion and future work in Section 6.

2. Related Work

In this section, we deliver the background of CARS, introduce CACF approaches, and illustrate the sparsity issue in CARS.

2.1. Context-Aware Recommender Systems

Traditional recommendation problem can be modeled as a two-dimensional (2D) prediction—R: *Users* $\times$ *Items* $\rightarrow$ *Ratings*, where the recommender system's task is to predict that user's rating for that item. By contrast, context-aware recommender systems try to additionally incorporate contexts to estimate user preferences, which turns the prediction into a "*multi-dimensional*" (MD) rating function—R: *Users* $\times$ *Items* $\times$ *Contexts* $\rightarrow$ *Ratings* [8]. In other words, CARS tries to adapt to user's preferences in different contextual situations. The appropriate recommendations should be produced by taking context information into account, since users' tastes may vary from context to context.

Context is 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" [18]. G. Adomavicius et al. [8] believed that context variables can be split according to two factors—what a recommender system knows about a contextual factor (i.e., fully observable, partially observable, and unobservable) and how the contextual factor changes over time (i.e., static, or dynamic). Therefore, CARS can be classified into two categories—one is representational CARS in which the contextual variables are fully observed and static. The other is interactional CARS in which contextual factors are not fully observed and usually change dynamically in the process of interactions among users, items, and contexts, such as the session-based recommender systems [19,20]. In this paper, we limit our discussion to the representational CARS.

Some recent development of CARS incorporated additional information into the recommendation models to obtain further improvements. These additional information include but not limited to trust information [21], social network [22], tag-based folksonomy [23], and so forth. In this paper, we only discuss the CARS models which were built based on users' contextual ratings on the items.

2.2. Context-Aware Collaborative Filtering

Traditional recommender systems usually use a variety of ways to adapt to user preferences, such as collaborative [3,24], content-based [5], knowledge-based [25], and hybrid [6] approaches. In CARS, researchers tried to incorporate contexts into these traditional recommendation approaches, especially the collaborative filtering techniques.

Adomavicius [26] pointed out that there were usually three ways to incorporate contexts into recommender systems. The *contextual pre-filtering* approaches [7,9,10] use contexts as filters to filter out irrelevant rating profiles, and any traditional recommendation algorithms can be applied to the remaining ratings to produce the recommendation list.

The first approach in this category was the reduction-based approach or exact-filtering [7] that performs an exact matching by using the whole or the reduced context variables. *Contextual post-filtering* [11,12] methods apply a recommender first and then use contexts as filters to filter out irrelevant recommendations. By contrast, contexts are considered as one part in the prediction function in the contextual modeling approaches to build MD recommenders directly, such as tensor factorization (TF) [14] and context-aware matrix factorization (CAMF) [13].

Most of the existing research incorporates contexts into the memory-based [3,27] and model-based [4,28,29] collaborative filtering by using the three methods above. With the development of deep learning-based recommendation models [30], CARS have been built on top of neural networks too. Most efforts by deep learning were devoted to the interactive CARS, such as the session-based recommender systems [19,20]. Few of them [31–33] were applied to representational CARS. However, none of these deep learning-based models took advantage of context similarities.

2.3. Sparsity Issue in CARS

As mentioned before, a user may not rate items in different contexts for several different times. This is well-known as the sparsity issue in CARS. For example, using the exact matching by contexts may lead to limited or even no rating profiles left. As a result, the recommendation models may not provide accurate or reliable recommendations, or even do not work if there are no matched rating profiles.

There could be three solutions to alleviate this sparsity issue: First, we can reduce the number of context variables. However, we may mistakenly remove contexts that are useful. Moreover, the latent-factor models, such as the CARS algorithms based on matrix factorization [13] or tensor factorization [14], can alleviate the sparsity issue. However, these models may reduce the degree of transparency or the explainability of the recommendations. In addition, researchers seek ways to measure context similarity, so that a larger proportion of the rating profiles can be utilized in the recommendation models. The context similarity can be utilized in a similar way to explain the recommendations as to the user–user or item–item similarities in the neighborhood-based collaborative filtering (NBCF). We will summarize and discuss these approaches in Section 4.

Our work in this paper specifically focuses on context-aware collaborative filtering using context similarities in which we highlight the importance of context similarities in alleviating the sparsity of CARS and provide an empirical comparison among related techniques.

3. Preliminary: Collaborative Filtering

In this section, we discuss the classical collaborative filtering algorithms for the traditional recommender systems.

3.1. Memory-Based Collaborative Filtering

The neighborhood-based collaborative filtering, such as user-based collaborative filtering (UBCF) [3] and item-based collaborative filtering [27], are the two most popular memory-based collaborative filtering. Taking UBCF for example, it assumes a user's preference on one item is close to a group of users' taste on the same item. This group of users should share similar preferences with the target user and is usually named as user neighborhood.

$$\hat{r}_{u,i} = \bar{r}_u + \frac{\sum\limits_{a \in N_u} (r_{a,i} - \bar{r}_a) \times sim(a,u)}{\sum\limits_{a \in N_u} sim(a,u)} \tag{1}$$

The standard prediction function in UBCF can be described by Equation (1), 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 suffers from the sparsity issue, since user-user similarities may not be reliable if the number of co-rated items is limited.

3.2. Model-Based Collaborative Filtering

Matrix factorization (MF) [4] is one of the most effective model-based collaborative filtering in the traditional recommender systems. In MF, both users and items are represented by latent-factor 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 Equation (2).

$$\hat{r}_{u,i} = \overrightarrow{p_u} \cdot \overrightarrow{q_i} \tag{2}$$

The values in $\overrightarrow{p_u}$ can be considered as the weights on the factors that why the user u likes an item, while the values in $\overrightarrow{q_i}$ can represent how much an item i obtains these factors or characteristics. Therefore, the dot product in Equation (2) can represent how user u likes the item i.

$$\hat{r}_{u,i} = \mu + b_u + b_i + \overrightarrow{p_u} \cdot \overrightarrow{q_i} \tag{3}$$

$$\underset{p*,q*,b*}{Minimize} \sum_{(u,i)\in T_r} (r_{u,i} - \hat{r}_{u,i})^2 + \lambda(||\overrightarrow{p_u}||^2 + ||\overrightarrow{q_i}||^2 + b_u^2 + b_i^2) \tag{4}$$

Koren et al. [4] suggests adding the global rating (i.e., μ), user bias (i.e., b_u), and item bias (i.e., b_i) in the rating prediction, as shown by Equation (3), to produce better rating predictions. Stochastic gradient descent (SGD) or alternating least squares can minimize the sum of squared errors by learning the user and item vectors, as well as the user and item biases in the model. The loss function can be shown by Equation (4), where T_r is the rating data set and $r_{u,i}$ refers to the known or real rating. λ refers to the regularization rate assigned to the regularization term with L_2 norm, as shown by Equation (4).

4. Context-Aware Collaborative Filtering Using Context Similarity

In this section, we discuss different ways to compute or learn the context similarity and present the approaches that take advantages of context similarity to build better context-aware recommendation models. Finally, we discuss the advantages and disadvantages in Section 4.6.

Some studies did not use context similarity directly in their models but derive a similarity function to better calculate user–user [34–36] or item–item [36,37] similarities by considering contexts. These works were excluded from our discussions in this section.

According to our literature reviews, we classify these approaches using context similarity into four categories—semantic similarity, matching-based similarity, inferred similarity from ratings, and learned similarity representations—which can be described in Table 1. In addition, we indicate the type of the collaborative filtering (NBCF or MF) the context similarity can be fused to, and which category of CARS algorithm (pre-filtering or contextual modeling) they belong to.

We can observe that 9 out of 12 works built contextual modeling algorithms, and no existing research incorporates context similarity into post-filtering models. Among these four categories, only the work by Zheng et al. [38] proposed and learned different similarity representations, while others tried to compute the context similarity based on the semantic ontology [39] or existing ratings [7,10,40–42]. We discuss these techniques in the following sections, respectively.

Table 1. Summary of Context Similarity Approaches.

	References	Pre-Filtering	Contextual Modeling	NBCF	MF
Semantic Similarity	Liu et al. [39] Kolahkaj et al. [43]		✓	✓	
Matching-based Similarity	Adomavicius et al. [7],	✓		✓	
	Zheng et al. [40,41,44], Gupta et al. [45] Linda et al. [46]		✓	✓	
Inffered Similarity from Ratings	Chen [17]		✓	✓	
	Codina et al. [10] Ferdousi, et al. [42]	✓			✓
Learned Similarity Representations	Zheng et al. [38]		✓		✓

4.1. Terminology and Notations

To better describe the CARS algorithms, we introduce the terminology and notations in this Section. In Table 2, 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, or 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,* or *family*.

Table 2. Contextual Ratings on Movies.

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

The symbols or notations used in the following discussions are listed in Table 3.

Table 3. Notations.

Notation	Explanations
M, N, Z	the number of users, items, and context dimensions, respectively,
c, x, y	context situations
c^E	a special context situation, with all dimensions as empty or N/A values. a non-contextual rating can be viewed as a rating in c^E
$c_1, c_2, ...c_Z$	context conditions in the 1th, 2th, . . . , Zth dimension of the situation c
$r_{u,i,c}$	rating given by user u on item i in context situation c
r_{u,i,c_t}	rating given by user u on item i in context condition c_t
$r_{u,i}$	rating given by user u on item i without considering contexts
T_r, T_e	training and testing set, respectively,

4.2. Semantic Similarity

The notion of *semantic similarity* here only refers to the similarity of contexts from the perspective of semantics in the contextual variables and conditions, i.e., the similarity measured from the textual values. Taking companion at movie watching for example,

"with sisters" may be more similar to "with family" rather than "with colleagues" from the perspective of semantics. The work by Codina et al. [10] tried to infer context similarity from rating behaviors, where we consider their work as inferred similarities from ratings, instead of semantic similarities.

Liu et al. [39] proposed to treat numerical and nominal contexts differently in a case study of hotel recommendations. They used a ratio value to represent context similarity for the numerical context dimension and adopted an ontology to measure the context similarity for nominal variables. Note that most of the context variables are categorical in existing data sets for CARS. In their work, they built an ontology tree for the *companion* in the trip scenario, e.g., traveling with *children* may be more similar with having a trip with *spouse* rather than with *friends*. There are two concerns with this method—on one hand, two contexts with a high context similarity usually infer that users may have similar preferences in these two contextual situations, while it is not necessary to be consistent with the semantic similarity. Taking companion at movie watching above for example again, "with sisters" may be more similar to "with family" rather than "with colleagues" from the perspective of semantics. However, the rating behaviors for watching movie "with sisters" may be similar to "with colleagues", but different from "with family". On the other hand, the semantic method is difficult to be generalized since it may need domain knowledge to build an ontology.

Kolahkaj et al. [43] focused on the temporal and geographical contexts. More specifically, they used a time-decay function to measure the similarity between timestamps. By contrast, they measured the spatial distance for the geographical contexts, if we have the latitude and longitude information. A weighted sum of the temporal and geographical context similarity can be considered as the similarity between two context situations. They take advantage of the temporal and geographical semantics. However, temporal and geographical contexts are usually represented in nominal values in most of the existing context-aware data, e.g., a "time" dimension may have "weekend, weekday, morning, evening" as the conditions, while a "location" may have context conditions such as "at home or at work".

4.3. Matching-Based Similarity

We consider context matching as a special case in measuring context similarity. There are usually two approaches—*exact matching* and *weighted matching*. The reduction-based approach proposed by Adomavicius et al. [7] is an exact-matching method that filters out rating profiles not-matched with the given contexts. Afterwards, any traditional recommendation algorithms (e.g., UBCF, MF, etc.) can be applied to the remaining rating profiles. The differential context relaxation (DCR) by Zheng et al. [40] used the exact matching in relaxed contexts. They decomposed the rating prediction function in UBCF into four components—neighbor selection, neighbor contribution, calculation of user-user similarities, and average rating of the target user, as shown by Figure 1. For each component, they proposed to perform an exact matching on selected or relaxed context dimensions. Taking the data shown in Table 2 for example, we can use time and location in the exact matching for the neighbor selection process but use time and companion in the neighbor contribution component. They proposed to use binary particle swarm optimizer [47] to learn the best selected or relaxed context dimensions for each component, where the solution can be encoded by a binary vector, so that the value of one indicates that the dimension is selected, and zero tells the dimension is not selected in the context relaxation.

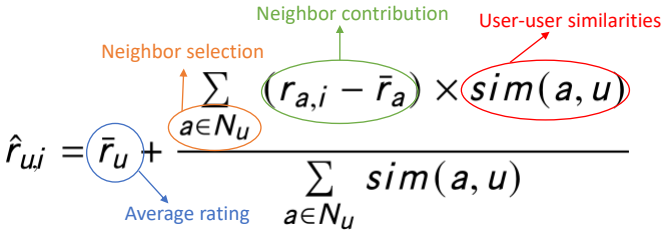

Figure 1. Four components in UBCF.

Zheng et al. [41] also proposed the weighted matching method which is referred to differential context weighting (DCW). Instead of an exact matching for the four components in the UBCF, they suggested to calculate context similarity and set a minimal threshold to have a set of rating profiles for each component. More specifically, they assigned a weight for each context dimension, and adopted a weighted Jaccard metric to calculate the context similarity, as shown by Equation (5), where x and y are two context situations, w is the weighting vector for context dimensions, Z is the number of context dimensions, and f refers to the index of matched context dimensions between x and y. Only the weights on the matched dimensions will be contributed to the similarity computations.

$$sim(x, y, w) = \frac{\sum_{f \in x \cap y} w_f}{\sum_{t=1}^{Z} w_t} \tag{5}$$

Accordingly, particle swarm optimizer [48] can be adopted to learn these weights in DCW. More specifically, the weights for each context dimension (i.e., w_t in Equation (5)) is encoded by a vector or real values which can further optimized by a process of population-based heuristic search (e.g., particle swarm optimizer or genetic algorithm).

There are several extensions to these DCR and DCW recommendation models. For example, Linda et al. [46] proposed to utilize real-coded genetic algorithm [49] to learn the weights in DCW. Gupta et al. [45] tried different weighted similarity measures (e.g., cosine similarity, Dice's coefficient, etc.) rather than the Jaccard metric, while they all are weighted matching based similarity, since nominal context conditions need to be transformed into binary values in the computations. Recently, the non-dominated DCR and DCW [44] were proposed. Context relaxation and context weighting were reused in these models, but they utilize a dominance relation to select better neighbors that can be referred as "non-dominated user neighbors". These neighbors are defined as the neighbors that dominate others from different perspectives of the user-user similarities, such as the user-user similarities based on co-rated items and co-rated context dimensions or conditions. They were demonstrated as better improvements over DCR and DCW in the rating prediction task.

4.4. Inferred Similarity from Ratings

Chen [17] first proposed to utilize context similarity in collaborative filtering in 2005, but these methods were not evaluated since there were no context-aware data sets available. It is surprising that the following research after 2005 did not evaluate Chen's methods too. Given two situations x and c, Chen proposed to measure similarity of them at tth dimension with respect to an item i by Equation (6). M refers to the number of users in the data, and r_{u,i,x_t} is used to denote a rating given by user u on item i in the context condition x_t, while $\bar{r}_i$ refers to the average rating on item i.

$$sim(c_t, x_t, i) = \frac{\sum_{u=1}^{M} (r_{u,i,c_t} - \bar{r}_i) \cdot (r_{u,i,x_t} - \bar{r}_i)}{\sqrt{\sum_{u=1}^{M} (r_{u,i,c_t} - \bar{r}_i)^2} \sqrt{\sum_{u=1}^{M} (r_{u,i,x_t} - \bar{r}_i)^2}} \tag{6}$$

Therefore, the predicted rating by u on item i in context situation c can be represented by Equation (7), while N_c denotes a set of similar context situations with c. The context

similarity between two situations with respect to i can be an average over all dimensions as shown by Equation (8).

$$\hat{r}_{u,i,c} = \frac{\sum_{x \in N_c} r_{u,i,x} \cdot sim(c,x,i)}{\sum_{x \in N_c} sim(c,x,i)} \tag{7}$$

$$sim(c,x,i) = \frac{1}{Z} \sum_{t=1}^{Z} sim(c_t, x_t, i) \tag{8}$$

Or the user neighborhood can be utilized to produce the final prediction as shown by Equation (9). If neighbor a's rating $r_{a,i,c}$ is not available, it can be estimated from Equation (7).

$$\hat{r}_{u,i,c} = \bar{r_u} + \frac{\sum_{a \in N_u} (r_{a,i,c} - \bar{r_a}) \cdot sim(a,u)}{\sum_{a \in N_u} sim(a,u)} \tag{9}$$

We use "Chen$_1$" to denote the rating prediction by Equation (7) and "Chen$_2$" for the method in Equation (9). Chen's methods suffer from sparsity issues seriously. In Equation (6), it is not guaranteed that a user has rating on an item i, not to mention the rating in context condition c_t, which results in inaccurate computation of context similarity with respect to item i in Equation (8).

Alternatively, Codina et al. [10] and Ferdousi, et al. [42] proposed to use a distributed vector to represent context conditions and dimensions. More specifically, the semantic pre-filtering (SPF) method by Codina et al. [10] first computed the influence of the condition c_t on the item i, denoted by w_{i,c_t}, as shown by Equation (10). We use R_{i,c_t} to denote the set of ratings on the item i in context condition c_t, and β is a decay factor.

$$w_{i,c_t} = \frac{1}{|R_{i,c_t}| + \beta} \sum_{r_{u,i,c_t} \epsilon R_{i,c_t}} (r_{u,i,c_t} - \hat{r}_{u,i}) \tag{10}$$

By this way, we can build a condition-item matrix in which rows are the unique context conditions in the data set, columns are the items, and the matrix can be filled by the influence of the condition c_t on the item it above. Each context condition is therefore represented by an influence vector over all items. A context situation c can be represented by the average vector over its condition vectors, and a cosine similarity is used to compute the similarity between two situations. A minimal similarity threshold can be set up to filter-out dissimilar rating profiles in the pre-filtering setting. Traditional recommendation algorithms, such as MF, can be applied to the remaining ratings to produce rating predictions or the list of recommendations.

Ferdousi, et al. [42] proposed a correlation-based pre-filtering (CBPF), which is a similar pre-filtering method with SPF. The influence of the condition c_t on the item it, w_{i,c_t}, was computed by the Pearson correlation between a rating entry in the data and the context condition c_t. After that, a context situation c is representation by a vector which is a concatenation of the condition vectors, as shown by Figure 2. The calculation of w_{i,c_t} in CBPF has significant computational costs since it must visit all rating entries and context conditions. Ferdousi, et al. suggested to compute w_{i,c_t} on an item-cluster basis, instead of an influence on the item basis [42]. They clustered the items over item content features and demonstrated that it was able to speed up the computation process.

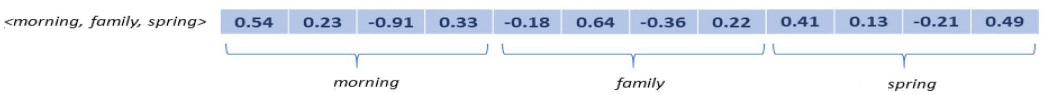

Figure 2. A context situation is a concatenation of the condition vectors.

All these methods in this category rely on the independent assumptions, so that the similarity of context situations can be aggregated from context conditions. There may be dependency among contexts. Given one example in movie watching, a user might prefer

to watch movie *at cinema* if it is *weekend*, since he or she may have time on the weekend to drive to the cinema.

4.5. Learned Similarity Representations

Zheng et al. [38] proposed to learn different similarity representations. The general rating prediction function can be described by Equation (11) in which the non-contextual rating $\hat{r}_{u,i}$ is estimated by the dot product of the user and item vectors in matrix factorization. The function estimates a contextual rating from non-contextual rating by multiplying with the context similarity between a regular situation c and the empty contexts c^E. For example, c could be {at home, at weekend, with parents}, while c^E refers to {N/A, N/A, N/A} which is a special context situation to denote the empty or unknown contexts. It can also be a function from the perspective of transfer learning. Namely, the context similarity is used to transfer user preferences without considering contexts to user preferences in specific context situations.

$$\hat{r}_{u,i,c} = \overrightarrow{p_u} \cdot \overrightarrow{q_i} \cdot sim(c, c^E) \tag{11}$$

Three similarity representations were proposed—independent context similarity (ICS), latent context similarity (LCS) and multi-dimensional context similarity (MCS). In ICS, the model initializes a similarity value (in [0, 1]) for each pair of the context conditions from a same context dimension, e.g., similarities for <at home, at cinema>, <weekend, weekday>, and so forth. The context similarity between c and x situations can be depicted by Equation (12) which is a multiplication of the similarity of conditions in each dimension. Note that, $sim(c_t, x_t)$ were initialized at the beginning, and they can be learned together by using the SGD towards the sum of squared loss.

$$sim(c, x) = \prod_{t=1}^{Z} sim(c_t, x_t) \tag{12}$$

The representation by ICS may be affected by the sparsity issue. For example, we need the similarity between "at home" and "at cinema", but this pair was never learned since it did not appear in the training set. LCS can alleviate this issue by representing each context condition as a latent-factor vector. The dot-product of the vectors can be considered as the similarity between two conditions, while Equation (12) can also be applied to measure the similarity between two context situations.

LCS can alleviate the problems in ICS, but it is possible that the vector representation for a context condition is never trained due to the sparsity. MCS may be able to further alleviate the sparsity problem. To better understand MCS, we visualized the approach by Figure 3. Each context dimension is considered as an axis in the multi-dimensional space. Each condition is initialized with a real value. By this way, each contextual situation is mapped to a point in the space. The dissimilarity between two contextual situations can be captured by the Euclidean distance between the mapped points. The model can learn the real values for each context condition, so that the similarity can be adjusted in the optimization process.

ICS, LCS, and MCS can be considered as general ways to represent context similarity. Theoretically, a contextual rating in situation c can be estimated from rating in another situation x by multiplying with the context similarity between c and x. The prediction function in Equation (11) utilizes the rating in c^E as a source, to reduce the complexity and further alleviate the sparsity issue. The algorithms based on LCS and MCS were optimized on a relaxed loss function, e.g., the dot product is used to represent context similarity in LCS. In this case, the output in Equation (11) is not guaranteed to stay in the original rating scale, but it can be considered as ranking score to sort and rank items. Therefore, the methods based on LCS and MCS are used for top-N recommendations only, while the algorithm based on ICS can be used for both rating predictions and top-N recommendations.

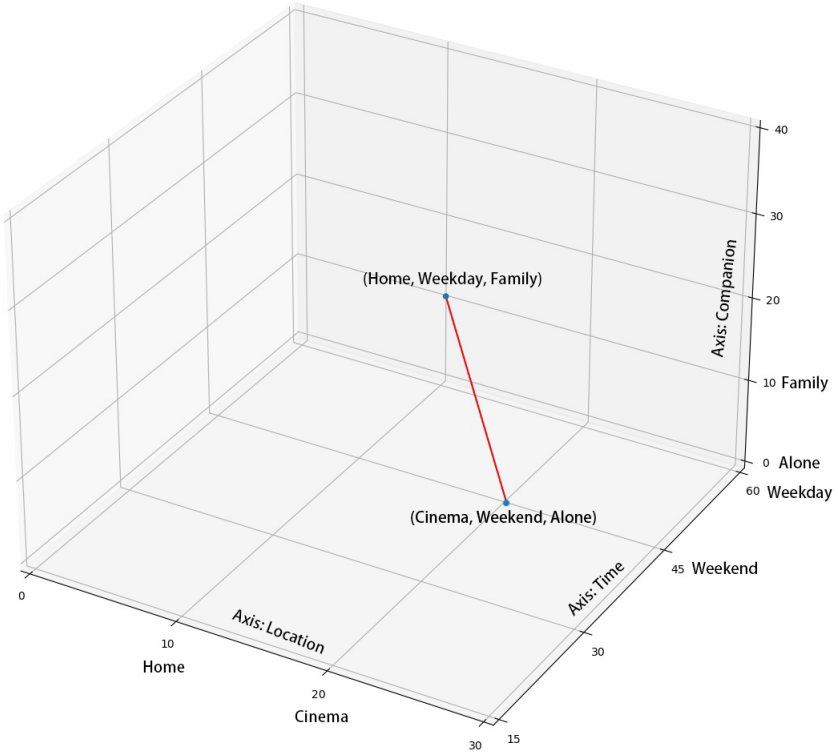

Figure 3. Visualization of multi-dimensional space in MCS.

4.6. Summary: Pros and Cons

Among the five categories of these context similarities, the semantic similarity may only work in limited applications. On the one hand, it requires domain knowledge to build the ontology for the purpose of semantic similarities. On the other hand, the semantic similarities may not be consistent with context similarities. User will probably give similar ratings on the items in two similar context situations, but these contexts may not be necessary to be semantically similar. The matching-based similarities and the inferred similarities from ratings can be generally applied to any CARS, but they may suffer from the sparsity issue, since we may not have enough knowledge to deliver reliable similarity values. The learned similarity representations can be optimized by the recommendation models. They may help to improve the recommendation qualities. However, there are more parameters to be learned, and it is difficult to explain the latent representations.

5. Experiments and Results

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

5.1. Contextual Data Sets

Due to the difficulty in context collection, there is a limited number of contextual data sets available for research. We selected six data sets as shown in Table 4, and most of them can be found from this data repository (https://github.com/irecsys/CARSKit/tree/master/context-aware_data_sets, accessed around 1 August 2021). The data are small, since most of them were collected from user surveys. In real practice, a user may not consume or rate an item for several times in different context situations.

Table 4. List of Contextual Data Sets.

	Food	Restaurant	CoMoDa	Music	STS	Frappe
# of users	212	50	121	42	325	957
# of items	20	40	1232	139	249	4082
# of context dimensions	2	2	8	5	11	3
# of context conditions	8	7	37	21	53	14
# of ratings	6360	2309	2292	3251	2354	87,580
Rating scale	1–5	1–5	1–5	1–5	1–5	0–4.46
Density	9.4%	9.6%	1.4×10^{-7}	3.8×10^{-4}	1.3×10^{-9}	9.4×10^{-5}

- The *Food* data [50] was collected from surveys in which the subjects were asked to give ratings on Japanese 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, and normal. This is a good data set for exploring contextual preferences, since each user gave multiple ratings on a same item in different contexts.
- The *Restaurant* data [11] is also a data set collected from a survey. Subjects gave ratings to the popular restaurants in Tijuana, Mexico, by considering two contextual variables: time and location.
- The *CoMoDa* data [51] is a publicly available context-aware movie data collected from surveys. There are 12 context dimensions that captured users' various situations, including mood, weather, time, location, companion, etc.
- The *South Tyrol Suggests (STS)* data [52] was collected from a mobile app that provides context-aware suggestions for attractions, events, public services, restaurants, and much more for South Tyrol. There are 14 contextual dimensions, such as budget, companion, daytime, mood, season, weather, etc.
- The *Music* data [53] was collected from InCarMusic, which is a mobile application (Android) offering music recommendations to the passengers of a car. Users are requested to enter ratings for some items using a web application. The contextual dimensions include driving style, road type, landscape, sleepiness, traffic conditions, mood, weather, and natural phenomena.
- The *Frappe* data [54] comes from the mobile usage in the app named 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 2 months.

5.2. Evaluation Protocols

We use 5-fold cross validation for all the data sets above and evaluate different context-aware recommendation models based on the rating prediction task and top-10 recommendation task by using the CARSKit library [55], which is a Java-based open-source library for context-aware recommendations.

In the rating predictions, we use mean absolute error (MAE) as the metric, as shown by Equation (13), while T_e refers to the test set.

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

In terms of the top-10 recommendations, we evaluate the relevance and ranking quality by F_1 measure and normalized discounted cumulative gain (NDCG) [56]. The precision is the fraction of recommendations that are relevant, while recall is the fraction of relevant

items that were recommended. F_1 is the metric which combined precision and recall, as shown in Equation (14).

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{14}$$

NDCG a metric for listwise ranking in the well-known learning-to-rank methods. Assuming each user u has a "gain" g_{ui} from being recommended an item i, the average Discounted Cumulative Gain (DCG) for a list of J items is defined in Equation (15).

$$DCG = \frac{1}{N} \sum_{u=1}^{N} \sum_{j=1}^{J} \frac{g_{uij}}{max(1, log_b j))} \tag{15}$$

where the logarithm base is a free parameter, typically between 2 and 10. A logarithm with base 2 is commonly used to ensure all positions are discounted. NDCG is the normalized version of DCG given by Equation (16), where DCG^* is the ideal DCG, i.e., the maximum possible DCG.

$$NDCG = \frac{DCG}{DCG^*} \tag{16}$$

In traditional recommender systems, we may produce a list of recommenders given a user. In CARS, we recommend items given a user and the specific context situation. An example of NDCG calculation can be followed by the workflow described in Algorithm 1. For each unique pair of a user u and a context situation c in the test set, we retrieval the list of the ground truth $\text{List}_{u,c}^{Truth}$ and the list of top-N recommendations $\text{List}_{u,c}^{TopN}$, so that the NDCG for the pair u and c can be obtained. We achieve the NDCG for each user by averaging the NDCG values over all user and context situation pairs, and finally output the NDCG by an average value over all users. It results in small values in these metrics, since a user may not rate several items in a same context situation.

Algorithm 1: Calculation of NDCG in CARS.

```
NDCG = [];
for each unique user in test set do
    u = current user;
    NDCG_u = [];
    for each unique pair of user u and context situation c in test set do
        List_{u,c}^{Truth} = the ranked list of items associated with (u, c) in test set;
        List_{u,c}^{TopN} = the top-N recommendation list associated with (u, c);
        NDCG_{u,c} = NDCG(List_{u,c}^{TopN}, List_{u,c}^{Truth}) by Equation (16);
        NDCG_u.append(NDCG_{u,c});
    end
    NDCG.append(mean(NDCG_u));
end
ndcg_final = mean(NDCG_u)
```

In our experiments, we compared different CACF models that are listed as follows.

- CACF using context similarity.
  - Exact filtering (EF), which is the reduction approach proposed by Adomavicius et al. [7]. We use the contexts for exact filtering and apply MF in the remaining rating profiles to produce recommendations.
  - DCR uses the exact filtering on relaxed contexts, and DCW calculates context similarity based on a weighted matching. We present the results based on the non-dominated simplified DCR and DCW (i.e., noted by ND_s-DCR and ND_s-DCW), which are the latest variants of the DCR and DCW models mentioned in Section 4.3.

  - Chen's method including $Chen_1$ and $Chen_2$ that use the prediction function by Equations (7) and (9), respectively.
  - SPF [10] and CBPF [42], which are two pre-filtering methods that rely on the context similarity based on the distributed vector representation for the context conditions. Note that CBPF runs slowly if there are several items and context conditions. The authors suggested to build the correlations on item clusters to speed up the computation process. We used K-Means clustering to build ten item clusters for the CoModa and Frappe data.
  - Context-aware matrix factorization using ICS, LCS, and MCS [38], which learns different similarity representations.
- Other CACF methods.
  - UISplitting [57], which is a pre-filtering model that combines user splitting and item splitting.
  - Context-aware matrix factorization (CAMF) [13], which learns a bias for each context condition. We use the version that assumes this bias is associated with an item. Namely, the bias for a same context condition may vary from items to items.
  - Tensor factorization (TF) [14], which considers each context variance as an individual dimension in the tensor CANDECOMP/PARAFAC decomposition [58].

Note that we did not add the models using semantic similarity to measure context similarity, since we do not have domain knowledge to build the ontology for each contextual data set.

5.3. Results and Discussions

We present the experimental results in this section. Particularly, we wish to explore the following questions.

- Which one is the winner in terms of the comparison between CACF using context similarity and other CACF approaches?
- Which approach is the best among these CACF using context similarity?
- Among the three categories of CACF using context similarity (i.e., matching-based similarity, inferred similarity, learned similarity), which method is the best in each category?

5.3.1. Performance on Rating Predictions

First, we focus on the performance on the rating prediction task by different CARS models. The results based on MAE are shown in Table 5, where the numbers in bold and italic are the best results by CACF using context similarity, and the underlined values tell the best results by other CACF methods. We further compared the best model from these two categories (i.e., using context similarity or not), and use * to indicate significance at 95% confidence level.

According to Table 5, we can observe that CACF using context similarity may produce predicted ratings with lower MAE in comparison with other CACF approaches. More specifically, ND_s-DCW delivers significant lower MAE on three data sets (i.e., food, restaurant and CoMoDa) than the ones by the UISplitting which is the best performing CACF using other methods rather than context similarity. UISplitting can produce lower MAE on other three data sets, but there are no significant differences with the CACF model using context similarity (e.g., ICS on the music data, SPF on the STS data, and ND_s-DCW on the Frappe data).

Among the models using matching-based context similarity (i.e., EF, ND_s-DCR, ND_s-DCW), ND_s-DCW is clear winner. In terms of the approaches using inferred similarities (i.e., $Chen_1$, $Chen_2$, SPF and CBPF), SPF is the overall winner, while $Chen_1$ method can beat SPF on the music data and obtain comparable results on the CoMoDa data. Regarding the

learned similarity representations, only ICS is applied for rating predictions, while LCS and MCS were developed for top-N recommendations only.

Table 5. MAE Results.

	Food	**Restaurant**	**CoMoDa**	**Music**	**STS**	**Frappe**
EF	0.900	1.026	0.833	1.165	0.961	0.409
ND_s-DCR	0.740	0.787	0.726	1.092	0.934	0.386
ND_s-DCW	***0.725*** *	***0.735*** *	***0.726*** *	1.048	0.923	***0.379***
$Chen_1$	1.105	1.010	0.846	0.686	1.020	0.527
$Chen_2$	1.023	1.090	0.857	1.110	0.952	0.563
SPF	0.900	0.808	0.819	0.918	***0.900***	0.382
CBPF	1.068	0.972	0.830	1.110	1.060	0.402
ICS	0.858	0.825	0.777	***0.678***	0.986	0.388
UISplitting	0.805	0.813	0.775	0.657	0.893	0.378
CAMF	0.845	0.860	0.795	0.727	1.019	0.398
TF	0.966	0.945	0.858	0.864	0.916	0.392

5.3.2. Performance on Top-10 Recommendations

The performance based on the top-10 item recommendations can be depicted by Figure 4. In our experiments, we tuned up parameters for the rating prediction and top-10 recommendations, respectively. Therefore, the patterns in the top-10 recommendations may not be consistent with the results in the rating prediction task.

In Figure 4, we use bars to denote the results of F_1 measure with respect to the y-axis on the left. The curve in the figure tells the results in NDCG with respect to the y-axis on the right.

The comparison of the CACF using context similarity with other CACF methods, the CACF using learned context similarities (i.e., ICS, LCS, MCS) can be considered as the overall winner, except the CoMoDa data where UISplitting and CAMF work better, and the Frappe data where ND_s-DCW and $Chen_1$ methods can produce better F_1 scores.

ND_s-DCW is the best performing CACF using matching-based context similarity. The only exception is shown on the food data where EF works better. As mentioned before, the food data is a rating data with dense contextual ratings—each subject was asked for rate selected items in all six contextual situations. It is not surprising that EF can work better than ND_s-DCW in this case. ND_s-DCW generally works better than others in the rating prediction task, but its performance on top-N recommendations is not as good as the ones in the rating predictions. One of the underlying reasons is that we used sum of squared prediction errors as the fitness function in particle swarm intelligence. By switching to using ranking metrics as the fitness function, its performance on top-N recommendations may be improved. However, the computational cost will also be increased significantly.

In contrast, there are no clear patterns in the CACF using inferred context similarities. Chen's methods perform better than SPF and CBPF on the music, STS, and Frappe data, while SPF works better than others slightly on the food and CoMoDa data. CBPF seems to perform well on the data sets with dense ratings, such as the food and restaurant data.

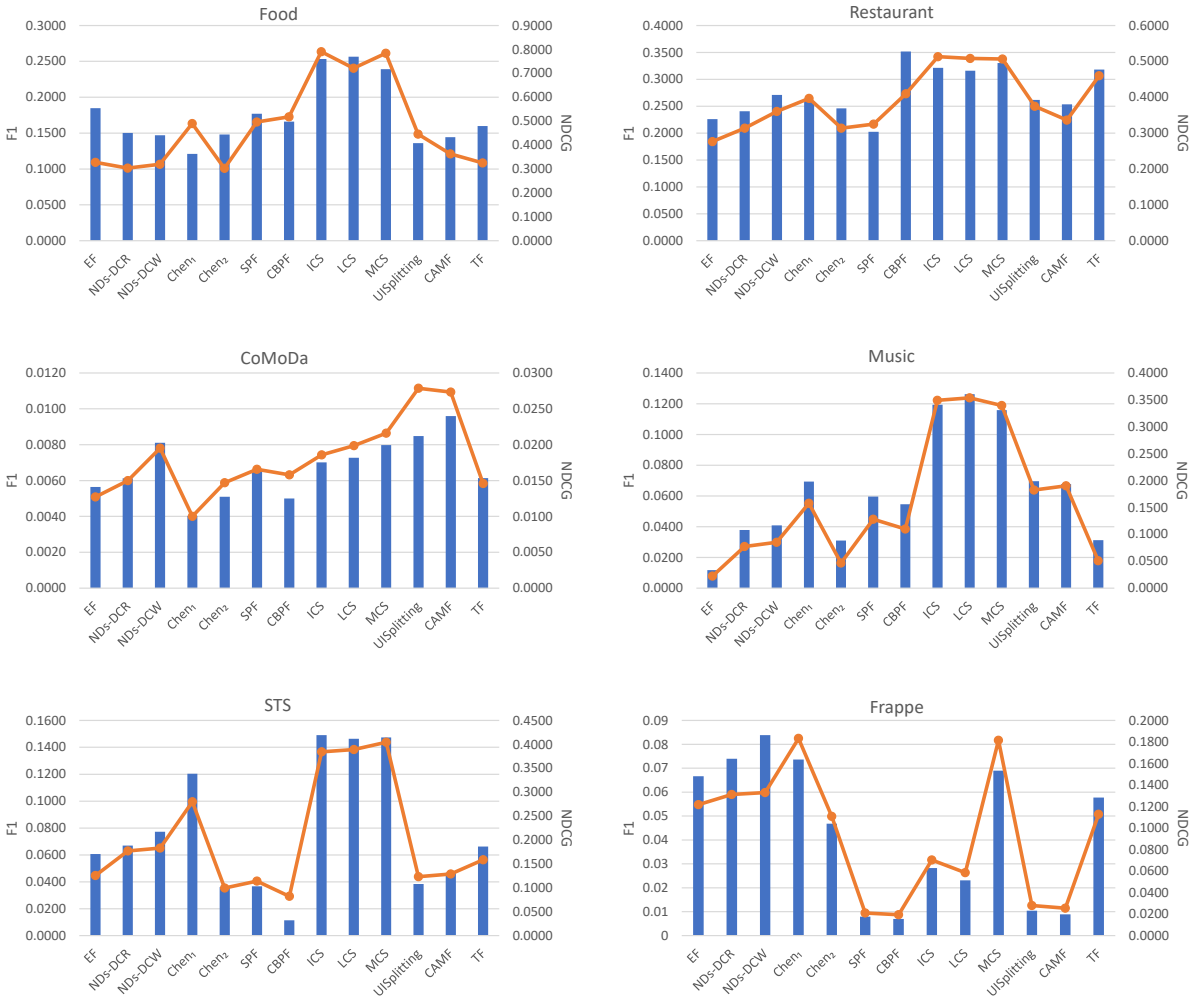

Figure 4. Results of top-10 recommendations.

The CACF models using learned context similarities (i.e., ICS, LCS, and MCS) generally perform well on the top-10 recommendations. The method based on MCS shows significant advantages over ICS and LCS on the CoMoDa and Frappe data. There is no clear winner between ICS and LCS. LCS is expected to further alleviate the sparsity issue, but there are also more parameters to be learned in LCS in comparison with ICS.

Frappe is the only large data in our experiments. The NBCF works better than MF on this data, if we do not consider context information. Therefore, the CARS models built on NBCF (e.g., EF, ND_s-DCR, ND_s-DCW, and Chen's methods) usually outperform the models built upon MF (e.g., SPF, CBPF, ICS, LCS, MCS) on the Frappe data. Surprisingly, the context-aware matrix factorization using MCS presents improved performance in comparison with other CACF models based on MF. It infers the potential advantages of MCS in context-aware recommendations.

6. Conclusions and Future Work

Context-aware recommender systems were built and developed based on the assumption that a user's decisions or preferences on the items may vary from contexts to contexts. However, the recommendation models may suffer from the sparsity issue, while one of the solutions is measuring and utilizing the context similarity in the recommendation approach.

This article delivers a review of existing context-aware collaborative filtering models using context similarity. Particularly, we also provide an empirical study of these models based on six real-world contextual rating data sets. Our experimental results showed that using context similarity can alleviate the sparsity issue and improve the recommendation

models. More specifically, the models based on the matching-based context similarity may perform well in the rating prediction task, while the context-aware collaborative filtering approaches using learned context similarity usually work better in the top-N recommendations.

There are two promising research directions that could be considered in our future work. First, most methods measuring context similarity ignores the dependency among the context dimensions or conditions. Taking the dependency into consideration may be able to produce more accuracy and reliable context similarities. In addition, it is interesting to explore new approaches to incorporate context similarity in deep learning-based recommendation models.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Publicly available datasets were analyzed in this study. This data can be found here: https://github.com/irecsys/CARSKit/tree/master/context-aware_data_sets (accessed on 20 November 2021).

Conflicts of Interest: The author declares no conflict of interest.

References

1. Gross, B.M. *The Managing of Organizations: The Administrative Struggle*; JSTOR: New York, NY, USA, 1964; Volumes I and II.
2. Ruff, J. *Information Overload: Causes, Symptoms and Solutions*; Harvard Graduate School of Education: Cambridge, MA, USA, 2002; pp. 1–13.
3. 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, Chapel Hill, NC, USA, 22–26 October 1994; pp. 175–186.
4. Koren, Y.; Bell, R.; Volinsky, C. Matrix factorization techniques for recommender systems. *IEEE Comput.* **2009**, *42*, 30–37. [CrossRef]
5. Lops, P.; De Gemmis, M.; Semeraro, G. Content-based recommender systems: State of the art and trends. In *Recommender Systems Handbook*; Springer: New York, NY, USA, 2011; pp. 73–105.
6. Burke, R. Hybrid Recommender Systems: Survey and Experiments. *User Model. User-Adapt. Interact.* **2002**, *12*, 331–370. [CrossRef]
7. Adomavicius, G.; Sankaranarayanan, R.; Sen, S.; Tuzhilin, A. Incorporating contextual information in recommender systems using a multidimensional approach. *ACM Trans. Inf. Syst. (TOIS)* **2005**, *23*, 103–145. [CrossRef]
8. Adomavicius, G.; Mobasher, B.; Ricci, F.; Tuzhilin, A. Context-Aware Recommender Systems. *AI Mag.* **2011**, *32*, 67–80.
9. Baltrunas, L.; Ricci, F. Context-based splitting of item ratings in collaborative filtering. In Proceedings of the ACM Conference on Recommender Systems, New York, NY, USA, 23–25 October 2009; pp. 245–248.
10. Codina, V.; Ricci, F.; Ceccaroni, L. Distributional semantic pre-filtering in context-aware recommender systems. *User Model. User-Adapt. Interact.* **2016**, *26*, 1–32. [CrossRef]
11. 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*; Springer: New York, NY, USA, 2014; pp. 695–707.
12. Zheng, Y. Context-Aware Mobile Recommendation by a Novel Post-Filtering Approach. In Proceedings of the FLAIRS Conference, Melbourne, FL, USA, 21–23 May 2018; pp. 482–485.
13. Baltrunas, L.; Ludwig, B.; Ricci, F. Matrix factorization techniques for context aware recommendation. In Proceedings of the Fifth ACM Conference on Recommender Systems, Chicago, IL, USA, 23–27 October 2011; pp. 301–304.
14. 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, Barcelona, Spain, 26–30 September 2010; pp. 79–86.
15. Zheng, Y.; Mobasher, B.; Burke, R. Integrating context similarity with sparse linear recommendation model. In Proceedings of the International Conference on User Modeling, Adaptation, and Personalization, Dublin, Ireland, 29 June–3 July 2015; pp. 370–376.
16. Zheng, Y. Interpreting Contextual Effects by Contextual Modeling In Recommender Systems. In Proceedings of the ACM CIKM, the Workshop on Interpretable Data Mining (IDM)—Bridging the Gap between Shallow and Deep Models, Singapore, 6–10 November 2017.
17. Chen, A. Context-aware collaborative filtering system: Predicting the user's preferences in ubiquitous computing. In Proceedings of the CHI'05 Extended Abstracts on Human Factors in Computing Systems, Portland, OR, USA, 2–7 April 2005; pp. 1110–1111.
18. 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*; Springer: New York, NY, USA, 1999; pp. 304–307.

19. Hidasi, B.; Karatzoglou, A.; Baltrunas, L.; Tikk, D. Session-based recommendations with recurrent neural networks. *arXiv* **2015**, arXiv:1511.06939.
20. Jannach, D.; Mobasher, B.; Berkovsky, S. Research directions in session-based and sequential recommendation. *User Model. User-Adapt. Interact.* **2020**, *30*, 609–616. [CrossRef]
21. El Yebdri, Z.; Benslimane, S.M.; Lahfa, F.; Barhamgi, M.; Benslimane, D. Context-aware recommender system using trust network. *Computing* **2021**, *103*, 1919–1937. [CrossRef]
22. Chen, B.; Xie, H. A Context-Aware Collaborative Filtering Recommender System Based on GCNs. In Proceedings of the 2020 International Conference on Intelligent Transportation, Big Data & Smart City (ICITBS), Vientiane, Laos, 11–12 January 2020; pp. 703–706.
23. Qassimi, S.; Hafidi, M.; Qazdar, A. Towards a folksonomy graph-based context-aware recommender system of annotated books. *J. Big Data* **2021**, *8*, 67. [CrossRef]
24. Resnick, P.; Varian, H.R. Recommender systems. *Commun. ACM* **1997**, *40*, 56–58. [CrossRef]
25. Burke, R. Knowledge-based recommender systems. *Encycl. Libr. Inf. Syst.* **2000**, *69*, 175–186.
26. Adomavicius, G.; Tuzhilin, A. Context-aware recommender systems. In *Recommender Systems Handbook*; Springer: New York, NY, USA, 2011; pp. 217–253.
27. Sarwar, B.; Karypis, G.; Konstan, J.; Riedl, J. Item-based collaborative filtering recommendation algorithms. In Proceedings of the 10th International Conference on World Wide Web, Hong Kong, China, 1–5 May 2001; pp. 285–295.
28. Ning, X.; Karypis, G. SLIM: Sparse linear methods for top-n recommender systems. In Proceedings of the 2011 IEEE 11th International Conference on Data Mining, Vancouver, BC, Canada, 11 December 2011; pp. 497–506.
29. Wu, D.; Shang, M.; Luo, X.; Wang, Z. An L1-and-L2-Norm-Oriented Latent Factor Model for Recommender Systems. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, 1–14. [CrossRef] [PubMed]
30. Zhang, S.; Yao, L.; Sun, A.; Tay, Y. Deep learning based recommender system: A survey and new perspectives. *ACM Comput. Surv. (CSUR)* **2019**, *52*, 1–38. [CrossRef]
31. Unger, M.; Shapira, B.; Rokach, L.; Bar, A. Inferring contextual preferences using deep auto-encoding. In Proceedings of the 25th Conference on User Modeling, Adaptation and Personalization, Bratislava, Slovakia, 9–12 July 2017; pp. 221–229.
32. Jhamb, Y.; Ebesu, T.; Fang, Y. Attentive contextual denoising autoencoder for recommendation. In Proceedings of the 2018 ACM SIGIR International Conference on Theory of Information Retrieval, Tianjin, China, 14–17 September 2018; pp. 27–34.
33. Unger, M.; Tuzhilin, A.; Livne, A. Context-Aware Recommendations Based on Deep Learning Frameworks. *ACM Trans. Manag. Inf. Syst. (TMIS)* **2020**, *11*, 8. [CrossRef]
34. Wasid, M.; Ali, R. Context Similarity Measurement Based on Genetic Algorithm for Improved Recommendations. In *Applications of Soft Computing for the Web*; Springer: New York, NY, USA, 2017; pp. 11–29.
35. Dixit, V.S.; Jain, P. Proposed similarity measure using Bhattacharyya coefficient for context aware recommender system. *J. Intell. Fuzzy Syst.* **2019**, *36*, 3105–3117. [CrossRef]
36. Huynh, H.X.; Phan, N.Q.; Pham, N.M.; Pham, V.H.; Abdel-Basset, M.; Ismail, M. Context-Similarity Collaborative Filtering Recommendation. *IEEE Access* **2020**, *8*, 33342–33351. [CrossRef]
37. Shi, Y.; Larson, M.; Hanjalic, A. Mining contextual movie similarity with matrix factorization for context-aware recommendation. *ACM Trans. Intell. Syst. Technol. (TIST)* **2013**, *4*, 16. [CrossRef]
38. Zheng, Y.; Mobasher, B.; Burke, R. Similarity-based context-aware recommendation. In Proceedings of the International Conference on Web Information Systems Engineering, Miami, FL, USA, 1–3 November 2015; pp. 431–447.
39. Liu, L.; Lecue, F.; Mehandjiev, N.; Xu, L. Using context similarity for service recommendation. In Proceedings of the 2010 IEEE Fourth International Conference on Semantic Computing, Pittsburgh, PA, USA, 22–24 September 2010; pp. 277–284.
40. Zheng, Y.; Burke, R.; Mobasher, B. Optimal Feature Selection for Context-Aware Recommendation Using Differential Relaxation. Available online: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.416.4093&rep=rep1&type=pdf (accessed on 20 November 2021).
41. Zheng, Y.; Burke, R.; Mobasher, B. Recommendation with differential context weighting. In Proceedings of the International Conference on User Modeling, Adaptation, and Personalization, Rome, Italy, 10–14 June 2013; pp. 152–164.
42. Ferdousi, Z.V.; Colazzo, D.; Negre, E. Correlation-based pre-filtering for context-aware recommendation. In Proceedings of the 2018 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops), Athens, Greece, 19–23 March 2018; pp. 89–94.
43. Kolahkaj, M.; Harounabadi, A.; Nikravanshalmani, A.; Chinipardaz, R. A hybrid context-aware approach for e-tourism package recommendation based on asymmetric similarity measurement and sequential pattern mining. *Electron. Commer. Res. Appl.* **2020**, *42*, 100978. [CrossRef]
44. Zheng, Y. Non-Dominated Differential Context Modeling for Context-Aware Recommendations. *Appl. Intell.* **2022**, 1–14. [CrossRef]
45. Gupta, A.; Gusain, K. Selection of Similarity Function for Context-Aware Recommendation Systems. In *Computational Intelligence in Data Mining*; Springer: New York, NY, USA, 2017; pp. 803–811.
46. Linda, S.; Minz, S.; Bharadwaj, K. Effective Context-Aware Recommendations Based on Context Weighting Using Genetic Algorithm and Alleviating Data Sparsity. *Appl. Artif. Intell.* **2020**, *34*, 730–753. [CrossRef]

47. Kennedy, J.; Eberhart, R. A discrete binary version of the particle swarm algorithm. In Proceedings of the 1997 IEEE International Conference on Systems, Man, and Cybernetics, Computational Cybernetics and Simulation, Orlando, FL, USA, 12–15 October 1997; Volume 5, pp. 4104–4108.
48. Kennedy, J.; Eberhart, R. Particle swarm optimization. In Proceedings of the ICNN'95—International Conference on Neural Networks, Perth, WA, Australia, 27 November–1 December 1995; Volume 4, pp. 1942–1948.
49. Eshelman, L.J.; Schaffer, J.D. Real-coded genetic algorithms and interval-schemata. In *Foundations of Genetic Algorithms*; Elsevier: Amsterdam, The Netherlands, 1993; Volume 2, pp. 187–202.
50. Ono, C.; Takishima, Y.; Motomura, Y.; Asoh, H. Context-Aware Preference Model Based on a Study of Difference between Real and Supposed Situation Data. In Proceedings of the International Conference on User Modeling, Adaptation, and Personalization, Trento, Italy, 22–26 June 2009; pp. 102–113.
51. Košir, A.; Odic, A.; Kunaver, M.; Tkalcic, M.; Tasic, J.F. Database for contextual personalization. *Elektrotehniski Vestn.* **2011**, *78*, 270–274.
52. 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*; Springer: New York, NY, USA, 2013; pp. 87–100.
53. 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 *E-Commerce and Web Technologies*; Springer: New York, NY, USA, 2011; pp. 89–100.
54. Baltrunas, L.; Church, K.; Karatzoglou, A.; Oliver, N. Frappe: Understanding the Usage and Perception of Mobile App Recommendations In-The-Wild. *arXiv* **2015**, arXiv:1505.03014.
55. 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, Atlantic City, NJ, USA, 14–17 November 2015.
56. Järvelin, K.; Kekäläinen, J. Cumulated gain-based evaluation of IR techniques. *ACM Trans. Inf. Syst. (TOIS)* **2002**, *20*, 422–446. [CrossRef]
57. 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, Gyeongju, Korea, 24–28 March 2014; pp. 274–279.
58. Harshman, R.A. *Foundations of the PARAFAC Procedure: Models and Conditions for an "Explanatory" Multimodal Factor Analysis*; University of California at Los Angeles: Los Angeles, CA, USA, 1970.