Journal of Computer Science

Original Research Paper

A Comparative Analysis of Various Approaches for Incorporating Contextual Information into Recommender Systems

[1]Quang-Hung Le, [1]Son-Lam Vu and [2]Anh-Cuong Le

[1]Department of Information Technology, Quy Nhon University, Vietnam
[2]Natural Language Processing and Knowledge Discovery Laboratory, Faculty of Information Technology, Ton Duc Thang University, Vietnam

Article history
Received: 16-12-2021
Revised: 08-03-2022
Accepted: 15-03-2022

Corresponding Author:
Quang-Hung Le
Department of Information Technology, Quy Nhon University, Vietnam
Email: lequanghung@qnu.edu.vn

Abstract: Recommender systems are being widely applied in many fields, such as e-commerce, e-documents, places and travel, multimedia, news and advertising and transportation. These systems are similar to an information filtering system that helps to identify a set of items that best satisfy the users' demands based on their preference profiles. The integration of contextual information (e.g., location, weather conditions and user mood) into recommender systems to improve their performance has recently received considerable attention in the research literature. Studies in the relevant literature have focused on incorporating contextual information into conventional recommender systems by employing three approaches: Pre-filtering, post-filtering and modeling. In this paper, we conduct a systematic comparison of various approaches and show how to integrate contextual information into recommender systems. Additionally, we provide an in-depth analysis of the most notable studies to date and point out the strengths, weaknesses and application scenarios for each of the approaches. We also empirically evaluate the real-world datasets, analyzing distinct recommendation quality metrics and characteristics of the datasets. An important result is that accuracy-based comparisons show no clear winner among the approaches.

Keywords: Recommender Systems, Context-Aware Recommender System, Context-Awareness, Pre-Filtering, Post-Filtering, Contextual Modeling

Introduction

A recommender system can be defined as a form of an information filtering system that is intended to provide items[1] that could be of interest to the user (Deshpande and Karypis, 2004). Currently, along with the development and variety of products and services, recommender systems are increasingly widely used in areas such as online shopping (e.g., Amazon), e-news (e.g., Yahoo! News Today), music (e.g., Last.fm), travel (e.g., TripAdvisor), movies (e.g., Netflix) and social networks (e.g., Facebook). Three main approaches are commonly used to build recommender systems: Content-based, collaborative filtering and hybrid (combinations of various inputs and/or compositions of different mechanisms). With the content-based approach, information about the items is used to make appropriate recommendations. Some studies on this approach can be mentioned, as in (Pazzani and Billsus, 2007; Ahn *et al*., 2007; Mooney and Roy, 2000). An alternative approach that is often chosen is collaborative filtering, as in (Le and Le, 2021; Nguyen and Huynh, 2017; Koren, 2008). Instead of relying on the description of the item, this approach explores the user's review histories and item/user similarities.

However, the history of user reviews or item descriptions is not sufficient to give the best relevant recommendations. The preferences of a user can change depending on the context in which they experience the item. Contextual information such as the time, location, weather and mood plays an important role and influences user's item experience (Kulkarni and Rodd, 2020). Obviously, without consideration of context factors, the recommender system can make inappropriate recommendations. The role of context has been recognized in improving the performance of a recommender system (Adomavicius *et al*., 2020). Therefore, many practical applications have integrated context into their systems. An example is Source one[2], which

[1]The term "items" is used to refer to entities: products, services, news, advertisements, and so on

[2]www.sourcetone.com

allows selecting songs based on the listener's mood. Hydra (Liu *et al*., 2019) is a recommender system that offers multimodal transportation planning and is adaptive to various situational contexts (e.g., nearby point-of-interest distribution and weather).

Context-Aware Recommender Systems (CARSs) have become a topic that has received much attention from researchers. Many CARS workshops are organized to discuss issues of integrating contextual information in recommender systems, such as the CARS workshop series (2009-2012, 2019) and CARR (context-aware retrieval and recommendation) workshop series (2011-2014). Recently, a new generation of CARS 2.0 was discussed in CARS workshop 2019, in which researchers revolved around topics such as latent context, exploiting the sequential actions of users (sequence-based recommender systems) and using contextual information from different data sources, such as text, images, videos and speech (Adomavicius *et al*., 2020).

Based on the stages of contextual information integrated into the recommendation process, Adomavicius *et al*. (2011) divided CARS into three paradigms: Contextual pre-filtering, contextual post-filtering and contextual modeling. Many algorithms have been proposed following the above approaches. The algorithms of Exact Pre-Filtering and Generalized Pre-Filtering (Adomavicius *et al*., 2005), Distributional-semantics Pre-Filtering (Codina *et al*., 2016), Item Splitting (Baltrunas and Ricci, 2009), User Splitting (Baltrunas and Amatriain, 2009) and UI Splitting (Zheng *et al*., 2013) belong to the contextual pre-filtering approach, which allows removing the contextual dimensions before applying traditional recommender algorithms that do not account for contextual information. In contrast, the contextual post-filtering algorithms of Weight PoF, Filter PoF (Panniello *et al*., 2009) and Adjusting PoF (Ramirez-Garcia and Garcıa-Valdez, 2014) use contextual information to make appropriate recommendations based on results from traditional recommender systems. At the same time, contextual modeling algorithms do not use results from a traditional recommender system and instead, they directly integrate contextual information into the recommendation function. Popular algorithms under this approach are Contextual-neighbors CM (Panniello and Gorgoglione, 2012), Tensor Factorization (Karatzoglou *et al*., 2010), CAMF (Baltrunas *et al*., 2011b) and Contextual SLIM (Zheng *et al*., 2014).

The success of CARSs requires a comparative analysis of various approaches to incorporate contextual information into recommender systems for successive researchers and practitioners to better understand the strengths, weaknesses and application scenarios of these approaches. To the best of our knowledge, there are very few related studies that shape this area and position existing studies and current progress. In 2009-2014, some studies presented comparisons of context-based approaches (Panniello *et al*., 2009; Panniello and Gorgoglione, 2012; Panniello *et al*., 2014). However, the authors focused only on empirical analysis with some algorithms. In contrast to previous studies, we aim to provide a comparative analysis with the most notable studies to date for each of the approaches in both theory and experiment. The main contributions of this work are listed as follows:

1. We conduct a systematic comparison of various approaches and show how to integrate contextual information into recommender systems
2. We provide an in-depth analysis of the most notable studies to date and point out the strengths, weaknesses and application scenarios for each of the approaches
3. We empirically evaluate on real-world datasets, analyzing distinct recommendation quality metrics and characteristics of the datasets

Preliminaries

Formal Definition of Recommender System

Basically, a recommender system works on two main entities, the user and the item. Formally, let us denote by U the set of users and u is a user in U; I is the set of items, i is an item in I and Y is the rating matrix. We denote with r_{ui} the rating of user u for item i and $\hat{r}_{ui}$ presents the predicted rating of user u for item i. A recommender system attempts to estimate a rating function r such that r: $U \times I \rightarrow R$, mapping each $(u,i) \in U \times I$ to rating set R. In other words, with the rating function r, it estimates values for unknown user-item pairs, $\hat{r}_{ui}$ in the Y matrix (Adomavicius *et al*., 2011). Table 1 depicts the notations and key concepts used in our paper.

Table 1: Glossary

Symbol	Description
U,I,C,R	Set of users, set of items, set of contexts and set of rating values, respectively.
Y	The utility matrix or rating matrix.
M,N,K,L	Number of users, items, contextual dimensions and contextual conditions (for all contextual dimensions), respectively.
F_k	The *k-th* contextual factor or the *k-th* contextual dimension (e.g., F_1 = time, F_2 = weather).
d_k	A contextual condition in the *k-th* contextual factor. It is a specific value in a contextual factor (e.g., d_1 = {Weekend, Weekday}, d_2 = {Sunny, Rainy}).
c	Context, contextual information or contextual situation (e.g., values of a set of contextual factors, $c = (d_1,d_2,...,d_K)$).
r_{ui} =	$r(u,i)$ The rating of user u for item i.
$\hat{r}_{ui}$ =	$\hat{r}(u,i)$ The predicted rating of user u for item i.
$ruic$ =	$r(u,i,c)$ The rating of user u for item i in context c.
$\hat{r}_{uic}$ =	$\hat{r}(u,i,c)$ The predicted rating of user u for item i in context c.

Context in Recommender Systems

In the recommender system, many definitions of context have been given for fields such as data mining, e-commerce personalization, databases, information retrieval, ubiquitous and mobile context-aware systems, marketing and management (Adomavicius and Tuzhilin, 2011). A simple definition of context that is widely used by many studies is presented in (Dey, 2001), according to which "context is 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". For example, considering a tourism recommender system, the entities are travelers and places.

According to Dourish (2004), context can be classified into two views: The representational view and the interactional view. In the first view, context describes the circumstances in which a user chooses or experiences an item. It is represented through a known set of properties that do not change over time. At the same time, the interactional view assumes a cyclical relationship between context and activity, where the activity gives rise to context and the context influences activity (Adomavicius *et al*., 2011). Recent research studies such as (Quadrana *et al*., 2018; Jannach *et al*., 2015; Hariri *et al*., 2012; Smirnova and Vasile, 2017; Massimo and Ricci, 2018) have shown that sequence contextual information (interactional view) can improve the accuracy of recommendations since sequences enable modeling of both the long- and short-term preferences of the user (Adomavicius *et al*., 2020). In a more detailed classification, contexts can be divided into six categories based on two aspects: (i) What a recommender system knows about contextual factors (e.g., fully observable, partially observable and unobservable) and (ii) how contextual factors change over time (e.g., static and dynamic) (Adomavicius *et al*., 2011).

To build a CARS, contextual information must be collected in both phases: The design-building phase of the system and when the system makes recommendations. In the first phase, contextual information is required to build predictive models. In the second phase, the system must obtain contextual information that represents the current situation of the user and the item. Contextual information can be obtained in three ways: Explicitly, implicitly, or inferring from historical data (Adomavicius and Tuzhilin, 2011).

Context-Aware Recommendation Problem Statement

The traditional recommender system (also known as a two-dimensional recommender system) uses only two information dimensions about the users and items, including rating data, user profiles and item content features, to make recommendations. CARSs are an extension of traditional recommender systems, giving recommendations to users and accounting for contextual information (e.g., weather, time, user's mood) or latent contexts (Adomavicius *et al*., 2011; Unger *et al*., 2016). On the other hand, user preference data are expanded into a multidimensional dataset, including users, items and contextual information.

Formally, denoting with C a set of contextual information, $c \in C$ is a context that represents the situation when the user experiences the item. The CARS problem with the rating function is defined as $r: U \times I \times C \rightarrow R$, where $\hat{r}_{uic}$ is the predicted value of the rating function r for user u on item i in context c or for mapping each set $(u,i,c) \in U \times I \times C$ into the set of rating values R (Adomavicius *et al*., 2011).

In the following, we use *contextual dimensions* or *contextual factors* to refer to *contextual variables* such as "time" and "weather". For each contextual dimension, there is a set of possible values called *contextual conditions* (e.g., "weekend" and "weekday" are two contextual conditions for the "time" contextual factor, while "sunny" and "rainy" are two contextual conditions in the "weather" contextual factor). The term *context* refers to *contextual information* or the context of a situation, which is defined by the values of a set of contextual factors (e.g., context c = (weekend, sunny)).

Comparative Analysis

The traditional recommendation process consists of three components: Input, recommendation function and output, which can be presented as follows: [Data (input)] → [Recommendation function] → [Predicted ratings/Ranking → Recommendation items (output)]. A recommendation function is built on these data to predict the user preferences for unrated items, thereby ranking the results and producing a list of the most relevant items for the users. Adomavicius and Tuzhilin (2011) classified CARS into three paradigms depending on which component of the recommendation process the contextual information is included in: Contextual pre-filtering, contextual post-filtering and modeling. The approaches are shown in Fig. 1.

Contextual Pre-filtering

The contextual pre-filtering approach affects the input component in the recommendation process. The main idea of this approach is to use contextual information to convert a multidimensional dataset *User* × *Item* × *Context* ($U \times I \times C$) to a two-Dimensional (2D) dataset *User*×*Item* ($U \times I$), which can be used by traditional 2D recommendation functions. The contextual information can be used to extract a subset that matches the given context from the original multidimensional data or to reconstruct the data (Adomavicius and Tuzhilin, 2011). The traditional 2D algorithms will then be applied to this new dataset. We will analyze two approaches of contextual pre-filtering paradigm in the rest of this section.

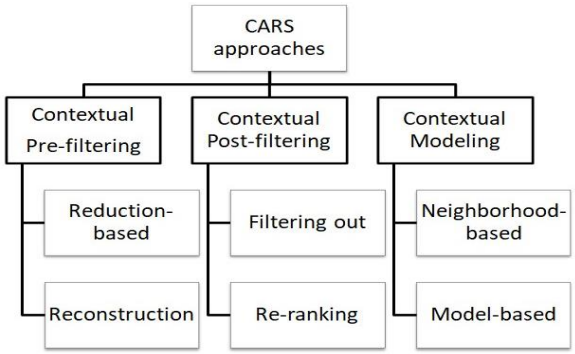

Fig. 1: Context-aware approaches for recommender systems

Reduction-Based Methods

The simplest transformation is Contextual Exact Pre-Filtering (EPF), given by Adomavicius *et al*. (2005). For example, it could be used to recommend restaurants for users when they go with their *girlfriend* on *Saturday*. First, the algorithm filters from the multidimensional dataset *D*, selects only ratings in the context {*Companion = Girlfriend and Time = Saturday*} and then removes the contextual dimensions to obtain a 2D matrix *User×Item*. If we call $f_c(D)$, the function converts from a multidimensional dataset *D* to a 2D dataset with contextual information *c*; then, $f_c(D)$ is essentially a transformation that performs a selection over *D* with the condition {*Companion = Girlfriend and Time = Saturday*}, followed by a projection to obtain *User×Item*:

$$fc(D) = \Pi User,\ Item\left(\sigma Companion = Girl\ friend \wedge Time = Saturday(D)\right) \quad (1)$$

Generally, the $f_c(D)$ function in the EPF method is formally defined as follows:

$$fc(D) = \Pi User,\ Item\left(\sigma Context = c(D)\right) \quad (2)$$

Although the contextual EPF can be performed simply and can select a subset of the dataset that best matches the context, it reduces the amount of data to predict (Adomavicius *et al*., 2005), which leads to several limitations:

- First, it ignores ratings that are collected in contexts that are only slightly different from the given context, creating unnecessary rigidity that can lead to a decrease in the predictive efficiency. For example, ratings of users when going to the restaurant with their girlfriend on *Saturday* or *Sunday* are similar and thus using the condition {*Time = Weekend*} would be more suitable
- Second, there might not be enough data points for the given context. For example, there are very few ratings on *Saturday*, but using a more general context such as {*Time = Weekend*} there will be more data points

The contextual Generalized Pre-Filtering (GPF) method proposed in (Adomavicius *et al*., 2005) can solve the above problems. Accordingly, the filter conditions will be generalized to higher levels to obtain a more suitable two-dimensional dataset. Formally, assume that context *c* is described as consisting of contextual conditions $(d_1,...,d_K)$. The context $c' = (d_1', ..., d_K')$ is a generalization of context $c = (d_1,...,d_K)$ if and only if for every $i = 1 .. K$, we have $d_i \rightarrow d_i^0$ (in other words, d_i and d_i^0 have a relationship *is-a* or *belongs-to*) in the context hierarchy. In this case, ^{c0} can be used instead of *c* as the contextual filter on the input dataset.

Formally, the transformation function $f_c(D)$ can be redefined as follows:

$$fc(D) = \Pi User,\ Item\left(\sigma Context \in S_c(D)\right) \quad (3)$$

where, S_c is a set of generalized contexts of *c*, also known as contextual segmentation. Thus, the GPF method uses the S_c context segmentation instead of the exact context *c* as in the EPF method.

For example, let us recommend to our users which restaurants to go to when they go with their family on Saturday. Thus the context, in this case, is *c* = (*Family*, *Saturday*). In addition, we have the context hierarchy as in Fig. 2.

Then, the generalizations of context *c* could be: c_1 = (*Family*, *Weekend*); c_2 = (*Family*, *Any Time*); and c_3 = (*Notalone*, *Saturday*)... Choosing a correct generalized context to make a filter obtain the best recommendations is an important problem. In particular, in some cases, the number of generalization contexts is very large, when there are many contextual dimensions and there are many values in each dimension.

Another notable point in this approach is that building a prediction model on a reduced dataset is not always better than on a full dataset. The reason is the sparse data problem. Therefore, the approach that combines multiple filters can help to increase the overall efficiency (Adomavicius and Tuzhilin, 2011).

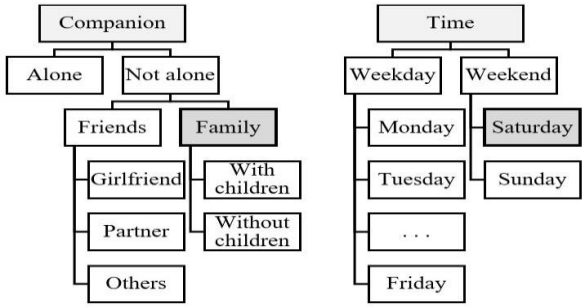

Fig. 2: The context hierarchy

Reconstruction Methods

In this approach, the multidimensional dataset is restructured into two-dimensional data by using given contextual information. Splitting-based methods are often used to refactor the input data to improve the effectiveness of the recommender system while still leveraging traditional recommender system algorithms.

Item Splitting is the first splitting-based algorithm, which was proposed by Baltrunas and Ricci (2009). The main idea is that a rating on an item can be influenced by contextual factors and thus, the authors split the set of ratings for each item into two groups according to the target context (e.g., one group includes ratings in the context *Weekend* and the other includes ratings in the context *Weekday*). This splitting is performed only on the item if it forms two groups with a statistically significant difference. The authors used a number of criteria to evaluate between two groups, such as t_{mean}, t_{prop}, t_{size} and t_{IG}. The method is implemented through two phases:

- Split items into virtual items (performed offline). At the end of this phase, a two-dimensional matrix with M users × (N + p) items is generated, where p is the number of split items.
- Predict the rating of user u for item i in the context of c on the new two-dimensional dataset.

Assume that item i is split into i_1 and i_2 by contextual condition d_k, where i_1 relates the ratings in the context that contains d_k and i_2 is associated with the ratings in the context that do not contain d_k. The rating prediction function for user u on item i in context c, $\hat{r}(u,i,c)$, represented by the traditional 2D recommendation function, $\hat{r}_{2D}(u,i)$, is as follows:

$$\hat{r}(u,i,c)=\begin{cases}\hat{r}_{2D}(u,i_1) & d_k \in c\\ \hat{r}_{2D}(u,i_2) & d_k \notin c\end{cases} \quad (4)$$

For example, we have a subset of the rating dataset of 2 users u_1 and u_2 rating on item i in different contexts, given in Table 2.

Assuming that item i is divided into two virtual items i_1 and i_2 by the contextual condition {*Season* = *Summer*}, we have a new two-dimensional dataset as shown in Table 3. In particular, the ratings on item i in the context {*Season* = *Summer*} are grouped and associated with virtual item i_1. The others are associated with the virtual item i_2.

Table 2: Contextual rating dataset

User	Item	Context	Rating
$u1$	i	{Season = Spring; Companion = Friends}	3
$u1$	i	{Season = Summer; Weather = Sunny}	5
$u2$	i	{Season = Summer; Companion = Friends}	2
$u2$	i	{Season = Autumn; Weather = Rainy}	4

Table 3: The rating dataset with context removed

User	Item	Rating	*Context (removed)*
$u1$	$i2$	3	{Season = Spring; Companion = Friends}
$u1$	$i1$	5	{Season = Summer; Weather = Sunny}
$u2$	$i1$	2	{Season = Summer; Companion = Friends}
$u2$	$i2$	4	{Season = Autumn; Weather = Rainy}

Let us say we need to predict the rating of user u_1 for item i in context c = {*Time* = *Summer*; *Companion* = *Family*}, then: $\hat{r}(u_1,i,c) = \hat{r}_{2D}(u_1,i_1)$.

An approach similar to *Item Splitting* was also proposed by Baltrunas and Amatriain (2009), called User micro profiling or *User Splitting*. Accordingly, each user profile can be split into micro-profiles that represent the user in specific contexts. For example, the set of ratings for user u can be divided into two groups, one that includes ratings in rainy weather context and one that includes ratings in sunny weather context. User rating prediction will be performed on these micro-profiles instead of the original user profile. The resulting two-dimensional matrix will contain (M + p) users × N items, where p is the number of user profiles split.

UI Splitting is an algorithm that combines *Item Splitting* and *User Splitting* (Zheng *et al*., 2013). Accordingly, the items are divided into subitems and the user profiles are divided into micro-profiles according to the context. The resulting two-dimensional matrix will contain (M + p) users ×(N + q) items, where p is the number of user profiles split and q is the number of items split.

Contextual Post-filtering

In contrast to the contextual pre-filtering approach, the contextual post-filtering approach applies the traditional 2D recommendation algorithms on the input dataset regardless of the context factors and then, the results are changed based on given contextual information. For each specific user and context, recommendations (in the *top-N* items task) or predicted ratings are adjusted in two ways: Filtering out any inappropriate results or adjusting the rating value (Adomavicius and Tuzhilin, 2011).

Adjusting

This approach adjusts the predicted rating values and then reranks the list of recommendations. For example, when considering the context of rain, this attraction will have a lower predicted rating than the predicted rating from the traditional 2D model (without considering the context). The first algorithm of contextual post-filtering is *Weight PoF* (Panniello *et al*., 2009). The authors calculated the probability that the user would select the item in a particular context and used this probability to adjust the output from the collaborative filtering system, which is expressed by Eq. 5:

$$\hat{r}(u,i,c)=\hat{r}(u,i)\times P(u,i,c) \quad (5)$$

where, $\hat{r}(u,i,c)$ is the predicted rating of user u for item i in context c. Here, $\hat{r}(u,i)$ is the rating predicted by the traditional 2D model of user u for item i without

considering contextual factors. $P(u,i,c)$ is the probability that user u chooses item i in context c, which is calculated as the number of neighbors of user u who used item i in context c divided by the total number of neighbors of user u. In other words, the larger the number of people similar to user u using item i in context c, the more likely it is that user u prefers item i in context c; otherwise, the system should not recommend this item.

Ramirez-Garcia and Garcıa-Valdez (2014) also proposed a contextual post-filtering algorithm, *PoF-Adjusting*, by adjusting the predicted rating value, which is given by the following equation:

$$\hat{r}(u,i,c) = \beta \times \hat{r}(u,i) + (1-\beta) \times \bar{r}(i,c) \quad (6)$$

where, $r(i,c)$ is the average of all of the ratings for item i in context c. The ratio β with a value in the range 0−1 represents the level of contributions between the two components $\hat{r}(u,i,c)$ and $r(i,c)$.

The solutions in (Panniello et al., 2009) and (Ramirez-Garcia and Garcıa-Valdez, 2014) have a limitation concerning the sparse data problem. For example, it is possible that very few neighbors of user u rate item i in context c. Computation based on the exact context c can lead to a lack of data problems.

Filtering Out

In this approach, we remove inappropriate items from the list of recommendations outputted by a traditional recommender system based on the given context. Probabilistic models can be used to eliminate inappropriate results. The algorithm *Filter PoF* (Panniello *et al*., 2009) is represented by Eq. 7, in which the probability is used to represents the likelihood that a user will select an item in a particular context based on their neighbors:

$$\hat{r}(u,i,c) = \begin{cases} \hat{r}(u,i) & P(u,i,c) \geq p \\ 0 & P(u,i,c) < p \end{cases} \quad (7)$$

Contextual Modeling

With this paradigm, contextual information can be directly integrated into the prediction function. This paradigm can be classified into two approaches: Heuristic-based collaborative filtering and model-based collaborative filtering.

Heuristic-Based Collaborative Filtering

This approach is similar to the heuristic-based collaborative filtering approach in traditional 2D recommender systems. The general idea is to provide a function that predicts the rating of user u for item i in context c based on users similar to u or on items similar to i that were reviewed by user u. As such, the main task in this approach is to find the set of similar users (in user-based collaborative filtering) or the set of similar items (in item-based collaborative filtering) in a given context.

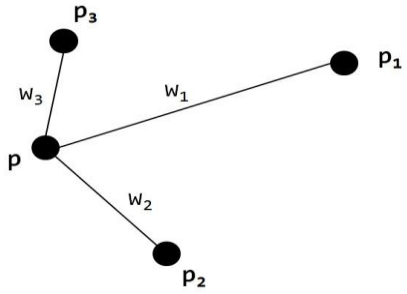

Fig. 3: An example of the influence of neighbors with different weights in a heuristic-based approach

If we consider each user rating as a data point $p=(u,i,c)$ in a multidimensional space $U \times I \times C$, then $r\hat{}(u,i,c) = r\hat{}_p$ is the rating of user u on item i in context c predicted through other $p0$ data points in the space $U \times I \times C$. In particular, the larger the number of neighbors that are similar to point p, the more they will affect the rating predicted at point p.

Figure 3 shows an example of how neighborhoods affect the rating prediction in a heuristic-based approach.

Suppose that p has 3 neighbors, p_1, p_2 and p_3 with impact weights w_1,w_2 and w_3, respectively. In this case, point p_1 is farthest from point p and has the smallest impact weight and in contrast, point p_3 is closest to point p and has the largest weight. This approach can be generalized by the following equation:

$$\hat{r}(u,i,c) = \hat{r}_p = k \times \sum_{p \neq p'} w(p,p') \times r_{p'} \quad (8)$$

where, k is the normalizing factor. r_p' is the rating value at the neighbor p'; and $w(p, p')$ denotes the contribution weight of neighbor point $p0$ for the predicted rating value at point p. This approach must consider two issues: One issue is how the function $w(p, p')$ is defined and the other is how many neighbors to p should be taken for the best accuracy. If taking too few neighbors, there will be not enough data to predict the ratings at point p; otherwise, taking too many neighbors will result in noise values. It also costs time to execute the algorithm.

Typically, the function $w(p, p')$ is defined as the inverse of the distance between points p and $p0$ in a multidimensional space. Different distance measures can be used to calculate the distance between two points such as the Euclidean distance metric or Manhattan distance metric (Adomavicius and Tuzhilin, 2011). In addition, the function $w(p, p')$ can also be expressed through a function that measures the similarity between points p and $p0$. For example, Panniello and Gorgoglione (2012) use cosine measures to calculate the similarity between two user profiles in different contexts, thereby finding the N closest neighbors for user u in a given context c.

Model-Based Collaborative Filtering

Matrix factorization techniques are widely used in traditional recommender systems. The method characterizes both items and users by vectors of factors inferred from item

rating patterns (Koren *et al*., 2009). In CARSs, Tensor Factorization method proposed by Karatzoglou *et al*. (2010), extended the two-dimensional matrix factorization method into multidimensional matrix factorization. Multidimensional matrices are decomposed into smaller-dimensional matrices, where the user, item and each contextual condition are represented as a feature vector.

Methods using regression models such as tensor factorization must learn many model parameters from the training data. Specifically, the number of parameters increases exponentially with the number of contextual dimensions and it is necessary to use large-sized training data to increase the prediction accuracy. This circumstance means that the computational costs also increase (Karatzoglou *et al*., 2010). Baltrunas *et al*. (2011b) proposed a new method, CAMF (context-aware matrix factorization), which is based on matrix factorization and showed that with small size data and fewer parameters, it is still possible to achieve equal or even better results than tensor factorization. The authors have presented models with different levels of context modeling:

- CAMF-C: Each contextual condition has a global influence on the user ratings. With a given contextual condition, this effect is the same regardless of the item or user. Therefore, if the number of contextual conditions is *L*, then there are *L* model parameters that must be learned from the training data
- CAMF-CC: Each contextual condition has a different effect on each item category. These categories of items can be built based on expert knowledge. Thus, if there are *T* item categories and *L* contextual conditions, the number of model parameters to learn is $L \times T$
- CAMF-CI: Each contextual condition has a different effect on each item. If there are *N* items and *L* contextual conditions, the number of model parameters to learn would be $L \times N$. This model has a more detailed context modeling level than the CAMF-C and CAMF-CC models
- CAMF-CU: Similar to CAMF-CI, each contextual condition has a different effect on each user. Thus, if there are *M* users and *L* contextual conditions, the number of model parameters to learn would be $L \times M$

CAMF models are based on Matrix Factorization (MF). This technique decomposes the rating matrix as follows:

$$Y_{M \times N} = V_{M \times D} \cdot Q^{T}_{N \times D} \tag{9}$$

where *Y* is a two-dimensional rating matrix of size *M* users $\times N$ items. *V* and *Q* are factor matrices of size $M \times D$ and $N \times D$, respectively. The *V* matrix can be seen as the user matrix, where each row in the matrix represents the preference of user *u* over the *D*-dimensional vector, $v\sim_u \in \mathrm{R}^D$. The *Q* matrix can be seen as an item matrix, where each row in the matrix represents item *i* through a *D*dimensional vector, $\sim q_i \in \mathrm{R}^D$. The model that predicts user *u*'s rating for item *i* in context $c = (d_1, d_2, .., d_K)$, which is composed of contextual conditions d_k, is constructed as follows:

$$\hat{r}(u,i,c) = \vec{v}_u \cdot \vec{q}_i + b_i + b_u + \sum_{d_k \in c} B_{d_k} \tag{10}$$

where:

- b_i is the baseline parameter for item *i*, which is calculated as the average of all ratings on item *i* in the entire dataset
- b_u is the baseline parameter of user *u*. For example, a very difficult user *u* will tend to give a low rating and then, the b_u value will be correspondingly low
- B_{dk} are the parameters that model the interaction of the contextual condition $d_k \in c$ on the rating of user *u* for item *i*

The choice of the level of context modeling (e.g., many model parameters or few model parameters) should be considered depending on the data domain and the data size (Baltrunas *et al*., 2011b). This technique is commonly used in recommender systems. However, this method is based on matrix factorization, which represents users and items through vectors with latent factors and thus, it is not favorable for the interpretation of recommended results (e.g., it cannot explain why the system recommends these tourist destinations).

The CAMF method has limitations in providing interpretations of recommendation results. At the same time, collaborative filtering methods based on the exploitation of neighboring information (e.g., Item KNN or User KNN) more easily explain the results. For example, the system recommends to user *u* this tourist destination when traveling with family because users similar to *u* also give high ratings of that attraction when they also go with their family. SLIM is the method proposed by Ning and Karypis (2011), which combines neighborhood-based collaborative filtering and matrix factorization for the *top-N* recommendation task.The SLIM model predicts the ranking score for user *u* on item *i* based on all of the ratings on the other items that have been rated by user *u* and the similarity between these items and item *i*. A square matrix *W* of size $N \times N$ (where *N* is the total number of items in the system) is used to represent the aggregation coefficients between each pair of items. The predictive model is shown through the following equation:

$$\hat{S}(u,i) = Y_{u*} \cdot W_{*i} \tag{11}$$

where:

- $\hat{S}(u,i)$ is the predicted ranking score for user *u* on item *i*
- Y_{u*} is a row vector in the rating matrix *Y*, which represents all ratings of user *u* on different items. The vector has a dimension number of *N*, which corresponds to the number of items in the system
- W_{*i} is a column vector in the coefficient matrix *W* and represents similarities between other items and item

i. The aggregation coefficients are learned from training data through error optimization

Depending on whether the coefficient matrix W represents similarities between items or between users, it can be divided into two models: SLIM-I (based on similarities between items, such as ItemKNN) and SLIM-U (based on similarities between users, such as UserKNN). The coefficients learned by the SLIM models can be used for the interpretation of the recommendation results.

Based on the SLIM method, Zheng *et al*. (2014) integrated contextual information into the model and proposed a Contextual SLIM (CSLIM) approach.

Specifically, $\hat{S}(u,i,c)$ is the prediction of the ranking score of user u on item i in context c by an aggregation of u's ratings on other items in the same contexts c, which is calculated using the following equation:

$$\hat{S}(u,i,c) = Y_{u*c} W *_{*i} \tag{12}$$

In the above formula, Y_{u*c} denotes all of the ratings of user u on the other items in context c. One issue is that users can rarely rate items in the same context. In other words, the vector Y_{u*c} is very sparse (most values are 0). In this case, Zheng *et al*. (2014) proposed a solution based on Contextual Rating Deviations (CRDs). They estimated user u's ratings on item i in context c (i.e., $\hat{r}(u,i,c)$) based on the user's noncontextual rating on this item (i.e., $r(u,i)$, ratings without considering contexts) and the aggregated contextual rating deviations (i.e., the rating deviations in different contextual conditions). There are 6 variants for the CSLIM model: CSLIM-I-CI, CSLIM-I-CU, CSLIM-ICC, CSLIM-U-CI, CSLIM-U-CU and CSLIM-U-CC.

Pros and Cons of the Approaches

Each contextual pre-filtering, post-filtering and modeling paradigm has different strengths and limitations. The contextual pre-filtering and post-filtering paradigms allow the reuse of traditional recommender algorithms, while contextual modeling rebuilds the predictive function with the integration of contextual information. The approaches in each of these paradigms also have different advantages and disadvantages. For example, in the pre-filtering paradigm, the reduction-based approach allows the creation of a reduced dataset that is suitable for the given context, but generating this sub-dataset must be done at the time of execution and leads to a costly amount of time. At the same time, the reconstruction approach allows converting multidimensional data into two-dimensional data before making recommendations, helping to provide quicker recommendations for users. Based on the analysis of the algorithms in each paradigms, we summarize the strengths and limitations of each approach in Table 4.

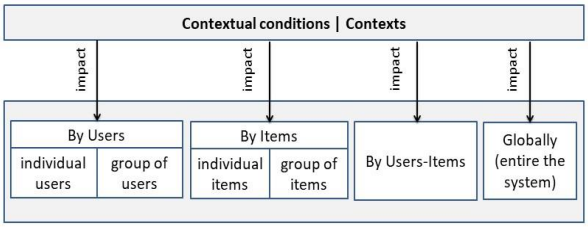

Fig. 4: Aspects of integrating contextual information into recommender systems

Integration Aspects

The classification of the approaches in CARS into contextual pre-filtering, contextual post-filtering and contextual modeling allow the system designer to determine the position to inject contextual information in the recommendation process. However, this classification has not shown how to integrate contextual information into the recommender system. In fact, the algorithms in CARS integrate contextual information into the system by demonstrating the impact of the context on the user ratings on items. Based on incorporating contextual information into the system, we classify the approaches according to the aspects that represent the impact of the context as follows and illustrated in Fig. 4:

- Context integration by users
- Context integration by items
- Context integration by users-items
- Global context integration

In the *context integration by users* approach, each context has a different impact on each user's rating. For example, considering user *A*, he/she often gives very high ratings for items he/she experiences under certain contextual conditions (such as sunny); in contrast, he/she always gives bad reviews to items in case it rains. Therefore, weather (sunny/rainy) is an important contextual factor that greatly influences this user's ratings. In other words, for each pair (c,u), there is a value that represents the degree of influence of context c on the rating of the corresponding user u. Algorithms that follow this approach try to represent the impact of contexts on each user. For example, in the User Splitting algorithm (Baltrunas and Amatriain, 2009), the rating set of each user can be divided into two groups by a contextual condition depending on whether this contextual condition has a large influence on the user's rating. In (Baltrunas *et al*., 2011b; Zheng *et al*., 2014), the algorithms CAMF-CU, CSLIM-I-CU and CSLIM-U-CU build a model with parameters that show the effect of each contextual condition on each user, thereby predicting the user's rating according to the respective context.

Table 4: Comparing the approaches in CARS.

Approaches	Advantages	Disadvantages	Publications
Contextual pre-filtering: Reduction-based	Simple, easy to explain results. Reuse of traditional recommender algorithms. Only those data relevant to the given context are used to predict ratings.	Sparse data, fewer data points to predict ratings. Must extract the dataset according to the given context at runtime. Building context segments (i.e., a set of similar contexts, such as generalized pre-filtering) can reduce the problem of sparse data, but it takes a large amount of time to execute and select the optimal model.	Adomavicius *et al*. (2005); Codina *et al*. (2016)
Contextual pre-filtering: Reconstruction	Simple. Reuse of traditional recommender algorithms. The model can be built offline on the reconstructed dataset regardless of the given context, allowing for faster recommendations.	The data points to predict the rating for each user-item pair can be reduced because of the reconstruction (e.g., in the splitting-based methods, the set of ratings is subdivided into subsets). This approach reconstructs the input dataset based on the interaction of contextual information on the user's ratings. However, representing the impact of the context on the ratings is not simple (e.g., each contextual condition also has an interplay, not simply an independent effect).	Baltrunas and Ricci (2009); Baltrunas and Amatriain (2009); Zheng *et al.* (2013)
Contextual post-filtering: Filtering out	Simple. Reuse of traditional recommender algorithms. Can use the input dataset fully without being filtered out of data points. Suitable for *top-N* items recommendation task.	The impact of the context on the ratings cannot be preserved because contextual information is lost during the transformation from a multidimensional dataset to a two-dimensional dataset. For example, averaging all ratings of the same user on the same item in different contexts as the representative rating of that user on the item. Not suitable for the rating prediction task since it sets low ratings to 0.	Adomavicius and Tuzhilin (2011); Panniello *et al.* (2009)
Contextual post-filtering: Adjusting	Simple. Reuse of traditional recommender algorithms. Can use the input dataset fully without being filtered out of data points. Suitable for the rating prediction task.	- The impact of the context on the ratings cannot be preserved because contextual information is lost during the transformation from a multidimensional dataset to a two-dimensional dataset.	Panniello *et al*. (2009); RamirezGarcia and Garcıa-Valdez (2014)
Contextual modeling: Heuristic-based	Simple, easy to explain results. Can use the input dataset with contextual information fully without any transformation.	- The prediction function must redesigned with the integration contextual dimensions.	be of Panniello and Gorgoglione (2012); Adomavicius and Tuzhili (2011)
Contextual modeling: Model-based	Can use the input dataset with contextual information fully without any transformation. Ability to represent and learn the impact of contextual conditions on user ratings, even in complex cases. Fast computing.	The prediction function must redesigned with the integration contextual dimensions. Difficult to explain the results. The number of parameters to learn be very large. Large size training data is required to learn the parameters.	Koren *et al.* (2009); Karatzoglou *et al*. (2010); Baltrunas *et al*. (2011b); Zheng *et al.* (2014)

In the context integration by items approach, each contextual information has a different impact on the user's rating for different items. For example, considering the movie *A*, users usually give good reviews when going with their girlfriend. As such, the contextual condition (i.e., companion = girlfriend) had a positive impact on this movie. However, this contextual condition could have a negative impact on another movie. In other words, for each pair (*c*,*i*), there is a value that represents the influence of context *c* on the corresponding item *i*. The algorithms Item Splitting (Baltrunas and Ricci, 2009), CAMF-CI (Baltrunas *et al*., 2011b), CSLIM-I-CI and CSLIM-U-CI (Zheng *et al*., 2014), are examples of this approach.

The context integration by users and items approach is the finest grain and there is a parameter that represents the impact of contextual information on each user and item combination. In other words, for each tuple of (*c*,*u*,*i*), there is a value that represents the effect of context *c* on user u's rating on item i. The algorithms UIS plitting (Zheng *et al*., 2013) and CAMF-CUCI (Baltrunas *et al*., 2011b), are examples of this approach.

In global context integration, each contextual information has a global influence on all users' ratings on all items. For example, during summer, most tourists will give high ratings to all tourist destinations on the system. Therefore, the contextual condition, summer, has a positive effect on the experience of almost all users at all tourist destinations. The algorithm CAMF-C (Baltrunas *et al*., 2011b) is implemented at this level of representation since there is only one parameter for each contextual condition on the entire system.

In addition, the context can affect a group of users classified by a certain criterion or a group of items categorized by category. For example, in the CAMF-CC algorithm (Baltrunas *et al*., 2011b), each contextual condition has an impact on each item category. The contextual information that we mention above can be broadly understood on any level of contexts, such as a contextual condition or a group of contextual conditions or context segments. By combining the levels of contextual information and the level of entities affected (i.e., users, items, users-items, global), we can build a variety of prediction models.

Experiments

In this section, we empirically evaluate on real-world datasets, analyzing distinct recommendation quality metrics and characteristics of the datasets.

Datasets

The context-aware datasets used in our experiment includes Tijuana Restaurant and InCarMusic, specifically as follows:

- Tijuana Restaurant: A context-aware dataset built by (Ramirez-Garcia and Garcıa-Valdez, 2014) on the restaurant domain. This dataset includes 50 users surveyed, ratings for 40 restaurants with values from 1 − 5. Each restaurant is rated 6 times corresponding to 6 different contexts. There are 2 contextual dimensions: time (weekday and weekend) and location (school, home, work). A total of 1422 ratings was collected.
- In C arMusic: A context-aware dataset built by (Baltrunas *et al*., 2011a) on the music field. This dataset includes 8 contextual dimensions: Driving style, road type, landscape, sleepiness, traffic conditions, mood, weather and natural phenomena. The users were asked to rate the played tracks on a scale from 1 to 5. There were 139 tracks from 10 music genres rated.

Experimental Setup

We have selected the standard benchmarks for three types of approaches, as follows:

- Contextual pre-filtering: Item Splitting and User Splitting algorithms
- Contextual post-filtering: PoF-Adjusting algorithm
- Contextual modeling: CAMF-CU and CAMF-CI algorithms

The 5-fold cross-validation method with two metrics, MAE and RMSE, is used to compare the efficiency between the algorithms. The MAE and RMSE are common metrics for evaluating rating prediction task (Shani and Gunawardana, 2011), which are based on the error magnitude, which means the difference between the predicted rating and the actual rating. Let T be an evaluation dataset that consists of pairs (u,i,c) with hidden rating values. The RMSE measure is given by the formula:

$$RMSE = \sqrt{\frac{1}{|T|}\sum_{(u,i,c)\in T}\left(\hat{r}_{uic}\right)\left(\hat{r}_{uic} - r_{uic}\right)^2} \quad (13)$$

and MAE is calculated by the formula:

$$MAE = \frac{1}{|T|}\sum_{(u,i,c)\in T}\left|\hat{r}_{uic} - r_{uic}\right| \quad (14)$$

The open-source tool Carskit (Zheng *et al*., 2015) is used to empirically compare the rating prediction accuracy between the three approaches. The parameters for the above algorithms are set as follows: The maximum number of neighbors is 20 and the number of hidden factors in matrix factorization is 10. Pearson's correlation is used to measure the similarity between items or users.

We have implemented a post-filtering algorithm3, PoF-Adjusting, on Carskit. The Carskit framework has an architecture that allows the expansion and integration of new algorithms. Accordingly, the algorithms are divided into two groups: (i) Baseline, including traditional algorithms that address two-dimensional matrices; (ii) CARS, including contextual recommendation algorithms.

The contextual post-filtering algorithm is integrated into Carskit to process and adjust the predicted rating value from the baseline algorithms based on the given contextual information. By integrating into Carskit, we can use all features that the framework provides, such as the data reader, configuration and evaluations.

Results and Discussion

The results that compare the accuracy between the algorithms of the three approaches of contextual pre-filtering, contextual post-filtering and contextual modeling on the two metrics MAE and RMSE are shown in Fig. 5, 6, 7 and 8.

The experimental results show that the accuracy of the algorithms depends on the metric used to evaluate them. Specifically, with MAE (Fig. 5), User Splitting is the algorithm that gives the best results (MAE = 0.62) on the Tijuana Restaurant dataset, while with RMSE (Fig. 6), the best result belongs to the CAMF-CU algorithm (RMSE = 0.96). This finding can be explained by the evaluation of the two metrics being different. Specifically, the RMSE will square the error before averaging. Therefore, for cases with large errors, RMSE penalizes the errors more than MAE. This circumstance means that the RMSE metric is most useful when large errors are especially undesirable.

The results also show that there is no clear winner among the paradigms. For example, algorithms in the contextual modeling paradigm are not always better than algorithms from other paradigms. Specifically, considering the Tijuana Restaurant dataset (Fig. 6), although the CAMF-CI algorithm belongs to the contextual modeling paradigm (with RMSE = 1.09), it gives better results than the contextual pre-filtering algorithm, Item Splitting (with RMSE = 1.13) and the contextual post-filtering algorithm, PoF-Adjusting (with RMSE = 1.13); however, it is not better than the contextual pre-filtering algorithm User Splitting (with RMSE = 1.02). Similarly, considering the InCarMusic dataset (Fig. 8), the contextual pre-filtering algorithm, User Splitting (with RMSE = 0.99), gives higher accuracy than the contextual modeling algorithm, CAMF-CI (RMSE

[3]The source code is available at: https://github.com/qnu-nlp/carskitx

= 1.11), but is not better than the contextual post-filtering algorithm, PoF-Adjusting (RMSE = 0.91) and the contextual modeling algorithm, CAMF-CU (RMSE = 0.98).

The algorithms in CARS integrate contextual information into the recommender system by representing the impact of the context on the user ratings for items. As described in section Integration Methods, this representation of the impact can be in different aspects (e.g., by users, items, both users-items or globally). To further analyze the effect of the context on users or items, to determine in which case the representation will be better, we choose two algorithms that represent the two approaches above, Item Splitting and User Splitting, for analysis.

Experimental results with these two algorithms on the same dataset, Tijuana Restaurant and evaluation on the MAE metric over 5 folds are shown in Table 5.

Accordingly, although the average accuracy of the User Splitting algorithm (MAE = 0.62) is higher than that of the Item Splitting algorithm (MAE = 0.71), User Splitting is not always better than Item Splitting at all folds. Specifically, User Splitting gave lower accuracy on folds 4 and 5 with MAEs of 0.72 and 0.74, respectively (compared to Item Splitting's MAE on folds 4 and 5, which are 0.70 and 0.69, respectively).

In the 5 folds above, fold 2 has the largest difference in accuracy between the two algorithms. Therefore, we continue to perform in-depth error analysis on this fold. The evaluation dataset on fold 2 includes 44 users, 40 items and 283 ratings. In these, there are 21 cases where the two algorithms have the same prediction, which accounts for 7%; 107 cases of Item Splitting for higher prediction accuracy, which accounts for 38% and 155 cases of User Splitting for higher prediction accuracy, which accounts for 55%. The two algorithms represent the impact of the context on the ratings in different ways, leading to different results. Specifically, under the influence of contextual conditions, users are split into micro-profiles (in the User Splitting algorithm) and items are split into virtual items (in the Item Splitting algorithm). Information about this splitting is shown in Table 6 and 7. For example, in Table 6, the ratings of user 28 are divided into 2 groups, a group that consists of this user's rating under the contextual conditional *time:* 2 (where 2 is an encoded contextual condition in the contextual factor *time*) and a group of remaining ratings. The t-value (13.88) and p-value (0.0002) show this splitting, which results in statistically significant different groups.

Table 8 shows the top-20 cases with the largest difference in prediction results between the two algorithms, sorted in descending order. Based on this result, it can be seen that, in most cases, the User Splitting algorithm gives better results by splitting the rating set into two groups by the users. Specifically, there are 16/20 cases where User Splitting gives better results. In these 16 cases, there are 15/16 cases where the user's ratings are split according to the contextual condition, which accounts for 94%. In particular, the cases that involve users 28 and 44 give very high prediction accuracy with small errors. This finding can be easily explained based on Table 6; these users are split by contextual conditions with very small p-values and large t-values. In other words, the contextual condition had a significant impact on the ratings of these two users.

We continue to consider in 5 cases where user 28 has rated, in which 4 cases User Splitting gives better results, namely, cases 1,2,3,11 and 16. Especially in the case of 11, the split ratings by the user gave bad results (i.e., the actual rating is 3, but predicting 1). This finding could be due to the bad effect of the splitting method. Although this splitting algorithm helps to increase the rating prediction efficiency, it can lead to a decrease in the number of data points used to predict the rating for the user-item pair, thereby reducing the accuracy of the prediction.

Table 5: The results of splitting-based algorithms on the MAE metric over 5-fold (TijuanaRestaurant dataset).

Algorithm	MAE	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Item Splitting	0.71	0.74	0.83	0.61	0.70	0.69
User Splitting	0.62	0.64	0.55	0.46	0.72	0.74
Difference	0.09	0.11	0.28	0.15	-0.02	-0.05

Table 6: Users split by different contextual conditions in the User Splitting algorithm

#	Split users	Split by condition	t-value	p-value
1	28	Time: 2	13.88	0.0002
2	44	Time: 2	4.89	0
3	10	Location: 2	3.91	0.0004
4	48	Location: 2	2.84	0.0084
5	39	Time: 2	2.76	0.0125
6	50	Time: 2	2.51	0.0169
7	47	Time: 2	2.34	0.0301
8	49	Location: 3	2.10	0.0496

Table 7: Items split by different contextual conditions in the Item splitting algorithm

#	Split items	Split by condition	t-value	p-value
1	pancake house	Time: 2	3.14	0.0349
2	californias	Time: 2	2.66	0.0412
3	lalena	Time: 2	2.62	0.0355
4	elporton	Time: 2	2.54	0.0463
5	elmazateno	Location: 2	2.43	0.0177
6	casaplascencia	Location: 3	2.43	0.0361
7	applebees	Time: 2	2.35	0.0223
8	daruma	Location: 2	2.34	0.0248
9	kentucky fried chicken	Time: 2	2.34	0.0246
10	burger king	Location: 2	2.12	0.0378

Table 8: Top-20 cases with the largest difference in prediction results between user splitting and item splitting algorithms

					User Splitting		Item Splitting			
No	User Id	Item Id	Contexts	Rating	Prediction	Absolute error	Prediction	absolute error	Diff	Winner
1	28	la casa del mole	time:1;location:1	1	1.9	0.9	5.0	4.0	-3.1	User Splitting
2	28	la casa del mole	time:1;location:2	2	1.9	0.1	5.0	3.0	-2.9	User Splitting
3	28	dominos pizza	time:2;location:2	5	4.8	0.2	2.0	3.0	-2.8	User Splitting
4	44	burger king	time:2;location:2	5	4.5	0.5	2.0	3.0	-2.5	User Splitting
5	44	super antojitos	time:2;location:3	5	5.0	0.0	2.7	2.3	-2.3	User Splitting
6	44	los arcos	time:2;location:3	5	5.0	0.0	2.8	2.2	-2.2	User Splitting
7	44	dominos pizza	time:2;location:2	5	5.0	0.0	2.8	2.2	-2.2	User Splitting
8	37	burger king	time:2;location:2	5	3.5	1.5	1.3	3.7	-2.2	User Splitting
9	37	burger king	time:1;location:2	1	3.5	2.5	1.3	0.3	2.2	Item Splitting
10	44	el mazateno	time:2;location:2	2	1.0	1.0	5.0	3.0	-2.0	User Splitting
11	28	mc donals	time:1;location:3	3	1.0	2.0	3.0	0.0	2.0	Item Splitting
12	44	el mazateno	time:1;location:3	5	4.5	0.5	2.5	2.5	-2.0	User Splitting
13	44	tortas wash movile	time:1;location:1	1	2.5	1.5	4.5	3.5	-2.0	User Splitting
14	44	tortas wash movile	time:1;location:2	2	2.5	0.5	4.5	2.5	-2.0	User Splitting
15	44	daruma	time:2;location:3	5	4.5	0.5	2.7	2.3	-1.8	User Splitting
16	28	burger king	time:1;location:2	2	1.7	0.3	4.1	2.1	-1.8	User Splitting
17	1	el mazateno	time:1;location:2	5	3.3	1.7	5.0	0.0	1.7	Item Splitting
18	49	la espadana	time:1;location:3	3	5.0	2.0	3.3	0.3	1.7	Item Splitting
19	44	la casa del mole	time:2;location:1	1	2.0	1.0	3.7	2.7	-1.7	User Splitting
20	37	casa plascencia	time:2;location:3	1	1.0	0.0	2.6	1.6	-1.6	User Splitting

From the above analysis, it can be seen that representing the influence of the context on the ratings by users or items will produce different results. Depending on the data characteristics (e.g., t-values, p-values), we should choose the most appropriate representation for the best results.

Related Work

General paradigms for incorporating context in recommender systems have been proposed in research studies (Adomavicius and Tuzhilin, 2011; Adomavicius *et al*., 2011) including contextual pre-filtering, contextual post-filtering and modeling. The authors also classified context based on its changes over time (e.g., static and dynamic) and the level of system knowledge about the context (e.g., fully observable, partially observable and unobservable). This topic can be seen as one of the fundamental research topics for CARSs.

Villegas *et al.* (2018) classified CARS based on techniques used to integrate contextual information into recommender systems and the stages of context were integrated into the recommendation process and techniques to be used at each stage. In 2019, Raza and Ding (2019) presented a review on the continual development of CARSs by analyzing different types of contexts without being limited to any specific application domain. In addition, a list of datasets and evaluation metrics used in the setting of CARS are also shown in the study by the authors.

In addition to contextual information, recent studies have extended CARS by considering user preferences on various criteria to provide better recommendations (Vu and Le, 2022; Zheng et al., 2019; Dridi et al., 2019). For example, users on TripAdvisor can leave reviews with

contextual information such as: Date of trip, trip type. In addition to the overall rating, they also leave ratings on different criteria such as: Cleanliness, location and customer service.

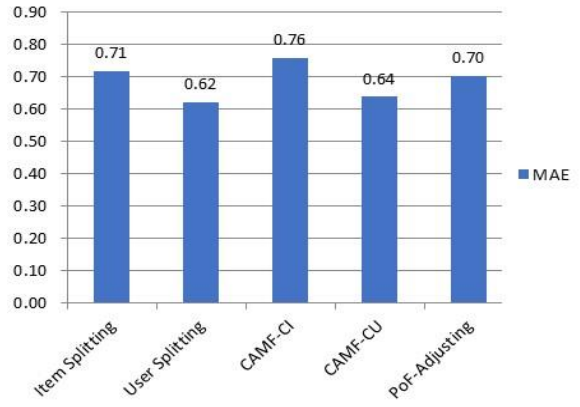

Fig. 5: Comparing the accuracy of algorithms on dataset Tijuana Restaurant (MAE)

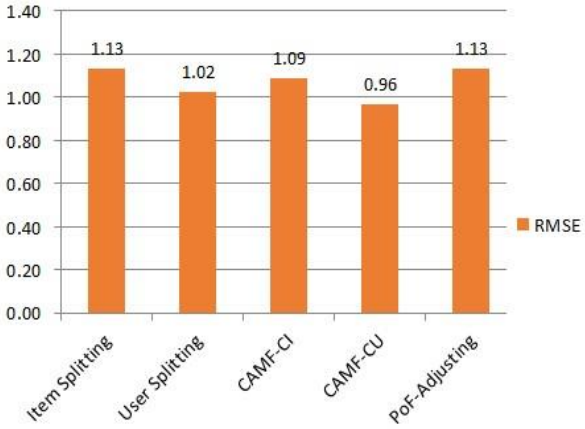

Fig. 6: Comparing the accuracy of algorithms on dataset Tijuana Restaurant (RMSE)

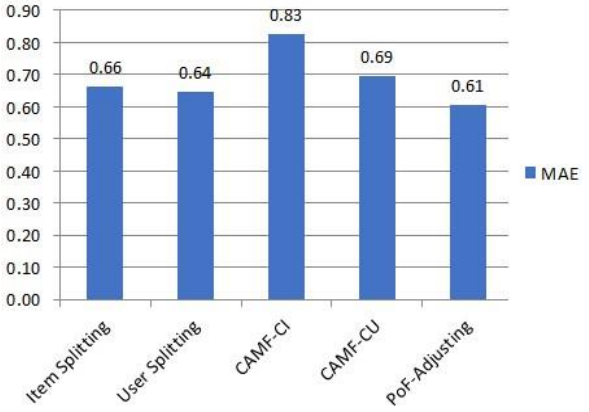

Fig. 7: Comparing the accuracy of algorithms on dataset InCarMusic (MAE)

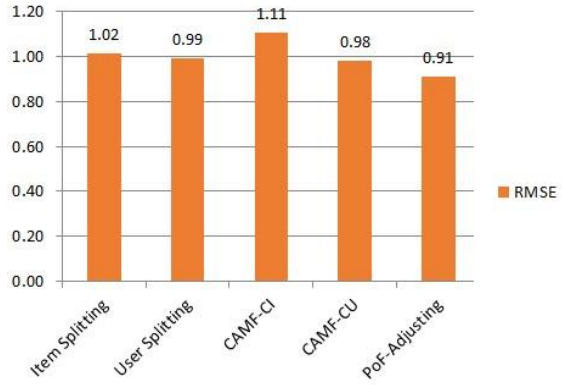

Fig. 8: Comparing the accuracy of algorithms on dataset InCarMusic (RMSE)

CARS has been applied in many different fields, such as places & travel (Renjith *et al*., 2020; Nugroho *et al*., 2019; Jankiewicz *et al*., 2019), news & multimedia (Lommatzsch *et al*., 2017; Liu *et al*., 2013; Hariri *et al*., 2012), ecommerce (Chatzidimitris *et al*., 2020; Twardowski, 2016), transportation (Liu *et al*., 2019), etc. In recent years, CARS has continued to attract the attention of researchers. Abdulkarem *et al*. (2019) presented an overview survey of CARS, frameworks, techniques used and their applications. Abdi *et al*. (2018) examined papers that applied the matrix factorization technique in CARS. This technique allows us to overcome the limitations of neighbor-based collaborative filtering, such as the problem of sparse data and scalability. In 2021, Le *et al*. (2021) surveyed CARS and its applications, focusing on journal articles and conferences published during the period 2016-2020 and identified techniques and applied domains in recent studies. In addition, the authors also presented challenges and future research directions.

Although there are many algorithms proposed in the different approaches, there is a lack of studies that perform a comparative analysis between these approaches and moreover, the comparative studies are mainly based on experimentation. Panniello *et al*. (2009) compared the effectiveness of contextual pre-filtering and post-filtering approaches to determine which approach is better and under what circumstances. The authors compared experimentally two post-filtering algorithms, Weight PoF and Filter PoF, with the EPF pre-filtering algorithm on two datasets in the e-commerce domain. Because no algorithm is always superior, a method of selecting a suitable algorithm for the recommender system based on experiments has been proposed by the authors. Expanding this study, Panniello and Gorgoglione (2012) experimentally compared all three approaches. In this study, the authors added a contextual

modeling algorithm, contextual-neighbors CM, to compare with pre-filtering and post-filtering algorithms (i.e., Weight PoF, Filter PoF and EPF). A comparison between these approaches based on accuracy and diversity was also presented in (Panniello *et al*., 2014).

Conclusion

In this study, we conducted a systematic comparison of various approaches and showed how to integrate contextual information into recommender systems. We provided an in-depth analysis of the most notable studies to date and pointed out the strengths, weaknesses and application scenarios for each of the approaches. Additionally, we implemented a post-filtering algorithm, PoF-Adjusting, for our experiment. We also empirically evaluated on real-world datasets, analyzing the results on different metrics.

All the aforementioned results point out that the recommendation effectiveness is dependent on the approach used and characteristics of the datasets. Thus, to produce optimal results, in addition to choosing the appropriate paradigm of context integration, we also need to pay attention to the representation of the impact of context on entities (e.g., users or items) and data domain. It is worth to emphasize that the contextual pre-filtering and post-filtering approaches allow for efficient reuse of traditional recommendation algorithms that have been widely used and thoroughly proven. Meanwhile, the contextual modeling approach builds algorithms that compute directly on the original contextual data without requiring data transformation.

As a follow-up in the future, we will extend our work by incorporating contextual information into deep learning models. Additionally, we will also combine CARS and multi-criteria recommender system. Furthermore, it would be interesting to explore the applicability of this research in real-world systems on the domains such as places and travel, news and multimedia, e-commerce, transportation, etc.

Funding Information

This work is supported by project B2020-DQN-08 from the Ministry of Education and Training of Vietnam.

Author's Contributions

All authors equally contributed in this study.

Ethics

This article is original and contains unpublished material. The corresponding author confirms that all of the other authors have read and approved the manuscript and no ethical issues involved

References

Abdi, M. H., Okeyo, G., & Mwangi, R. W. (2018). Matrix factorization techniques for context-aware collaborative filtering recommender systems: A survey. doi.org/10.5539/cis.v11n2p1

Abdulkarem, H. F., Abozaid, G. Y., & Soliman, M. I. (2019, February). Context-aware recommender system frameworks, techniques and applications: A survey. In 2019 International Conference on Innovative Trends in Computer Engineering (ITCE) (pp. 180-185). IEEE. https://DIOieeexplore.ieee.org/abstract/document/8646564

Adomavicius, G., & Tuzhilin, A. (2011). Context-aware recommender systems. In Recommender systems handbook (pp. 217-253). Springer, Boston, MA. doi.org/10.1007/978-0-387-85820-3_7

Adomavicius, G., & Tuzhilin, A. (2011). Context-aware recommender systems. In Recommender systems handbook (pp. 217-253). Springer, Boston, MA. doi.org/10.1007/978-0-387-85820-3_7

Adomavicius, G., Bauman, K., Mobasher, B., Ricci, F., Tuzhilin, A., & Unger, M. (2020, September). Workshop on context-aware recommender systems. In Fourteenth ACM Conference on Recommender Systems (pp. 635-637). doi.org/10.1145/3383313.3411533

Adomavicius, G., Sankaranarayanan, R., Sen, S., & Tuzhilin, A. (2005). Incorporating contextual information in recommender systems using a multidimensional approach. ACM Transactions on Information systems (TOIS), 23(1), 103-145. doi.org/10.1145/1055709.1055714

Ahn, J. W., Brusilovsky, P., Grady, J., He, D., & Syn, S. Y. (2007, May). Open user profiles for adaptive news systems: Help or harm?. In Proceedings of the 16th international conference on World Wide Web (pp. 11-20). doi.org/10.1145/1242572.1242575

Baltrunas, L., & Amatriain, X. (2009, October). Towards time-dependant recommendation based on implicit feedback. In Workshop on context-aware recommender systems (CARS'09) (pp. 25-30).

Baltrunas, L., & Ricci, F. (2009, October). Context-based splitting of item ratings in collaborative filtering. In Proceedings of the third ACM conference on Recommender systems (pp. 245-248). doi.org/10.1145/1639714.1639759

Baltrunas, L., Kaminskas, M., Ludwig, B., Moling, O., Ricci, F., Aydin, A., ... & Schwaiger, R. (2011, August). Incarmusic: Context-aware music recommendations in a car. In International conference on electronic commerce and web technologies (pp. 89-100). Springer, Berlin, Heidelberg. doi.org/10.1007/978-3-642-23014-1_8

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

Chatzidimitris, T., Gavalas, D., Kasapakis, V., Konstantopoulos, C., Kypriadis, D., Pantziou, G., & Zaroliagis, C. (2020). A location history-aware recommender system for smart retail environments. Personal and Ubiquitous Computing, 24(5), 683-694. doi.org/10.1007/978-0-387-85820-3_7

Codina, V., Ricci, F., & Ceccaroni, L. (2016). Distributional semantic pre-filtering in context-aware recommender systems. User Modeling and User-Adapted Interaction, 26(1), 1-32. doi.org/10.1007/s11257-015-9158-2

Deshpande, M., & Karypis, G. (2004). Item-based top-n recommendation algorithms. ACM Transactions on Information Systems (TOIS), 22(1), 143-177. doi.org/10.1145/963770.963776

Dey, A. K. (2001). Understanding and using context. Personal and ubiquitous computing, 5(1), 4-7. doi.org/10.1007/s007790170019

Dourish, P. (2004). What we talk about when we talk about context. Personal and ubiquitous computing, 8(1), 19-30. doi.org/10.1007/s00779-003-0253-8

Dridi, R., Tamine, L., & Slimani, Y. (2019, August). Context-aware multi-criteria recommendation based on spectral graph partitioning. In International Conference on Database and Expert Systems Applications (pp. 211-221). Springer, Cham. doi.org/10.1007/978-3-642-00958-7_25

Hariri, N., Mobasher, B., & Burke, R. (2012, September). Context-aware music recommendation based on latenttopic sequential patterns. In Proceedings of the sixth ACM conference on Recommender systems (pp. 131-138). doi.org/10.1145/2365952.2365979

Jankiewicz, P., Kyrashchuk, L., Sienkowski, P., & Wójcik, M. (2019, September). Boosting algorithms for a session-based, context-aware recommender system in an online travel domain. In Proceedings of the Workshop on ACM Recommender Systems Challenge (pp. 1-5). doi.org/10.1145/3359555.3359557

Jannach, D., Lerche, L., & Jugovac, M. (2015, September). Adaptation and evaluation of recommendations for short-term shopping goals. In Proceedings of the 9th ACM Conference on Recommender Systems (pp. 211-218). doi.org/10.1145/2792838.2800176

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

Koren, Y. (2008, August). Factorization meets the neighborhood: A multifaceted collaborative filtering model. In Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 426-434). doi.org/10.1145/1401890.1401944

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

Kulkarni, S. and Rodd, S. F. (2020). Context aware recommendation systems: A review of the state of the art techniques. Computer Science Review, 37, 100255.

Le, Q. H., & Le, T. X. (2021, November). Exploring Set-Inspired Similarity Measures for Collaborative Filtering Recommendation. In 2021 13th International Conference on Knowledge and Systems Engineering (KSE) (pp. 1-6). IEEE. doi.org/10.1109/KSE53942.2021.9648825

Le, Q. H., Vu, S. L., & Le, T. X. (2021). A state-of-the-art survey on context-aware recommender systems and applications. International Journal of Knowledge and Systems Science (IJKSS), 12(3), 1-20. doi.org/10.4018/IJKSS.2021070101

Liu, H., Tong, Y., Zhang, P., Lu, X., Duan, J., & Xiong, H. (2019, July). Hydra: A personalized and context-aware multi-modal transportation recommendation system. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (pp. 2314-2324). doi.org/10.1145/3292500.3330660

Liu, N. N., He, L., & Zhao, M. (2013). Social temporal collaborative ranking for context aware movie recommendation. ACM Transactions on Intelligent Systems and Technology (TIST), 4(1), 1-26. doi.org/10.1145/2414425.2414440

Lommatzsch, A., Kille, B., & Albayrak, S. (2017, August). Incorporating context and trends in news recommender systems. In Proceedings of the international conference on web intelligence (pp. 1062-1068). doi.org/10.1145/3106426.3109433

Massimo, D., & Ricci, F. (2018, September). Harnessing a generalised user behaviour model for next-POI recommendation. In Proceedings of the 12th ACM Conference on Recommender Systems (pp. 402-406). doi.org/10.1145/3240323.3240392

Mooney, R. J., & Roy, L. (2000, June). Content-based book recommending using learning for text categorization. In Proceedings of the fifth ACM conference on Digital libraries (pp. 195-204). doi.org/10.1145/336597.336662

Nguyen, V. D., & Huynh, V. N. (2017). Two-probabilities focused combination in recommender systems. International Journal of Approximate Reasoning, 80, 225-238. doi.org/10.1016/j.ijar.2016.09.005

Ning, X., & Karypis, G. (2011, December). Slim: Sparse linear methods for top-n recommender systems. In 2011 IEEE 11th international conference on data mining (pp. 497-506). IEEE. doi.org/10.1109/ICDM.2011.134

Nugroho, L. E., Saputra, R. Y., Putri, V. M., & Efindo, Y. (2019, December). A context-aware adaptive tourist recommendation system. In Proceedings of the 21st International Conference on Information Integration and Web-based Applications & Services (pp. 453-457). doi.org/10.1145/3366030.3366088

Panniello, U., & Gorgoglione, M. (2012). Incorporating context into recommender systems: An empirical comparison of context-based approaches. Electronic Commerce Research, 12(1), 1-30. doi.org/10.1007/s10660-012-9087-7

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

Panniello, U., Tuzhilin, A., Gorgoglione, M., Palmisano, C., & Pedone, A. (2009, October). Experimental comparison of pre-vs. post-filtering approaches in context-aware recommender systems. In Proceedings of the third ACM conference on Recommender systems (pp. 265-268). doi.org/10.1145/1639714.1639764

Pazzani, M. J., & Billsus, D. (2007). Content-based recommendation systems. In The adaptive web (pp. 325-341). Springer, Berlin, Heidelberg. doi.org/10.1007/978-3-540-72079-9_10

Quadrana, M., Cremonesi, P., & Jannach, D. (2018). Sequence-aware recommender systems. ACM Computing Surveys (CSUR), 51(4), 1-36. doi.org/10.1145/3190616

Ramirez-Garcia, X., & García-Valdez, M. (2014). Post-filtering for a restaurant context-aware recommender system. In Recent advances on hybrid approaches for designing intelligent systems (pp. 695-707). Springer, Cham. doi.org/10.1007/978-3-319-05170-3_49

Raza, S., & Ding, C. (2019). Progress in context-aware recommender systems-An overview. Computer Science Review, 31, 84-97. doi.org/10.1016/j.cosrev.2019.01.001

Renjith, S., Sreekumar, A., & Jathavedan, M. (2020). An extensive study on the evolution of context-aware personalized travel recommender systems. Information Processing & Management, 57(1), 102078. doi.org/10.1016/j.ipm.2019.102078

Shani, G., & Gunawardana, A. (2011). Evaluating recommendation systems. In Recommender systems handbook (pp. 257-297). Springer, Boston, MA. doi.org/10.1145/1124772.1124930

Smirnova, E., & Vasile, F. (2017, August). Contextual sequence modeling for recommendation with recurrent neural networks. In Proceedings of the 2nd workshop on deep learning for recommender systems (pp. 2-9). doi.org/10.1145/3125486.3125488

Twardowski, B. (2016, September). Modelling contextual information in session-aware recommender systems with neural networks. In Proceedings of the 10th ACM Conference on Recommender Systems (pp. 273-276). doi.org/10.1145/2959100.2959162

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

Villegas, N. M., Sánchez, C., Díaz-Cely, J., & Tamura, G. (2018). Characterizing context-aware recommender systems: A systematic literature review. Knowledge-Based Systems, 140, 173-200. doi.org/10.1016/j.knosys.2017.11.003

Vu, S.-L. and Le, Q.-H. (2022). A deep learning based approach for context-aware multi-criteria recommender systems. Computer Systems Science and Engineering.

Zheng, Y., Mobasher, B., & Burke, R. (2014, November). Deviation-based contextual SLIM recommenders. In Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management (pp. 271-280). doi.org/10.1145/2661829.2661987

Zheng, Y., Mobasher, B., & Burke, R. (2015, November). Carskit: A java-based context-aware recommendation engine. In 2015 IEEE International Conference on Data Mining Workshop (ICDMW) (pp. 1668-1671). IEEE. doi.org/10.1109/ICDMW.2015.222

Zheng, Y., Mobasher, B., & Burke, R. D. (2013). The Role of Emotions in Context-aware Recommendation. Decisions@ RecSys, 2013, 21-28.

Zheng, Y., Shekhar, S., Jose, A. A., & Rai, S. K. (2019, April). Integrating context-awareness and multi-criteria decision making in educational learning. In Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing (pp. 2453-2460). doi.org/10.1145/3297280.3297522