Context-Aware Recommender Systems: From Foundations to Recent Developments

Gediminas Adomavicius, Konstantin Bauman, Alexander Tuzhilin, and Moshe Unger

1 Introduction and Motivation

Many existing approaches to recommender systems focus on recommending the most relevant items to individual users and do not take into consideration any contextual information, such as time, place, and the company of other people (e.g., for watching movies or dining out). In other words, traditionally recommender systems deal with applications having only two types of entities, users and items, and do not put them into a context when providing recommendations.

However, in many applications, such as recommending a vacation package, personalized content on a web site, or a movie, it may not be sufficient to consider only users and items—it is also important to incorporate the *contextual information* into the recommendation process in order to recommend items to users under certain *circumstances*. For example, using the temporal context, a travel recommender system would provide a vacation recommendation in the winter that can be very different from the one in the summer. Similarly, in case of personalized content delivery on a website, a user might prefer to read world news when she logs on the website in the morning and the stock market report in the evening, and on weekends to read movie reviews and do shopping, and appropriate recommendations should be provided to her in these different contexts.

G. Adomavicius
Carlson School of Management, University of Minnesota, Minneapolis, MN, USA
e-mail: gedas@umn.edu

K. Bauman (✉)
Fox School of Business, Temple University, Philadelphia, PA, USA
e-mail: kbauman@temple.edu

A. Tuzhilin · M. Unger
Stern School of Business, New York University, New York, NY, USA
e-mail: atuzhili@stern.nyu.edu; munger@stern.nyu.edu

F. Ricci et al. (eds.), *Recommender Systems Handbook*,
https://doi.org/10.1007/978-1-0716-2197-4_6

These observations are consistent with the findings in behavioral research on consumer decision making in marketing that have established that decision making, rather than being invariant, is contingent on the context of decision making. Therefore, accurate prediction of consumer preferences undoubtedly depends upon the degree to which the recommender system has incorporated the relevant contextual information into a recommendation method.

Over the past 15 years, context-aware recommendation capabilities have been developed by academic researchers and applied in a variety of different application settings, including: movie recommenders [4], restaurant recommenders [97, 134], travel recommenders and tourist guides [22, 42, 90, 104, 110, 141], general music recommenders [71, 84, 103, 108], specialized music recommenders (e.g., for places of interest [33, 130], in-car music [21], or music while reading [39]), mobile information search [48], news recommenders [80], shopping assistants [111], mobile advertising [36], mobile portals [119], mobile app recommenders [73, 130], and many others. In particular, *mobile* recommender systems constitute an important special case of context-aware recommenders, where context is often defined by spatial-temporal information such as location and time, and there exists a large body of literature dedicated specifically to mobile recommender systems (e.g., see [69, 124, 130, 132, 134, 137] for a few representative examples). In this chapter we focus on the issues related to the general area of context-aware recommender systems and, therefore, we do not provide a separate in-depth review of mobile recommender systems. Readers interested in a systematic coverage of mobile recommender systems are referred to [79].

Besides academic research on this topic, companies have also extensively incorporated contextual information into their recommendation engines across different applications. For example, in the movie recommendation domain, Netflix has been using contextual information, such as the time of the day or user's location, for a long time with significant business outcomes. As Reed Hastings, the CEO of Netflix, pointed out in 2012 [63], Netflix can improve the performance of its recommendation algorithms up to 3% when taking into account such contextual information. More recently, it has been observed at Netflix that "contextual signals can be as strong as personal preferences; ... make them central to your system and infrastructure" [19]. Following these observations, Netflix expanded the scope of contextual information being used and the range of ways it is utilized by the company [19]. In particular, Netflix considers the following types of contexts in their recommendation methods: location (country and region within the country), the type of device being used for watching videos, time, cultural/religious/national festivities, attention (if the user is focused on watching the video or it is being played in the background), companion with whom the video is being watched, outside events occurring at the same time (sport events, elections, etc.), weather, seasonality, and user's daily patterns (e.g., commuting to work) [19]. This contextual information is being used in several types of recommendation methods at Netflix, ranging from the Context-Aware Tensor Factorization [72] to Contextual RNNs [30].

In addition to movies, context is also used extensively in music recommendations [33, 39, 59, 84, 85]. For example, when choosing which songs to play for a

given user, some streaming platforms infer the mood of the user based on the users' short-term goals and their recent activities. In addition to the inferred mood, Spotify also relies on the following types of contexts in their recommendation algorithms when providing song recommendations, among others: day of the week, time of the day, user's region, type of user's device, and the platform on the device being used for listening [54, 58, 91].

Similarly, LinkedIn uses various types of contextual information, including date, time, location, device/platform and page, to provide career-related recommendations [11]. In fact, context plays a central role in LinkedIn recommendations, the goal for which has been explicitly formulated as "predict probability that a user will respond to an item *in a given context*" [emphasis added] [11].

In this chapter we review the topic of *context-aware recommender systems* (*CARS*). In particular, we discuss the notion of context and how it can be modeled in recommender systems. We also review the major approaches to modeling contextual information in recommender systems and further present recent work on contextual modeling techniques for CARS. Finally, we discuss important directions for possible future research.

2 Context in Recommender Systems

2.1 What Is Context?

Context is a multifaceted concept that has been studied across different research disciplines, including computer science (primarily in artificial intelligence and ubiquitous computing), cognitive science, linguistics, philosophy, psychology, and organizational sciences. In fact, an entire conference—CONTEXT[1]—is dedicated exclusively to studying this topic and incorporating it into various other branches of science, including medicine, law, and business. Since context is a multidisciplinary concept, each discipline tends to take its own idiosyncratic view that is somewhat different from other disciplines and is more specific than the standard generic dictionary definition of context as "interrelated conditions in which something exists or occurs".[2] Therefore, there exist many definitions of context across various disciplines and even within specific subfields of these disciplines. Bazire and Brézillon [29] present and examine 150 different definitions of context from different fields.

To bring some "order" to this diversity of views on what context is, Dourish [53] introduces taxonomy of contexts, according to which contexts can be classified into the representational and the interactional views. In the *representational* view, context is defined with a predefined set of observable factors, the structure (or

[1] https://link.springer.com/conference/context.

[2] https://www.merriam-webster.com/dictionary/context.

schema, using database terminology) of which does not change significantly over time. In other words, the representational view assumes that the contextual factors are identifiable and known a priori and, hence, can be captured and used within the context-aware applications. In contrast, the *interactional* view assumes that the user behavior is induced by an underlying context, but that the context itself is not necessarily observable. Furthermore, Dourish [53] assumes that different types of actions may give rise to and call for different types of relevant contexts, thus assuming a *bidirectional* relationship between activities and underlying contexts: contexts influence activities and different activities giving rise to different contexts.

Turning to recommender systems, one of their goals is estimating user's utility, e.g., expressed as ratings, for the yet-to-be-consumed items, typically by examining prior interactions between users and items. This means that *users* and *items* are fundamental entities considered in recommender systems, and therefore some of their features, i.e., user's personal characteristics and item's content attributes, can have an effect on user's preferences for items. Moreover, aside from the user and item features, a number of other factors that reflect the user's circumstances while consuming the items, may also impact these preferences, such as time, location and weather. These factors, affecting user's preferences besides user and items characteristics, are considered *contextual factors*.

In the remainder of Sect. 2, we further explore the notion of context and focus on what context is in recommender systems. We start with the traditional and popular representational approach to modeling contextual information in Sect. 2.2, explore and describe a broader classification of modeling contextual factors in Sect. 2.3, and discuss the ways to design and obtain contextual factors in Sect. 2.4.

2.2 Modeling Contextual Information in RS: Traditional Approach

Rating-based recommender systems typically start with the specification of the initial set of ratings that is either explicitly provided by the users or is implicitly inferred by the system. Once these initial ratings are specified, a recommender system (RS) tries to estimate the rating function

$$R : User \times Item \rightarrow Rating$$

for the (user, item) pairs that have not been rated yet by the users. *Rating* is typically represented by a totally ordered set, and *User* and *Item* are the domains of users and items respectively. Once the function R is estimated for the whole $User \times Item$ space, a RS can recommend the highest-rated items for each user, possibly also taking into account item novelty, diversity, or other considerations of recommendation quality [115]. We call such canonical systems *two-dimensional* (2D) since they consider only the *User* and *Item* dimensions in the recommendation process.

In context-aware recommender systems, the aforementioned rating estimation process is enhanced by incorporating contextual information (in addition to user and item information) as potential factors that can impact users' ratings for items:

$$R : User \times Item \times Context \rightarrow Rating,$$

where *Context* represents the domain of contextual information used in the recommender system.

It is important to note that *Context* can be modeled in CARS in a variety of different ways, as will be discussed in Sect. 2.3 in more detail. One popular approach—we refer to it as the *traditional (representational) approach to context-aware recommender systems*—assumes that the contextual information, such as time, location, and the company of other people, is explicitly described by a set of pre-defined *contextual factors* (sometimes called contextual dimensions, variables, or attributes), the structure of which does not change over time (i.e., is static). To illustrate this approach, consider the following example.

Example 1 Consider the application of recommending movies to users, where users and movies are described by the following factors:

- Movie: the set of all the movies that can be recommended; it is represented by MovieID, but can have additional item content features available (e.g., genre, actors, director, plot keywords).
- User: the people to whom movies are recommended; it is defined by UserID, but can have additional user characteristics available (e.g., demographic and socioeconomic attributes).

Further, the contextual information consists of the following three contextual factors:

- *Location*: the location from which the user watches the movie that is represented by LocationType ("home", "theater", "airplane", and "other").
- *Time*: the time when the movie can be or has been seen; it is represented by Date. Depending on the relevant granularity for a given application, Date can also be aggregated in different ways, such as DayOfWeek (with values Mon, Tue, Wed, Thu, Fri, Sat, Sun) or TimeOfWeek ("weekday" and "weekend").
- *Companion*: represents a person or a group of persons with whom one can see a movie. It is defined by CompanionType with the following values: "alone", "friends", "girlfriend/boyfriend", "parents", "extended family", "co-workers", and "other".

Hence, the star rating assigned to a movie by a person may depend not only on the movie and the user, but also on where and how the movie has been seen, with whom, and at what time, i.e., StarRating $= R$ (UserID, MovieID, LocationType, Date, CompanionType). For example, the type of movie to recommend to college student Jane Doe can differ significantly depending on whether she is planning to

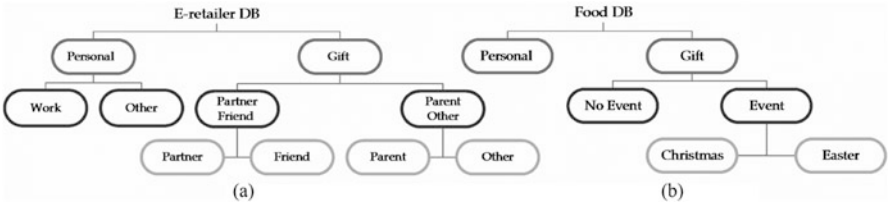

Fig. 1 Contextual information hierarchical structure: (**a**) e-retailer dataset, (**b**) food dataset [99]

see it on a Saturday night with her boyfriend in a movie theater vs. on a weekday with her parents at home.

In addition, contextual factors can have complex domains of possible values. Although this complexity can take many different forms, one popular defining characteristic is the *hierarchical* structure of contextual information that can be represented as trees, as is done in most of the context-aware recommender and profiling systems, including [4] and [99]. E.g., Example 1 already mentioned that the standard Date values (i.e., calendar dates) can be hierarchically aggregated to DayOfWeek and then further to TimeOfWeek. As another example, Fig. 1 presents a multi-level hierarchy for the PurchasingIntent contextual factor, which allows to specify the purpose of a purchasing transaction in an e-retailer application (Fig. 1a) and a groceries application (Fig. 1b) at different levels of granularity.

In [4, 10], the authors proposed to treat the contextual information as part of a *multidimensional data* (*MD*) *model* within the framework of Online Analytical Processing (OLAP) used for modeling multidimensional databases deployed in data warehousing applications [44, 74]. Mathematically, the OLAP model is defined with an n-dimensional *tensor* (see Sect. 4 for subsequent discussion of tensors and their factorization in the context of CARS). In particular, in addition to the classical *User* and *Item* dimensions, additional contextual dimensions, such as $Time$, $Location$, etc., are also included as part of the tensor. Formally, let $D_1, D_2, \ldots, D_n$ be dimensions, two of these dimensions being *User* and *Item*, and the rest being contextual factors. We define the recommendation space for these dimensions as a Cartesian product $S = D_1 \times D_2 \times \ldots D_n$. Moreover, let *Rating* be a rating domain representing the ordered set of all possible rating values. Then the *rating function* is defined over the space S as $R : D_1 \times \ldots \times D_n \rightarrow Rating$.

Visually, ratings $R(d_1, \ldots, d_n)$ on the recommendation space S can be represented by a multidimensional cube, such as the one shown in Fig. 2. For example, the cube in Fig. 2 stores ratings $R(u, i, t)$ for the recommendation space *User* $\times$ *Item* $\times$ *Time*, where the three tables define the sets of users, items, and times associated with *User*, *Item*, and *Time* dimensions respectively. For example, rating $R(101, 7, 1) = 6$ in Fig. 2 means that for the user with User ID 101 and the item with Item ID 7, rating 6 was specified during the weekday.

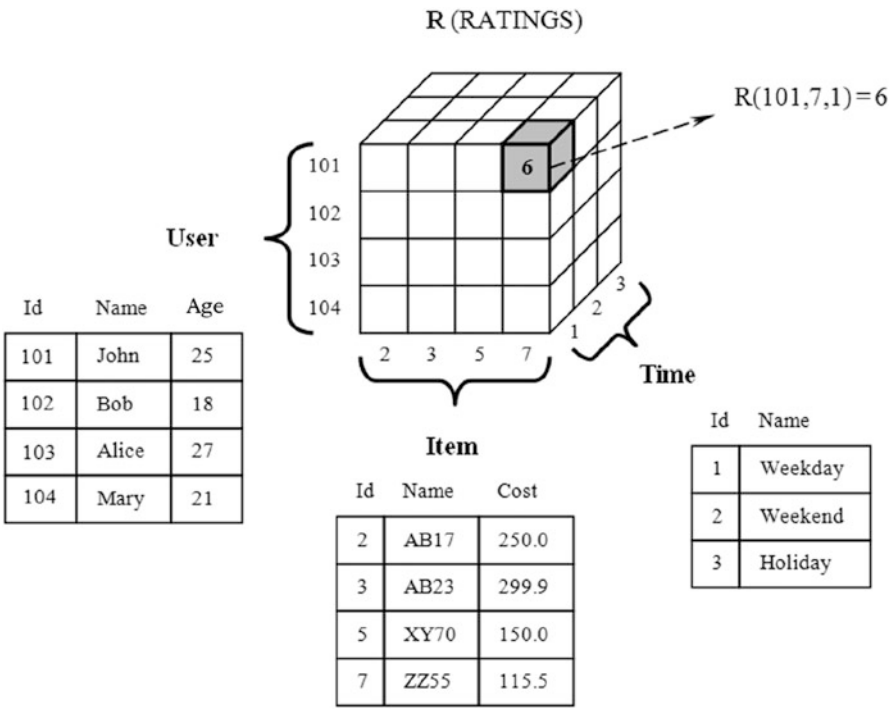

Fig. 2 Multidimensional (MD) model for the *User* × *Item* × *Time* recommendation space

2.3 *Modeling Contextual Information in RS: Major Approaches*

As mentioned earlier, many existing CARS approaches assume the existence of certain contextual factors (sometimes called contextual dimensions, variables, or attributes) that identify the context in which recommendations are provided. As discussed in Sect. 2.2, each contextual factor can be defined by (a) the set of values that the contextual factors take and (b) the structure according to which the values can be meaningfully aggregated, which includes potentially complex hierarchical structures defined using trees or OLAP hierarchies.

A broader classification of major approaches to modeling contextual information, i.e., classification that goes beyond the standard assumption of the explicit availability of predefined contextual factors with stable (static) structure, is based on the following two aspects of contextual factors [9]: (i) what a recommender system may assume (or know) about the structure of contextual factors, and (ii) how the structure of contextual factors changes over time.

The first aspect presumes that a recommender system can have different levels of knowledge about the contextual factors. This may include knowledge of the list

of relevant factors, their structure, and lists of their values. Depending on what is known about the contextual factors in a recommender system, we can classify this knowledge into the following categories:

- *Explicit:* The contextual factors relevant to the application, as well as their structure and lists of their values are known *explicitly*. For example, in a restaurant application, the recommender system may use only DayOfWeek, TimeOfDay, Company, and Occasion contextual factors. For each of these factors, the system may know the relevant structure and the complete list of their values, such as using values Morning, Afternoon, Evening, Night for the TimeOfDay variable.
- *Latent:* No information about contextual factors is explicitly available to the recommender system, and it makes recommendations by utilizing only the *latent* knowledge of context in an implicit manner. For example, the recommender system may build a latent predictive model, such as a hierarchical linear or hidden Markov model, to estimate unknown ratings, where context is modeled using latent variables. Alternatively, we can use deep-learning-based embeddings of users and items combined with latent contextual embeddings to provide context-aware recommendations [52, 126].

The second aspect of contextual factors takes into account whether and how their availability and structure change over time. The settings where the contextual factors are stable over time are classified as static, whereas the factors changing over time are classified as dynamic [9]:

- *Static:* The relevant contextual factors and their structure remain the same (stable) over time. For example, in case of recommending a purchase of a certain product, such as a shirt, we can include the contextual factors of Time, PurchasingPurpose, ShoppingCompanion and only these factors during the current modeling and deployment time-frame of the purchasing recommendation application. Furthermore, we assume that the structure of the PurchasingPurpose factor does not change over time: the set of purposes will remain the same throughout the relevant time-frame of the application. The same is applicable to the ShoppingCompanion factor when the same types of shopping companions remain throughout the application.
- *Dynamic:* This is the case when the contextual factors (e.g., the factors themselves, or their structure, or the list of their possible values) change over time. For example, in the online settings, hashtags of the recent user's posts on social-media platforms, such as Twitter, can be considered as contextual factor for recommendations. At each point of time, the system knows only a finite set of values of this contextual factor, but this list changes dynamically over time.

Combining the two aspects of knowledge about the structure of contextual factors results in four major categories of approaches to modeling contextual factors in recommender systems, as indicated in Fig. 3 (top-right), and we describe them below.

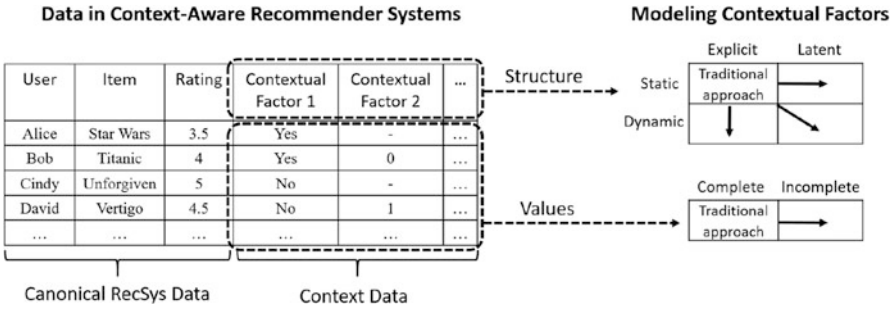

Fig. 3 Modeling contextual information in recommender systems. Top-right part shows four major modeling approaches: traditional (explicit static), latent static, explicit dynamic, and latent dynamic

Traditional (i.e., Explicit Static) Approach As discussed in Sect. 2.2, this approach corresponds to the *representational* view of context [53], which assumes that all the contextual information in a given application can be modeled with a predefined, explicit, finite set of observable factors, where each factor has a well-defined structure and the structure does not change significantly over time (i.e., is static). Vast majority of the first generation of CARS papers has focused on this approach, and it still remains popular because of its simplicity and clarity.

Latent Static Approach This approach represents recommendation settings with stable (i.e., static), yet directly unobservable contextual factors. For example, such contextual factors may include implicit context representation extracted from photos for inferring the companion or the mood of the user, or using compressed (latent) mobile sensor data to represent the user current context. The latent approach is mostly used in order to represent context in an efficient and reduced manner from high-dimensional data, where the relationships between the original contextual features are revealed. Because the contextual factor structure is stable, it can be modeled with latent variables, mostly in the form of a vector containing numeric attributes. These latent variables can be learned using machine-learning methods, such as matrix factorization [77], probabilistic latent semantic analysis (PLSA), deep learning (DL), or hierarchical linear models (HLMs). For example, generative models can be used to extract the implicit contextual representation from the interaction between users, items and explicit contexts. As shown in Fig. 4a, the latent context representation $\overrightarrow{lc}$ can be extracted from the hidden layer of a mixture attentional constrained auto-encoder [52], and represents the weighted combination of multiple implicit contextual representations. Another way to represent unstructured latent contexts is through hierarchical trees, as suggested by Unger and Tuzhilin [125] and shown in Fig. 4b. A hierarchical latent contextual vector is defined as the path (i.e., a set of contextual situations s_i) of the unstructured latent contextual vector from its leaf to the top of the tree. By representing latent contextual information as a set of contextual situations s_i (cluster ids) at different granularity levels extracted from a hierarchical tree [125], we can learn the structure

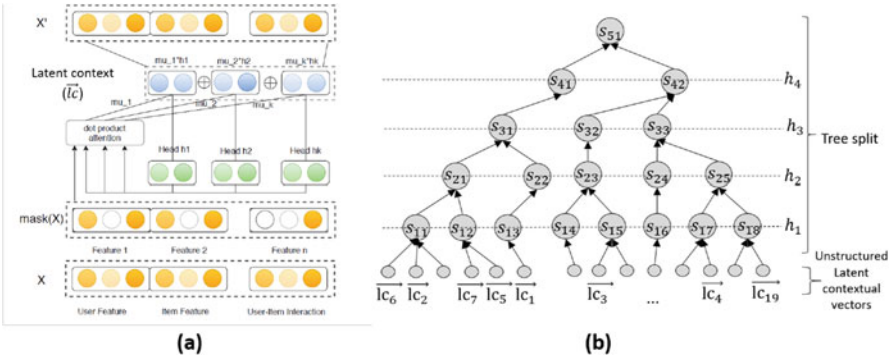

Fig. 4 Examples of latent context representations: (**a**) Implicit context [52] (**b**) Hierarchical latent context [125]

of latent contextual factors and the semantically meaningful relationships among them. Latent approach to contextual modeling constitutes a popular stream in the current generation of CARS research and, therefore, we will describe it at length below.

Explicit Dynamic Approach This approach represents recommendation settings where the explicit structure as well as the list of values of the contextual factor can change over time. For example, in the online settings, the recommender system may consider the website from where a customer arrived at your webpage as an explicit contextual factor PreviousWebsite. Although a customer can arrive to your webpage from numerous different websites, the system may only be aware of a limited set of such websites, each of them serving as a value of the contextual factor PreviousWebsite. Further, this list of previous websites can change dynamically over time, with some new sites being added to the PreviousWebsite variable, while older websites disappearing (and so are the corresponding values of the contextual factor). This makes explicit contextual factor PreviousWebsite dynamic. As yet another example, consider conversational recommender systems that can iteratively collect contextual information from the user. The list of types and values of such contextual information can also change dynamically over time when new contextual factors extracted from recent conversations being constantly added to the list.

One approach to dealing with a dynamically changing contextual information would be to reduce it to the well-studied static case. In particular, the standard static recommendation models can be re-trained once in a while based on the latest dynamically collected information about the context in the domain. Another way to apply static models in dynamic settings is to pre-process the contextual data. For example, values of the PreviousWebsite contextual factor can be generalized to PreviousWebsiteCategory, such as “search,” “social-media,” etc., which has a finite and stable set of values. Although the reduction of dynamic settings to the well-studied static case may have its benefits, the introduced simplifications could lead to

performance limitations. Exploring new models that specifically focus on modeling dynamic context constitutes an important future research direction.

Latent Dynamic Approach This approach represents recommendation settings with latent contextual factors that can change over time. The structure of such latent factors may change because of the dynamic nature of the underlying data. For example, in session-based recommendations, information about the latest websites the user visited (including text, images, etc.) could be used to construct low-level latent contextual factors describing characteristics of the user session. Dynamic changes in the list of websites, such as adding new websites, or modifications of their content may lead to corresponding dynamic changes in the structure of the constructed latent contextual factors. In addition, dynamic information collected from sensors, such as IoT devices as well as the availability of wireless networks can be used to construct latent factors and detect user's context [95].

Similarly to the explicit dynamic case, this approach can also be reduced to the static latent one by regular re-training of the standard latent model using the latest information about the data. For example, Hariri et al. [59] use a topic-modeling approach to map sequences of tags produced by user interactions to a sequence of latent topics of tags representations (tags being considered as different contexts) which capture trends in user's current interests. Although the list of such tags changes dynamically, the model described in [59] is static because it is trained periodically on most frequent tags and has to be retrained in order to capture the latest state of the underlying data. Further, some recent work on CARS has made a step towards the actual latent dynamic approach by implementing various models based on reinforcement learning. For example, [140] deals with the highly dynamic nature of news recommendation and proposes an online deep reinforcement learning recommendation framework that considers the context when a news request happens, including time, weekday, and the freshness of the news (the gap between request time and news publish time). In particular, they represent context as a set of latent factors that kept changing over time based on the reinforcement learning policy.

Complete vs. Incomplete Information Another important aspect affecting the specifics of context modeling approach in RS is the completeness of knowledge about the actual values of contextual factors for each observed user-item interaction. Therefore, based on the completeness of the observed contextual data, approaches to modeling contextual factors can also be classified into the following categories (Fig. 3 (bottom)):

- *Complete:* The values of all the contextual factors used in the application are known *explicitly* at the time when recommendations are made or when the user provides feedback (i.e., a rating) to the system. For example, in the case of recommending a purchase of a certain product, such as a shirt, the recommender system may use contextual factors Time, PurchasingPurpose, and ShoppingCcompanion, and know the actual values of the contextual factors at the

recommendation time (e.g., when this purchase is made, with whom, and for whom).

- *Incomplete:* Only a subset of the contextual factors values are known by the system at the time when recommendations are made or when the user provides feedback (i.e., a rating) to the system. In the previous example, the recommender system may know the actual purpose of the purchase and shopping companion only for some users, while explicitly observing the time factor for all of them. Incompleteness of information may be caused by the way it is being collected, such as through user reviews. For example, [61] applies topic modeling to extract users' trip types from their reviews and [27, 28] discover and extract contextual factors discussed in user reviews for an application. In particular, [27] uses dichotomy between specific and generic reviews to identify context related phrases. [28] studies the ways in which contextual information is expressed in user reviews and relies on the obtained insights to develop a Context Parsing method that extracts all relevant contextual factors along with their values from user reviews. However, not all the reviews contain information about the values of all contextual factors, which makes the observed data incomplete.

The majority of the CARS models working with explicit context are designed with an assumption of complete information about contextual values. However, not all those methods could be directly applied to the incomplete-information case and, thus, recommender system may require additional components. For example, it may pre-process the collected data, impute the missing values of the contextual factors (e.g., using a wide variety of data imputation techniques), and subsequently apply the same recommendation methods designed for the complete-information case. For example, missing contextual data could be imputed using the methods similar to the ones used for imputing missing multi-criteria ratings in [83]. Another way to deal with incomplete information is to modify the recommendation model itself, such as applying various relaxation techniques [141]. Some recent CARS methods [124, 125, 133] that use the latent approach apply multiple feature reduction and transformation techniques in order to automatically handle incomplete information in a lower-dimension representation. For example, as shown in Fig. 4a, encoding-decoding techniques can be used to extract contextual latent information for the recommendation process, where the encoder encodes the high-dimensional data with incomplete information to a latent representation and the decoder takes the lower-dimensional data and tries to reconstruct the original high-dimensional data.

Potential for Future Research As shown in the top-right portion of Fig. 3, there are four major categories of contextual modeling based on the knowledge about the structure of contextual factors, including traditional (i.e., explicit static), latent static, explicit dynamic, and latent dynamic approaches. The first generation of CARS-related research was mostly focused on the traditional approach of modeling contextual information assuming static and explicitly defined structure. The new wave of research papers on this topic explores the potential of large amounts of data collected through various means, such as user logs or various types of sensors, largely following the latent static approach to modeling contextual factors.

As we discuss in Sect. 4, recent work on the use of reinforcement learning in CARS started exploring the latent dynamic approach. More generally, however, the dynamic approaches (both explicit and latent) to modeling contextual factors have been highly under-explored and, thus, represent a strong potential for novel future research. We will discuss these issues further in Sect. 5.

2.4 Designing and Obtaining Contextual Factors

As mentioned earlier, there are several ways to model contextual factors in context-aware recommender systems. The data values for contextual factors can also be obtained in several ways, including:

- *Directly*, i.e., by asking users direct questions or eliciting this information directly from other sources of contextual information. For example, a website may obtain contextual information by asking a user to answer some specific questions before providing a context-aware recommendation. Similarly, a smartphone app may obtain time, location, and motion data from the phone's clock, GPS sensor, and accelerometer, respectively, and weather information can be obtained from a third-party resource by querying it with a specific time and location.
- *By inferring* the context using statistical or data mining methods. In other words, contextual information can be "hidden" in the data in some latent form but can still be *implicitly* used to better estimate the unknown ratings. For example, the household identity of a person flipping the TV channels (husband, wife, son, daughter, etc.) may not be explicitly known to a cable TV company; but it can be modeled as a latent variable using various machine learning methods by observing the TV programs watched and the channels visited. This information can then be used to estimate how much this household would like a particular TV program. As another example of inferring contextual information, consider online reviews, such as provided on Yelp, Amazon, and other popular websites. These reviews contain plenty of contextual information describing specific purchasing or consumption experiences, such a restaurant visits. For example, the user may indicate in a review that she went to the restaurant for dinner with her boyfriend to celebrate his birthday. Some recent work on this topic includes a method for analyzing such online reviews and extracting contextual information from them [27, 28] and a deep learning model that extracts unstructured latent contexts from rich contextual factors such as images and texts for event prediction [96]. Furthermore, [125] proposed a structured representation of latent contextual information that captures contextual situations in a hierarchical manner for context-aware recommendation. Still another approach toward inferring contextual information, albeit for non-RS related problems, was proposed in [75] where temporal contexts were discovered in web-sessions by decomposing these sessions into non-overlapping segments, each segment relating to one specific context. These contexts were subsequently identified using certain optimization and clustering methods.

Regardless of what approach is chosen to model contextual factors in a specific context-aware recommender system, a common challenge for the system designers is that there are many *candidate* contextual factors available for consideration. Thus, the question of which information observable by a recommender system should be considered as relevant, useful context for a given application and used as such in the recommendation process might not be straightforward to answer. In this section, we discuss several broad guidelines that could be helpful to the recommender systems designers in making these decisions.

Contextual Factors Should Not Be Static Properties of Users and Items The first guideline represents an observation that, in order to model a certain data attribute as a contextual factor, it should truly be representative of *context* and not be a static characteristic of users or items. For example, in a restaurant recommendation application, the user's dietary restrictions (e.g., vegetarianism or peanut allergy) would be more appropriate to model as user attributes, while the user's intended company for the meal (e.g., with a significant other vs. with small children vs. alone) would be more appropriate to model as contextual factors. A helpful analogy could be made in terms of Entity-Relationship Diagram design considerations in databases: if there is a clear functional dependency between the data attribute and UserID (or ItemID), then it is more natural to model this data attribute as a user (or item) characteristic and not as a contextual factor. Note, however, that dynamic information about the user or item, such as information that user is on vacation this week, would often be considered and modeled as context.

Contextual Factors Should Be Discoverable Both at System Recommendation Time and at User Feedback Time Interactions between users and recommender systems can be categorized into the following two broad categories based on their timing and directionality:

- *Recommendation* (system-to-user interaction): the system provides recommendations to the users of the items predicted to be most relevant to them;
- *Feedback* (user-to-system interaction): the users provide feedback to the system about their preferences for the consumed items.

In order to provide genuine context-aware recommendations, the recommender system must be able to use the contextual information (i.e., context in which the user intends to consume, which can be elicited from the user, observed or imputed directly by the system) *at the time of recommendation*. As importantly, however, *at the time of feedback* that the user is providing to the recommender system about a consumed item, the corresponding contextual information (i.e., context in which the item was consumed by the user) also needs to be elicited from the user or observed directly by the system. The latter enables the recommender system to get informative contextual training data for improving its predictive models and for making future context-aware recommendations. In other words, for context-aware recommender systems to be useful, system designers must model context and choose contextual factors in a way that encompasses *both* types of aforementioned interactions.

It is important to note that contextual information that is available (observable, discoverable) during *both* of these interactions may only be a subset of what is available at the recommendation time or at the feedback time separately, depending on the specific application domain. In particular, for the *delayed consumption recommender systems*, where the recommendation interactions and the feedback interactions may occur a substantial time apart, the available contextual information can be significantly different. For example, for a restaurant recommendation application, at the recommendation time it may be more useful to ask for the user's current food-related desires and moods (e.g., "I am in the mood for authentic Italian pizza"). However, at the feedback time, typically more specific contextual details that affected the actual restaurant experience can be collected, e.g., from the user's restaurant review (such as "I ended up getting spaghetti and meatballs", "it was too hot—air conditioning was not working properly", and "the waiter was rude"). Modeling the "common" view of contextual information that *combines* the recommendation-time and the feedback-time perspectives represents an important challenge for context-aware recommender systems designers, especially in the delayed consumption applications. In contrast, this challenge is typically less pronounced for *instant consumption recommender systems*, where the recommendation interactions and feedback interactions occur in close temporal proximity, and where the available contextual information can be treated as essentially identical. For example, for a music streaming platform, the recommendation for the next song can be made based on the user's self-reported current mood, and the user's immediate consumption of the song can be viewed as occurring directly in that same context.

Contextual Factors Should Be Broadly Relevant In order to be meaningful for recommendation purposes, a contextual factor should be relevant for a substantial number of users and/or items. This means that the values of a relevant contextual factors should not be constant across different user-item experiences. More importantly, this also means that relevant contextual factors are the ones that affect users' preference judgments (ratings).

Naturally, not all the available contextual factors might be relevant or useful for recommendation purposes. Consider, for example, a book recommender system. Many types of contextual data could potentially be obtained by such a system from book buyers, including: (a) the purpose of buying the book (possible options include for work, for leisure, etc.); (b) planned reading time (weekday, weekend, etc.); (c) planned reading place (at home, at school, on a plane, etc.); (d) the value of the stock market index at the time of the purchase. Clearly some types of contextual information can be more relevant in a given application than some other types. For instance, in this example, the value of a stock market index is likely to be much less relevant as contextual information than the purpose of buying a book.

Because relevance of contextual factors can vary dramatically from application to application (e.g., location as a recommendation context may matter significantly in one recommendation application, but have no impact in another), domain expertise typically plays a major role in identifying a candidate set of contextual factors for a given application. For example, for mobile recommendation

applications, the following four general types of contextual information are often considered [9, 55]: *physical context* (e.g., time, position, activity of the user, weather, light conditions, temperature), *social context* (e.g., is the user alone or in the group, presence and role of other people around the user), *interaction media context* (e.g., device characteristics—phone/tablet/laptop/etc., media content type—text/audio/video/etc.), *modal context* (e.g., user's state of mind—mood, experience, current goals).

In addition to using the *manual* approach, e.g., leveraging domain knowledge of the recommender system's designer or a domain expert, there are *computational* approaches to determining the relevance of a given contextual factor. In particular, a wide variety of existing feature selection procedures from machine learning [76, 86, 120] and statistics [43] can be used to identify relevant contextual factors based on existing ratings data. One example of how to decide which contextual factors should be used in a recommendation application (and which should not) is presented in [4], where the authors use a range of variables initially selected by the domain experts as possible candidates for the contextual factors for the application, build multiple recommendation models by incorporating different subsets of contextual factors, and let the predictive performance of these models point to the most advantageous contextual factors. A different approach [94] suggests that, after collecting the data, one could apply various types of statistical tests identifying which of the chosen contextual factors are truly significant in the sense that they indeed affect movie-watching experiences, as manifested by significant deviations in ratings across different values of a contextual factor. Furthermore, [99] demonstrated relevance of the contextual latent factors, such as intent of purchasing a product, by evaluating the predictive performance improvement of the Bayesian network classifier. A similar approach of using latent variables is presented in [14]. [28] tested the relevance of contextual factors extracted from Yelp reviews by comparing the rating prediction performance of models with and without each factor. Those contextual factors that demonstrated significant performance improvement were claimed to be relevant for an application. [125] used a similar approach to show that usage of hierarchical latent contextual representations leads to significantly better recommendations than the baselines for the datasets having high- and medium-dimensional contexts.

Another approach for assessing relevance of contextual information has been proposed by Baltrunas et al. [23], who developed a survey-based instrument that asks the users to judge what their preferences would be in a wide variety of *hypothetical* (i.e., imagined) contextual situations. This allows to collect richer contextual preference information in a short timeframe, evaluate the impact of each contextual factor on user preferences based on the collected data, and include into the resulting context-aware system only those factors that were shown to be important. Even though the collected data includes only hypothetical contextual preferences (i.e., preferences for items that users imagined consuming under certain contextual circumstances), the authors demonstrate that the resulting context-aware recommender system was perceived to be more effective by users as compared to the non-context-aware recommender.

3 Paradigms for Incorporating Context in RS

Once the relevant context information has been identified and obtained, the next step is to use context intelligently in order to produce better recommendations. In general, approaches to using contextual information in recommender systems can be broadly categorized into two groups: (1) recommendation via *context-driven querying and search*, and (2) recommendation via *contextual preference elicitation and estimation*. In the first approach, systems typically use contextual information (obtained either directly from the user, e.g., by specifying current mood or interest, or from the environment, e.g., obtaining local time, weather, or current location) to query or search a certain repository of resources (e.g., restaurants) and present the resources that best match a given query (e.g., nearby restaurants that are currently open) to the user. This approach has been used by a wide variety of mobile [3, 42] and travel/tourist recommendations, such as GUIDE [46], INTRIGUE [16], COMPASS [127], and MyMap [51] systems. The second general approach to using contextual information in the recommendation process constitutes techniques that model and learn contextual user preferences, e.g., by observing the interactions of this and other users with the systems or by obtaining preference feedback from the user on various previously recommended items. This approach represents a more recent trend in context-aware recommender systems literature [4, 10, 97, 100, 129, 138]. Some applications may also combine the techniques from both general approaches (i.e., both context-driven querying and search as well as contextual preference elicitation and estimation) into a single system, such as mobile tourist guide UbiquiTO [42] and personalized news recommendations system News@hand [41]. While both general approaches offer a number of research challenges, in the remainder of this chapter we will focus on the second, more recent trend of the contextual preference elicitation and estimation in recommender systems.

To start the discussion of the contextual preference elicitation and estimation techniques for the traditional approach to CARS, note that, in its general form, a canonical 2-dimensional (2D) ($User \times Item$) recommender system can be described as a *function*, which takes partial user preference data as its *input* and produces a list of recommendations for each user as an *output*. Accordingly, Fig. 5 presents a general overview of the canonical 2D recommendation process, which includes three components: data (input), 2D recommender system (function), and recommendation list (output). Note that, as indicated in Fig. 5, after the recommendation function is defined (or constructed) based on the available data, recommendation list

Fig. 5 General components of the canonical recommendation process

for any given user u is typically generated by using the recommendation function on user u and all candidate items to obtain a predicted rating for each of the items and then by ranking all items according to their predicted rating value. Later in this section, we will discuss how the use of contextual information in each of those three components gives rise to three different paradigms for context-aware recommender systems.

As mentioned in Sect. 2.2, canonical recommender systems are built based on the knowledge of *partial user preferences* presented in the form $< user, item, rating >$. In contrast, context-aware recommender systems are built based on the knowledge of *partial contextual user preferences* and typically deal with data records of the form $< user, item, context, rating >$, that also includes the contextual information in which the item was consumed by this user (e.g., *Context* = Saturday). In addition, context-aware recommender systems may also make use of the structures of context attributes, such as context hierarchies (e.g., Saturday $\rightarrow$ Weekend) mentioned in Sect. 2.2. Based on the presence of this additional contextual data, several important questions arise: How contextual information should be reflected when modeling user preferences? Can we reuse the wealth of knowledge in canonical (non-contextual) recommender systems to generate context-aware recommendations? We will explore these questions in this section in more detail.

In the presence of available contextual information, following the diagrams in Fig. 6, we start with the data having the form $U \times I \times C \times R$, where C is additional contextual dimension and end up with a list of contextual recommendations $i_1, i_2, i_3 \ldots$ for each user. However, unlike the process in Fig. 5, which does not take into account the contextual information, we can apply the information about

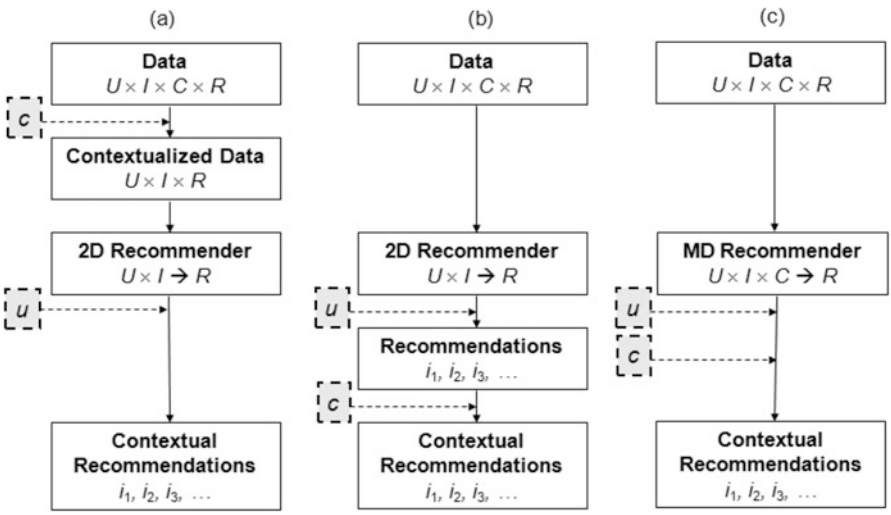

Fig. 6 Paradigms for incorporating context in recommender systems. (**a**) Contextual pre-filtering. (**b**) Contextual post-filtering. (**c**) Contextual modeling

the current (or expected) context c at various stages of the recommendation process. More specifically, the context-aware recommendation process that is based on contextual user preference elicitation and estimation can take one of the three forms, based on which of the three components the context is used in, as shown in Fig. 6:

- *Contextual pre-filtering* (or contextualization of recommendation input). In this recommendation paradigm (presented in Fig. 6a), contextual information drives data selection or data construction for that specific context. In other words, information about the current context c is used for selecting or constructing the relevant set of data records (i.e., ratings). Then, ratings can be predicted using any canonical 2D recommender system on the selected data.
- *Contextual post-filtering* (or contextualization of recommendation output). In this recommendation paradigm (presented in Fig. 6b), contextual information is initially ignored, and the ratings are predicted using any canonical 2D recommender system on the *entire* data. Then, the resulting set of recommendations is adjusted (*contextualized*) for each user using the contextual information.
- *Contextual modeling* (or contextualization of recommendation function). In this recommendation paradigm (presented in Fig. 6c), contextual information is used directly in the modeling technique as part of rating estimation.

Contextual Pre-filtering As shown in Fig. 6a, the contextual pre-filtering approach uses contextual information to select or construct the most relevant 2D ($User \times Item$) data for generating recommendations. In particular, context c essentially serves as a *query* for selecting (filtering) relevant ratings data. One major advantage of this approach is that it allows deployment of any of the numerous 2D recommendation techniques previously proposed in the literature [6]. For example, following the contextual pre-filtering paradigm, Adomavicius et al. [4] proposed a *reduction-based approach*, which reduces the problem of multidimensional (MD) contextual recommendations to the standard 2D $User \times Item$ recommendation space. An example of a contextual data reduction for a movie recommender system would be: if user u wants to see a movie on Saturday, among the set of existing ratings D, *only* the Saturday rating data $D[DayOfWeek = Saturday]$ is used to recommend movies to u. Note that this example represents an *exact pre-filter*. In other words, the data filtering query has been constructed using exactly the specified context.

However, the exact context sometimes can be too narrow. Consider, for example, the context of watching a movie with a girlfriend in a movie theater on Saturday or, i.e., c = (Girlfriend, Theater, Saturday). Using this exact context as a data filtering query may be problematic for several reasons. First, certain aspects of the overly specific context may not be significant. For example, user's movie watching preferences with a girlfriend in a theater on Saturday may be exactly the same as on Sunday, but different from Wednesday's. Therefore, it may be more appropriate to use a more general context specification, i.e., Weekend instead of Saturday. And second, exact context may not have enough data for accurate rating prediction, which is known as the "sparsity" problem in recommender systems literature. In other words, the recommender system may not have enough data points about the

past movie watching preferences of a given user with a girlfriend in a theater on Saturday.

In these cases, recommender system may utilize the hierarchical structure of contextual factors and apply *generalized pre-filtering* proposed by Adomavicius et al. [4]. This approach suggests to generalize the data filtering query obtained based on a specified context. More formally, let's define $c' = (c'_1, \ldots, c'_k)$ to be a generalization of context $c = (c_1, \ldots, c_k)$ if and only if c'_i is a parent of c_i in the corresponding context hierarchy for every $i = 1, \ldots, k$. Then, c' (instead of c) can be used as a data query to obtain contextualized ratings data. Note that there typically exist multiple different possibilities for context generalization, based on the context taxonomy and the desired context granularity. Therefore, choosing the "right" generalized pre-filter becomes an important research problem. For example, Jiang and Tuzhilin [70] examine optimal levels of granularity of customer segments in order to maximize predictive performance of segmentation methods. Applicability of these techniques in the context-aware recommender systems settings constitutes an interesting problem for future research.

So far, we have discussed applying only one pre-filter at a time. However, as it has been well-documented in recommender systems literature, often a combination (a "blend" or an ensemble) of several solutions provides significant performance improvements over the individual approaches [37, 38, 77, 106]. Therefore, it may also be useful to combine several contextual pre-filters into one model at the same time. The rationale for having a number of different pre-filters is based on the fact that, typically there can be multiple different (and potentially relevant) generalizations of the same specific context. Following this idea, Adomavicius et al. [4] use pre-filters based on the number of possible contexts for each rating, and then combine recommendations resulting from each contextual pre-filter. Note that the combination of several pre-filters can be done in multiple ways. For example, for rating estimation in a specific context, (a) one could choose the best-performing pre-filter, or (b) use an aggregate prediction from the entire "ensemble" of pre-filters.

Also note that the contextual pre-filtering approach is related to the problems of building local models in machine learning [13]. Rather than building the global rating estimation model utilizing all the available ratings, this approach builds a *local* rating estimation model that uses only the ratings pertaining to the user-specified criteria in which a recommendation is made (e.g., morning). The advantage of the local model is that it uses *more relevant* data, but at the same time it relies on *fewer* data points. Which of these two trends dominates on a particular segment may depend on the application domain and on the specifics of the available data.

Among other developments, Ahn et al. [12] use a technique similar to the contextual pre-filtering to recommend advertisements to mobile users by taking into account user location, interest, and time, and Lombardi et al. [88] evaluate the effect of contextual information using a pre-filtering approach on the data obtained from an online retailer. Also, Baltrunas and Ricci [18, 25] take a somewhat different approach to contextual pre-filtering in proposing and evaluating the benefits of the *item splitting* technique, where each item is split into several fictitious items based on the different contexts in which these items can be consumed. Similarly to the item

splitting idea, Baltrunas and Amatriain [17] introduce the idea of *micro-profiling* (or user splitting), which splits the user profile into several (possibly overlapping) sub-profiles, each representing the given user in a particular context. The predictions are done using these contextual micro-profiles instead of a single user model. The idea of generalized contextual pre-filtering has also been adopted in various studies; for example, Zheng et al. [141] use a similar approach (called context "relaxation") for travel recommendations. Furthermore, Codina et al. [49] provide a different approach to generalize the pre-filtering approach; they leverage semantic similarities between different contextual conditions and compute the recommendations based on the ratings taken not just from the single contextual condition, but from the similar contexts as well.

Although the pre-filtering paradigm has been predominantly used in conjunction with the Explicit Static approach to modeling contextual factors, extending it to more complex context modeling scenarios, such as latent and dynamic cases, constitutes a highly promising future research direction. Ability to pre-filter training data based on any given context—explicit vs. latent, static vs. dynamic—is largely a data querying/matching problem, addressing which would automatically enable the use of any traditional (2D) recommendation algorithms.

Contextual Post-filtering As shown in Fig. 6b, the contextual post-filtering approach ignores contextual information when generating the initial ranking of the list of all the candidate items from which top-N recommendations can be made. Then, the contextual post-filtering approach adjusts the obtained recommendation list for each user using the contextual information. The recommendation list adjustments can be made by:

- *Filtering* out recommendations that are irrelevant in a given context, or
- *Re-ranking* the list of recommendations based on a given context.

For example, in a movie recommendation application, if a person wants to see a movie on a weekend, and on weekends she only watches comedies, the system can filter out all non-comedies from the recommended movie list. More generally, the basic idea for contextual post-filtering approaches is to analyze the contextual preference data for a given user in a given context to find specific item usage patterns (e.g., user Jane Doe watches only comedies on weekends) and then use these patterns to adjust the item list, resulting in more "contextual" recommendations, as depicted in Fig. 7.

As with many recommendation techniques, the contextual post-filtering approaches can be classified into heuristic and model-based techniques. *Heuristic* post-filtering approaches focus on finding common item characteristics (attributes) for a given user in a given context, while *model-based* post-filtering approaches build predictive models to calculate the probability with which the user chooses a certain type of item in a given context. These characteristics and probabilities are subsequently used to adjust the recommendations. For example, Panniello et al. [100] provide an experimental comparison of the pre-filtering method versus two post-filtering methods based on several real-world e-commerce datasets. The

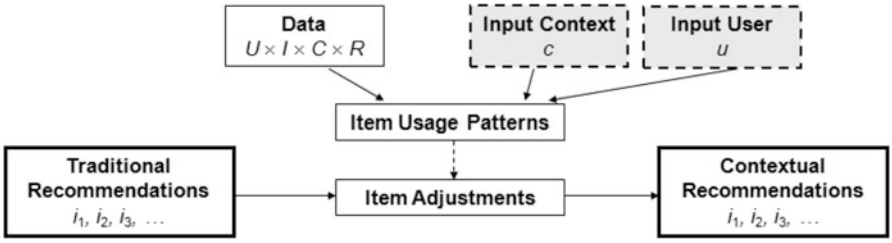

Fig. 7 Final phase of the contextual post-filtering approach: recommendation list adjustment

empirical results show that the best approach to use (pre- or post-filtering) highly depends on a given application.

Similar to the contextual pre-filtering, a major advantage of the contextual post-filtering approach is that it allows using any of the numerous canonical 2D recommendation techniques previously proposed in the literature [6]. Also, incorporating context generalization techniques into post-filtering techniques and extending them to dynamic and/or latent context settings constitute interesting issues for future research.

Contextual Modeling As shown in Fig. 6c, the contextual modeling approach incorporates the contextual information *directly* into the recommendation function as an explicit predictor of a user's rating for an item. While contextual pre-filtering and post-filtering approaches can use canonical 2D recommendation algorithms, the contextual modeling approach gives rise to truly multidimensional recommendation techniques, which essentially represent predictive models (built using decision trees, regressions, probabilistic models, deep learning models, or other techniques) or heuristic calculations that incorporate contextual information in addition to the user and item data. A significant number of novel contextual modeling approaches—based on a variety of heuristics as well as advanced predictive modeling techniques—have been developed over the last five years [102, 129]. In Sect. 4 we present both the classical approaches and more advanced machine learning techniques for contextual modeling, such as tensor factorization and deep learning.

4 Contextual Modeling Approaches

Contextual modeling, recognized by its effectiveness in improving performance of recommendations vis-a-vis alternative pre- and post-filtering methods in many cases, has become an increasingly popular paradigm for incorporating context into RS [102, 129]. In the contextual modeling approach, contextual attributes are used in the process of prediction of the ratings by recommendation systems. Contextual modeling challenges include the development of new techniques and mechanisms for: (i) integrating context into canonical recommendation models; (ii) improving

rating estimation methods by exploiting context and revealing hidden interactions between users, items, and contexts; and (iii) identifying the contextual factors that should be integrated into the recommendation model. In this section, we will describe different techniques used in CARS research along with their contribution in dealing with cold-start, data sparsity, and scalability problems and the type of contextual attributes used in them.

Classical Approaches A number of classical contextual modeling approaches that are based on heuristic as well as model-based techniques can be extended from the two-dimensional (2D) to the multidimensional recommendation settings. For example, [5] proposes a heuristic-based approach that extends the canonical 2D neighborhood-based approach [34, 112] to the multidimensional case, which includes the contextual information, by using an n-dimensional distance metric instead of the user-user or item-item similarity metrics traditionally used in such techniques. Another heuristic-based contextual modeling (CM) method is presented in [102], where four variants of the same CM method are considered. Each of these CM methods requires building a contextual profile $prof(u, c)$ for user u in context c, and then using the contextual profiles of all the users to find the N nearest neighbors of user u in terms of these profiles in context c.

In addition to the heuristic-based contextual modeling techniques, there have been several model-based techniques proposed in the literature. For example, Adomavicius and Tuzhilin [5] present a method of extending a regression-based Hierarchical Bayesian (HB) collaborative filtering model of estimating unknown ratings proposed by Ansari et al. [15] in order to incorporate additional contextual dimensions, such as time and location, into the HB model.

In order to efficiently model user, item, and context interactions, latent factor models suggest to embed users, items, and contexts as tensors (i.e., vectors) in a low-dimensional space of latent (or hidden) features. These models represent both users' preferences, their corresponding items' features and contexts' features in a unified way so that the relevance score of the user-item-context interaction can be measured as an inner product of their vectors in the latent feature space. Such predictive models include context-aware matrix factorization [2] and latent semantic models [122]. For example, matrix-factorization-based approach has been proposed by Baltrunas et al. [24] and Unger et al. [124], who extended the traditional matrix factorization technique to context-aware settings by introducing additional model parameters to model the interaction of the contextual factors with ratings.

In addition to extending the canonical $User \times Item$ model-based collaborative filtering techniques to incorporate the contextual dimensions, there have also been new techniques developed specifically for context-aware recommender systems based on the contextual modeling paradigm. For example, following this paradigm, Oku et al. [97] propose to incorporate additional contextual factors (such as time, companion, and weather) directly into recommendation space and use machine learning techniques to provide recommendations in a restaurant recommender system. In particular, they use the support vector machine (SVM) classification method, which views the set of liked items and the set of disliked items of a user in various

contexts as two sets of vectors in an n-dimensional space, and constructs a separating hyperplane in this space, which maximizes the separation between the two data sets. The resulting hyperplane represents a classifier for future recommendation decisions (i.e., a given item in a specific context will be recommended if it falls on the "like" side of the hyperplane, and will not be recommended if it falls on the "dislike" side). Furthermore, Oku et al. [97] empirically show that the context-aware SVM significantly outperforms the non-contextual SVM-based recommendation algorithm in terms of predictive accuracy and user's satisfaction with recommendations. Similarly, Yu et al. [138] use contextual modeling approach to provide content recommendations for smart phone users by introducing context as additional model dimensions and using hybrid recommendation technique (synthesizing content-based, Bayesian-classifier, and rule-based methods) to generate recommendations. Also, Hariri et al. [60] employ the Latent Dirichlet Allocation (LDA) model for use in context-aware recommender systems, which allows to model jointly the users, items, and the meta-data associated with contextual information. Finally, another model-based approach is presented in [1] where a Personalized Access Model (PAM) is presented that provides a set of personalized context-based services, including context discovery, contextualization, binding, and matching services, which can be combined to form and deploy context-aware recommender systems.

Tensor Factorization (TF) Tensor-based models provide a powerful set of tools for merging various types of contextual information, which increases flexibility, customizability, and quality of recommendation models [56]. Tensor Factorization (TF) extends the canonical two-dimensional Matrix Factorization (MF) problem to the n-dimensional problem by incorporating contextual information [72]. The resulting MD rating matrix $\mathcal{R}$ is factored into lower-dimensional representation, where the user, the item and each contextual dimension are represented with lower dimensional feature vectors, as follows:

$$\mathcal{R} \approx \mathcal{G} \times \mathbf{U} \times \mathbf{V} \times \mathbf{C} = \sum_{p=1}^{P} \sum_{q=1}^{Q} \sum_{l=1}^{L} g_{pql}\, \mathbf{u}^p \circ \mathbf{v}^q \circ \mathbf{c}^l$$

The rating matrix is illustrated in Fig. 8a, where $U \in \mathbf{R}^{M \times P}$, $V \in \mathbf{R}^{N \times Q}$, $C \in \mathbf{R}^{K \times L}$ are orthogonal matrices, having similar meaning as the latent feature matrices in the case of SVD. Tensor $\mathcal{G} \in \mathbf{R}^{P \times Q \times L}$ is called the *core tensor* of the Tensor Decomposition (TD). The elements of the core tensor $\mathcal{G}$ represent the latent interactions between different modes, where P, Q, and L are generally chosen to be smaller than the mode ranks M, N, and K of the original tensor. Thus, the core tensor $\mathcal{G}$ is often viewed as a compressed representation of the original tensor $\mathcal{R}$.

For example, the Multiverse model [72] is a generic TF-based context-aware model for the rating prediction task that integrates any contextual variables over a finite categorical domain. It demonstrates that high context influence leads

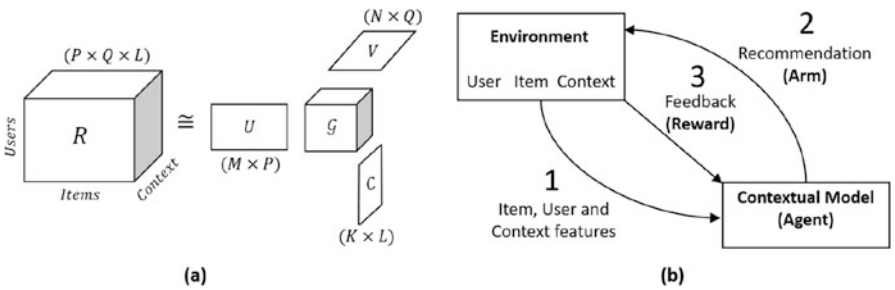

Fig. 8 Illustration of contextual modeling techniques: (**a**) Tensor factorization, and (**b**) Reinforcement learning

to significantly better rating prediction results using the proposed TF approach. While [72] focuses only on the rating prediction task, the CARTD (Context-Aware Recommendation Tensor Decomposition) model [109, 131] provides a generalized framework for an arbitrary number of contexts and targets an optimal ranking (top-N recommendation) instead of the rating prediction task.

TF can also be used in temporal models that are designed to learn time-evolving patterns in data by taking the time aspect into account [56]. For example, the use of a probabilistic TF model is proposed in several CARS frameworks [117, 136] to allow consideration of all the context simultaneously while modeling the system.

One of the main concerns for the higher-order models is the inevitable growth of computational complexity when the number of contextual dimensions increases. Since context is represented as a low-dimensional tensor, it raises various challenges in terms of context explainability, sparsity, potential for privacy issues, and structure. Improving the robustness of a tensor-based model is challenging due to the sparsity of the observed tensor and the multi-linear nature of TF. For example, Luan et al. [89] show that, if the tensor is big and sparse, then the degree of correlation between entities suffers. To overcome this, they propose a partition-based collaborative TF approach that considers correlations between two points of interest based on location. Specifically, a tensor is partitioned into sub-tensors and then the collaborative TF is applied. [45] proposed a model that combines TF and adversarial learning for context-aware recommendations and showed that their proposed method outperforms standard tensor-based methods and improves the robustness of a recommender model.

Traditional CARS methods model latent contextual information as vectors of high dimensionality and often ignore certain structural properties of latent contextual variables. While trying to reduce dimensionality of the contextual space, it is important to take into account the *structure* of latent contextual variables and the semantically meaningful interrelationships among them [93, 105]. For example, [133] proposed a bias tensor factorization model which uses encoded contextual features that are extracted from a regression tree.

Deep Learning (DL) Deep learning techniques have had a major impact on recommendation architectures [26, 139]. In context-aware recommender systems, deep-learning-based approaches are typically used for two purposes: (i) context representation and modeling and (ii) producing predictions by capturing preferences of users over both items and contexts. According to [26], the main reason why DL tends to improve recommendation accuracy is its power of extracting hidden features and jointly combining information from varying sources. Therefore, DL techniques are suitable for handling multi-dimensional problems and constraints that are complex and dynamic in nature.

Since some of the canonical recommender algorithms, such as matrix factorization and factorization machines, can also be expressed as neural architectures [64, 65], several DL-based CARS methods have been proposed to enhance these traditional neural architectures to model contextual representations [50, 62, 126, 135]. For example, [135] proposed a general context-aware algorithm based on factorization machines which combines automatic feature interaction modeling of factorization machine with convolutional neural networks. Figure 9 shows a general framework extending the Neural Collaborative Filtering (NCF) with contextual information [126]. The input of the network is s_u^{user}, s_i^{item}, and $s_c^{context}$, which are the representations of user profiles, item features, and context information, respectively. The embedding layer consists of low-dimensional representations for each of the inputs. The network is aimed to learn non-linear interactions between users, items, and contexts and can be used for multiple tasks (i.e., rating prediction, top-N recommendation, classification). Here, the scoring function is defined for the rating prediction task as follows:

$$\hat{r}_{uic} = f(U^T \cdot s_u^{user}, V^T \cdot s_i^{item}, C^T \cdot s_c^{context} \mid U, V, C, \theta)$$

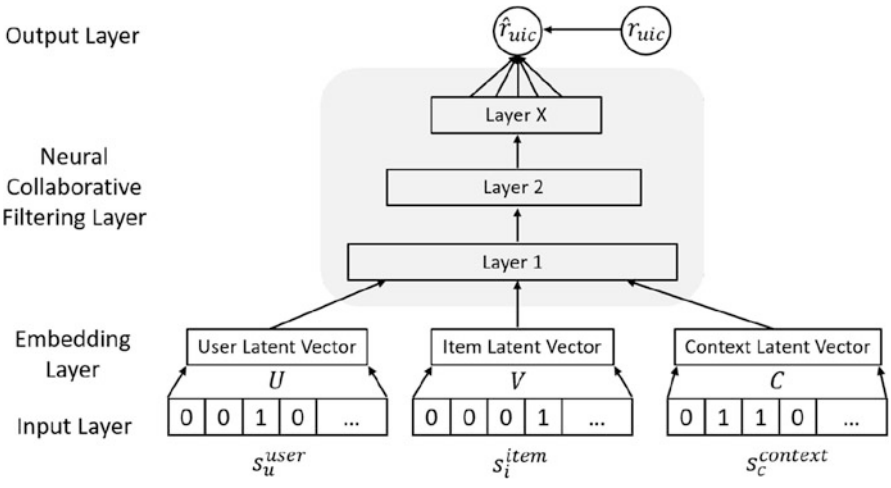

Fig. 9 Illustration of context-aware recommendation based on deep learning frameworks

where function $f(\cdot)$ represents the multilayer perceptron, and θ represents the parameters of this network. The network can be trained with weighted square loss (for explicit feedback) or binary cross-entropy loss (for implicit feedback).

Recent state-of-the-art context-aware methods represent the relations between users/items and contexts as a tensor, with which it is difficult to distinguish the impacts of different contextual factors and to model complex, non-linear interactions between contexts and users/items. Therefore, several attention-based recommendation models are used to enhance CARS through adaptively capturing the interactions between contexts and users/items and improve the interpretability of recommendations through identifying the most important contexts [52, 66, 92]. For example, [92] proposed a neural model, named Attentive Interaction Network (AIN), to enhance CARS through adaptively capturing the interactions between contexts and users/items, and [52] proposed an online neural-based recommendation model for ranking recommendations by capturing implicit interactions between users, items, and contexts. The implicit contexts were inferred by a Mixture Attentional Constrained Denoise AutoEncoder (MACDAE) model to combine multiple implicit contextual representations by using generative models. They assumed that different implicit contextual representations contribute differently to the final latent contextual representation and, therefore, their relative contribution can be learned automatically using the attention mechanism.

Context-aware sequential recommendations have been extensively used to monitor the evolution of user tastes over time, which helps to improve the quality of contextual recommendations [107]. Several DL-based techniques can be used to model user activity across time. For example, Recurrent Neural Networks (RNNs) can be used to estimate the next event in a session for a user and can implicitly capture contextual information and combine it with item embeddings [30, 87, 118]. RNNs have positive effects on coverage of recommendations and short-term predictions compared to conventional nearest-neighbor and matrix factorization-based approaches. Such success originates from RNNs' ability to take into account the evolution of users tastes and co-evolution between user, item, and context latent factors.

Reinforcement Learning (RL) Most recommendation models consider the recommendation process as static, which makes it difficult to capture users' temporal intentions and to respond to them in a timely manner. In recent years, RL has begun to garner attention [54, 98, 140, 143] in making contextualized recommendations. For example, [54] proposed a contextual bandit algorithm to decide which content to recommend to users, where the reward function takes into consideration contextual information, and [84] proposed a framework for online learning and adaptation to sequential preferences within a listening session in order to tailor the playlist to the listener's current context.

Unlike supervised learning (e.g., classification) that considers prescribed training data, a reinforcement learning (RL) algorithm actively explores its environment to gather information and exploits the learnt knowledge to dynamically make decisions or predictions. Multi-armed bandit is the most thoroughly studied RL problem.

It is inspired by slot machines in a casino: for a bandit (slot) machine with M arms, pulling arm i will result in a random payoff (reward) r, sampled from an unknown and arm-specific distribution p_i. The objective is to maximize the total reward of the user over a given number of interactions. Figure 8b provides a high-level illustration of contextual modeling recommendation approach using RL. The environment includes information about users, items, and contexts, and a decision-making agent represents the contextual modeling algorithm that produces a list of recommendations. Each time a user requests a recommendation, the information about the user, the contextual factors, and features of candidate items are passed to the agent. The agent will select the best action (i.e., recommending a list of items to a user), fetch user feedback as reward, and observe the new state of the environment. Specifically, the feedback of the user (reward function) is calculated based on the metric being optimized in the environment and the context of the user. Finally, the recommendation list and the user's feedback will be stored in the memory of the agent. Importantly, rather than learn from historical data, the only way the agent can learn which actions are good in different situations is to sequentially try them out and receive a reward from the user.

Previous works have used reinforcement learning to recommend web pages, travel information, books, videos, news, etc. For example, Tripathi et al. [123] combine the potential of reinforcement learning and deep bidirectional recurrent neural networks for automatic personalized video recommendation. They present a context-aware collaborative filtering approach, where the intensity of user's non-verbal emotional response toward the recommended video is captured through interactions and facial expression analysis. In [121], a Q-learning-based travel recommender is proposed, where trips are ranked using a linear function of several content and contextual features including trip duration, price, and country, and the weights are updated using user feedback. Shani et al. use a Markov decision process (MDP) to model the dynamics of user preference for book recommendations [116], where the purchase history is used to define the environment's states, and the generated profit is used as the reward. Liu et al. use MDP to recommend music based on a user's heart rate to help the user maintain it within the normal range [85]. States are defined as different levels of heart rate, and biofeedback is used as rewards. Chi et al. uses MDP to automatically generate playlist [47] and, similarly to [116], states are defined as mood categories of the recent listening history.

5 Discussion and Conclusions

Context-awareness is being recognized as an important issue in many recommendation applications, which is evidenced by the increasing number of papers being published on this topic. Looking at the current state of the art in context-aware recommender systems, the main research issues, challenges, and directions can be broadly classified into the following four general categories [8]:

- *Algorithms*, i.e., developing recommendation algorithms that can incorporate contextual information into recommender systems in advantageous ways.
- *Evaluation*, i.e., in-depth performance evaluation of various context-aware recommendation approaches and techniques, their benefits and limitations.
- *Engineering*, i.e., designing general-purpose architectures, frameworks, and approaches to facilitate the development, implementation, deployment, and use of context-aware recommendation capabilities.
- *Fundamentals*, i.e., deeper understanding the notion of context and modeling context in recommender systems.

Algorithms Not surprisingly, most of the existing research on context-aware recommender systems can be classified under the "Algorithms" category [8], i.e., researchers have focused primarily on how to take advantage of contextual information in order to improve the quality of recommendations for different recommendation tasks and applications. Compared to "Algorithms", the other three categories have been relatively under-explored, although there have been more work in several other areas in recent years.

Evaluation One representative example in the "Evaluation" category is the work by Panniello et al. [102], who performed a comprehensive evaluation and comparison of several contextual pre-filtering, post-filtering, and modeling techniques under variety of conditions, e.g., for different recommendation tasks ("find-all" vs. "top-k"), different recommendation utility metrics (accuracy vs. diversity), the granularity of the processed contextual information, as well as other characteristics. Among many findings, the comparison shows that there is no "universally" best technique under all evaluation dimensions. For example, the contextual modeling and post-filtering approaches demonstrate good accuracy performance in many situations, while the exact pre-filtering approach often exhibits better diversity. However, the contextual modeling approaches tend to achieve a comparatively good balance in the accuracy-diversity trade-off. Another example in the "Evaluation" category is the work by Campos et al. [40], who focused on exploring "time" as one of the most valuable and widely used contextual factors in many recommender systems applications. The authors review common evaluation practices and methodological issues related to the comparative evaluation of time-aware recommender systems. They also demonstrate that the choice of the evaluation conditions impacts the performance ranking of different recommendation strategies and propose a methodological framework for a robust and fair evaluation process. These works represent an important step in the direction of improved, standardized and, thus, more reproducible evaluation procedures for context-aware recommender systems.

Evaluating the CARS performance in real-world settings and in terms of domain-relevant metrics (e.g., based on economic or behavioral outcomes) represents another important direction for CARS research. As an example, [101] study how contextual information affects business outcomes in terms of sales volume, revenue, and customer trust in the provided recommendations using an empirical live study (A/B test) on a sample of customers of a major comics book publisher in Europe. In particular, [101] show that context-aware recommendation techniques outperform

canonical (non-contextual) approaches in terms of accuracy, trust, and several economics-based performance metrics across most of their experimental settings. Another interesting example of user studies with context-aware recommender systems is the work by Braunhofer et al. [31]. In this study, the users, after receiving a recommendation from a context-aware system that recommends points-of-interest, were asked to evaluate the system's performance on the following two dimensions: "Does this recommendation fit my preference?" (i.e., "personalization" performance) and "Is this recommendation well-chosen for the situation?" (i.e., "contextualization" performance). In this specific study, the authors show that their proposed technique is able to improve the baseline performance along one of these dimensions—contextualization. In summary, it is important for the CARS community to continue the lines of work described in [31, 101] and to provide strong additional evidence (e.g., via live controlled experiments) of the economic and usability advantages of CARS over the traditional recommendation methods.

One of the major issues that have slowed down the progress of the CARS field in the past was the availability of large-scale publicly available datasets on which novel CARS-based methods could be evaluated. This situation has improved significantly over the last few years when such datasets became publicly available. For example, DePaulMovie [142] and LDOS-CoMoDa [94] datasets contain movie ratings collected along with contextual information; InCarMusic [21], Frappe [20] and STS [32] datasets provide the apps usage logs. Furthermore, some recently published datasets with music listening history, such as the ones from Spotify [35] and Last.fm [128], contain large-scale information about sequence of music listening logs, where previous tracks and other user activity features can be considered as contextual information. Despite this progress, the CARS community would benefit significantly from additional publicly available, large-scale, context-oriented datasets, and this should also be an important priority for the research community.

Engineering Much of the work on context-aware recommender systems has been conceptual, where recommendation techniques are developed, tested on some (often limited) data, and shown to perform well in comparison to certain benchmarks. Historically, there has been little work done on the "Engineering" aspect for CARS, i.e., developing novel data structures, efficient storage methods, and new system architectures. One representative example of such system-building work is the study by Hussein et al. [67, 68], where the authors introduce a service-oriented architecture enabling to define and implement a variety of different "building blocks" for context-aware recommender systems, such as recommendation algorithms, context sensors, various filters and converters, in a modular fashion. These building blocks can then be combined and reused in many different ways into systems that can generate contextual recommendations. Another example of such work is Abbar et al. [1], where the authors present a service-oriented approach that implements the Personalized Access Model (PAM) previously proposed by the authors. In addition, there has been a number of research advances in database community in developing frameworks and techniques for building recommender systems functionality (including some context-aware functionality) directly into

relational database engines [81, 82, 113, 114], which provides a number of important benefits in terms of storage, query processing, query optimization, and others.

Another important "Engineering" aspect is to develop richer interaction capabilities with CARS that make recommendations more flexible. As compared to canonical recommender systems, context-aware recommenders have two important differences. The first is *increased complexity*, since CARS involve not only users and items in the recommendation process, but also various types of contextual information. Thus, the types of recommendations can be significantly more complex in comparison to the canonical non-contextual cases. For example, in a movie recommendation application, a certain user (e.g., Tom) may seek recommendations for him and his girlfriend of top 3 movies and the best times to see them over the weekend. The second difference is *increased interactivity*, since more information (i.e., context) usually needs to be elicited from the user in the CARS settings. For example, to utilize the available contextual information, a CARS system may need to elicit from the user (Tom) with whom he wants to see a movie (e.g., girlfriend) and when (e.g., over the weekend) before providing any context-specific recommendations. The combination of these two features calls for the development of more flexible recommendation methods that allow the user to express the types of recommendations that are of interest to them rather than consuming standard recommendations that are "hard-wired" into the recommendation engines provided by many current vendors. The second requirement of interactivity also calls for the development of tools allowing users to provide inputs into the recommendation process in an interactive and iterative manner, preferably via some well-defined user interfaces (UI).

Such flexible context-aware recommendations can be supported in several ways. First, Adomavicius et al. [7] developed a *recommendation query language* REQUEST that supports a wide variety of features and allows its users to express in a flexible manner a broad range of recommendations that are tailored to their own individual needs and, therefore, more accurately reflect their interests. In addition, Adomavicius et al. [7] provide a discussion of the expressive power of REQUEST and present a multidimensional recommendation algebra that provides the theoretical basis for this language. Another proposal to provide flexible recommendations is presented in [78], where the FlexRecs system and framework are described. FlexRecs approach supports flexible recommendations over structured data by decoupling the definition of a recommendation process from its execution. In particular, a recommendation can be expressed declaratively as a high-level parameterized workflow containing traditional relational operators and novel recommendation-specific operators that are combined together into a recommendation workflow.

Fundamentals "Fundamentals" represents arguably the least developed research direction. Context is a complex notion, and there are many diverse approaches of conceptualizing context in recommender systems, some of them still being debated among the researchers. Although the recommender systems community is gradually converging toward the common definition of context, and this chapter represents an attempt to integrate different approaches into one common frame-

work, additional work is still required for the CARS community to arrive at a more formalized definition of context and, therefore, many important questions still remain to be explored in a principled manner. For example, what are the underlying theoretical underpinnings for context relevance? Are there systematic approaches for identifying relevant contextual factors (i.e., on which to collect data)? If explicit contextual information is not available, when should we model context as a latent factor vs. ignore modeling the context altogether? What are the tradeoffs of different modelling assumptions (e.g., static vs. dynamic context)? The recommender systems community has been moving towards studying some of these questions. For example, Sect. 2 of this chapter covered certain foundational issues. Also, [57] examines in which stages of the recommender system life-cycle incorporation of contextual information into the recommender system is the most useful.

Furthermore, as the CARS field progresses, our understanding of what context is and how to deal with it is evolving. As stated in Sect. 2.3, most of the previous work on CARS has focused on the traditional Explicit-Static approach to modeling contextual factors. More recently, there has been significant work done on how to work with (and take advantage of) latent contextual information in CARS due, primarily, to the popularity of the DL-based methods, as discussed in Sect. 4. In contrast, the topic of *dynamic* context that is important in some applications, such as hashtag-based and news-based recommendations mentioned in Sect. 2.3, is barely explored in CARS. Moving forward, we believe that significantly more research is needed to better understand various aspects of latent and/or dynamic contextual information and how to leverage it in providing better recommendations. Another related and also under-explored research area is how to extract the contextual information from different data sources, such as user reviews, tweets, sensors, IoT devices, etc., that is not explicitly observed and/or recorded as context by the recommender system. This is especially useful when dealing with many different types of contextual factors "buried" in these data sources, most of which are not explicitly known in advance. We expect that much interesting research will be dedicated to these topics within the next several years, keeping the CARS field as vibrant as it has been over the last ten years.

In summary, the field of context-aware recommender systems (CARS) has matured very significantly over the last decade, and many leading companies have incorporated CARS into their recommender systems platforms. However, the field is still developing rapidly, as discussed in this section, and many interesting and practically important research problems still need to be explored further in a principled and comprehensive manner.

References

1. S. Abbar, M. Bouzeghoub, S. Lopez, Context-aware recommender systems: a service-oriented approach, in *VLDB PersDB Workshop* (2009)

2. M.H. Abdi, G. Okeyo, R.W. Mwangi, Matrix factorization techniques for context-aware collaborative filtering recommender systems: a survey (2018)
3. G.D. Abowd, C.G. Atkeson, J. Hong, S. Long, R. Kooper, M. Pinkerton, Cyberguide: a mobile context-aware tour guide. Wirel. Netw. **3**(5), 421–433 (1997)
4. G. Adomavicius, R. Sankaranarayanan, S. Sen, A. Tuzhilin, Incorporating contextual information in recommender systems using a multidimensional approach. ACM Trans. Inf. Syst. **23**(1), 103–145 (2005)
5. G. Adomavicius, A. Tuzhilin, Incorporating context into recommender systems using multidimensional rating estimation methods, in *Proceedings of the 1st International Workshop on Web Personalization, Recommender Systems and Intelligent User Interfaces (WPRSIUI 2005)* (2005)
6. G. Adomavicius, A. Tuzhilin, Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. IEEE Trans. Knowl. Data Eng. **17**(6), 734–749 (2005)
7. G. Adomavicius, A. Tuzhilin, R. Zheng, REQUEST: a query language for customizing recommendations. Inf. Syst. Res. **23**(1), 99–117 (2011)
8. G. Adomavicius, D. Jannach, Preface to the special issue on context-aware recommender systems. User Model. User-Adapt. Interact. **24**(1–2), 1–5 (2014)
9. G. Adomavicius, B. Mobasher, F. Ricci, A. Tuzhilin, Context-aware recommender systems. AI Mag. **32**(3), 67–80 (2011)
10. G. Adomavicius, A. Tuzhilin, Multidimensional recommender systems: a data warehousing approach, in *Electronic Commerce*, ed. by L. Fiege, G. Mühl, U. Wilhelm. Lecture Notes in Computer Science, vol. 2232 (Springer, Berlin, 2001), pp. 180–192
11. D. Agarwal, Scaling machine learning and statistics for web applications, in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '15* (Association for Computing Machinery, New York, 2015), p. 1621
12. H. Ahn, K. Kim, I. Han, Mobile advertisement recommender system using collaborative filtering: MAR-CF, in *Proceedings of the 2006 Conference of the Korea Society of Management Information Systems* (2006), pp. 709–715
13. E. Alpaydin, *Introduction to Machine Learning* (The MIT Press, London, 2004)
14. S.S. Anand, B. Mobasher, Contextual recommendation. WebMine, LNAI **4737**, 142–160 (2007)
15. A. Ansari, S. Essegaier, R. Kohli, Internet recommendation systems. J. Market. Res. **37**(3), 363–375 (2000)
16. L. Ardissono, A. Goy, G. Petrone, M. Segnan, P. Torasso, Intrigue: personalized recommendation of tourist attractions for desktop and hand held devices. Appl. Artif. Intell. **17**(8), 687–714 (2003)
17. L. Baltrunas, X. Amatriain, Towards time-dependant recommendation based on implicit feedback, in *Workshop on Context-Aware Recommender Systems (CARS 2009), New York* (2009)
18. L. Baltrunas, F. Ricci, Context-dependent items generation in collaborative filtering, in *Workshop on Context-Aware Recommender Systems (CARS 2009), New York* (2009)
19. L. Baltrunas, Keynote: contextualization at netflix, in *Workshop on Context-Aware Recommender Systems at the 13th ACM Conference on Recommender Systems, RecSys '19* (2019)
20. L. Baltrunas, K. Church, A. Karatzoglou, N. Oliver, Frappe: understanding the usage and perception of mobile app recommendations in-the-wild. Preprint, arXiv:1505.03014 (2015)
21. L. Baltrunas, M. Kaminskas, B. Ludwig, O. Moling, F. Ricci, A. Aydin, K.-H. Lüke, R. Schwaiger, Incarmusic: context-aware music recommendations in a car, in *E-Commerce and Web Technologies*, ed. by C. Huemer, T. Setzer. Lecture Notes in Business Information Processing, vol. 85 (Springer, Berlin, 2011), pp. 89–100

22. L. Baltrunas, B. Ludwig, S. Peer, F. Ricci, Context-aware places of interest recommendations for mobile users, in *Design, User Experience, and Usability. Theory, Methods, Tools and Practice*, ed. by A. Marcus. Lecture Notes in Computer Science, vol. 6769 (Springer, Berlin, 2011), pp. 531–540
23. L. Baltrunas, B. Ludwig, S. Peer, F. Ricci, Context relevance assessment and exploitation in mobile recommender systems. Pers. Ubiquit. Comput. **16**(5), 507–526 (2012)
24. L. Baltrunas, B. Ludwig, F. Ricci, Matrix factorization techniques for context aware recommendation, in *Proceedings of the Fifth ACM Conference on Recommender Systems, RecSys '11* (ACM, New York, 2011), pp. 301–304
25. L. Baltrunas, F. Ricci, Experimental evaluation of context-dependent collaborative filtering using item splitting. User Model. User-Adap. Inter. **24**(1–2), 7–34 (2014)
26. Z. Batmaz, A. Yurekli, A. Bilge, C. Kaleli, A review on deep learning for recommender systems: challenges and remedies. Artif. Intell. Rev. **52**(1), 1–37 (2019)
27. K. Bauman, A. Tuzhilin, Discovering contextual information from user reviews for recommendation purposes, in *Proceedings of the ACM RecSys Workshop on New Trends in Content Based Recommender Systems* (2014)
28. K. Bauman, A. Tuzhilin, Know thy context: parsing contextual information from user reviews for recommendation purposes. Inf. Syst. Res. (2021). Forthcoming
29. M. Bazire, P. Brézillon, Understanding context before using it, in *Proceedings of the 5th International Conference on Modeling and Using Context*, ed. by A. Dey et al. (Springer, Berlin, 2005)
30. A. Beutel, P. Covington, S. Jain, C. Xu, J. Li, V. Gatto, Ed.H. Chi, Latent cross: making use of context in recurrent recommender systems, in *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining* (2018), pp. 46–54
31. M. Braunhofer, M. Elahi, M. Ge, F. Ricci, Context dependent preference acquisition with personality-based active learning in mobile recommender systems, in *Learning and Collaboration Technologies. Technology-Rich Environments for Learning and Collaboration - First International Conference, LCT 2014, Held as Part of HCI International 2014, Heraklion, Crete, Greece, June 22–27, 2014, Proceedings, Part II*, ed. by P. Zaphiris, A. Ioannou. Lecture Notes in Computer Science, vol. 8524 (Springer, Berlin, 2014), pp. 105–116
32. M. Braunhofer, M. Elahi, F. Ricci, STS: a context-aware mobile recommender system for places of interest, in *Posters, Demos, Late-breaking Results and Workshop Proceedings of the 22nd Conference on User Modeling, Adaptation, and Personalization (UMAP2014), Aalborg, Denmark, July 7–11, 2014. CEUR Workshop Proceedings*, vol. 1181, ed. by I. Cantador, M. Chi, R. Farzan, R. Jäschke (2014). CEUR-WS.org
33. M. Braunhofer, M. Kaminskas, F. Ricci, Recommending music for places of interest in a mobile travel guide, in *Proceedings of the Fifth ACM Conference on Recommender Systems, RecSys '11* (ACM, New York, 2011), pp. 253–256
34. J.S. Breese, D. Heckerman, C. Kadie, Empirical analysis of predictive algorithms for collaborative filtering, in *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence, San Francisco, CA*, vol. 461 (1998), pp. 43–52
35. B. Brost, R. Mehrotra, T. Jehan, The music streaming sessions dataset, in *Proceedings of the 2019 Web Conference* (ACM, New York, 2019)
36. R. Bulander, M. Decker, G. Schiefer, B. Kolmel, Comparison of different approaches for mobile advertising, in *Proceedings of the Second IEEE International Workshop on Mobile Commerce and Services, WMCS '05* (IEEE Computer Society, Washington, DC, 2005), pp. 174–182
37. R. Burke, Hybrid recommender systems: survey and experiments. User Model. User-Adap. Inter. **12**(4), 331–370 (2002)
38. R. Burke, Hybrid web recommender systems, in *The Adaptive Web* (2007), pp. 377–408
39. R. Cai, C. Zhang, C. Wang, L. Zhang, W.-Y. Ma, Musicsense: contextual music recommendation using emotional allocation modeling, in *Proceedings of the 15th International Conference on Multimedia, MULTIMEDIA '07* (ACM, New York, 2007), pp. 553–556

40. P.G. Campos, F. Díez, I. Cantador, Time-aware recommender systems: a comprehensive survey and analysis of existing evaluation protocols. User Model. User-Adapt. Interact. **24**(1–2), 67–119 (2014)
41. I. Cantador, P. Castells, Semantic contextualisation in a news recommender system, in *Workshop on Context-Aware Recommender Systems (CARS 2009), New York* (2009)
42. F. Cena, L. Console, C. Gena, A. Goy, G. Levi, S. Modeo, I. Torre, Integrating heterogeneous adaptation techniques to build a flexible and usable mobile tourist guide. AI Commun. **19**(4), 369–384 (2006)
43. S. Chatterjee, A.S. Hadi, B. Price, *Regression Analysis by Example* (Wiley, New York, 2000)
44. S. Chaudhuri, U. Dayal, An overview of data warehousing and olap technology. ACM Sigmod Rec. **26**(1), 65–74 (1997)
45. H. Chen, J. Li, Adversarial tensor factorization for context-aware recommendation, in *Proceedings of the 13th ACM Conference on Recommender Systems* (2019), pp. 363–367
46. K. Cheverst, N. Davies, K. Mitchell, A. Friday, C. Efstratiou, Developing a context-aware electronic tourist guide: some issues and experiences, in *Proceedings of the SIGCHI conference on Human factors in computing systems* (ACM, New York, 2000), pp. 17–24
47. C. Chi, R.T. Tsai, J. Lai, J.Y. Hsu, A reinforcement learning approach to emotion-based automatic playlist generation, in *2010 International Conference on Technologies and Applications of Artificial Intelligence* (2010), pp. 60–65
48. K. Church, B. Smyth, P. Cotter, K. Bradley, Mobile information access: a study of emerging search behavior on the mobile internet. ACM Trans. Web **1**(1), 4-es (2007)
49. V. Codina, F. Ricci, L. Ceccaroni, Exploiting the semantic similarity of contextual situations for pre-filtering recommendation, in *User Modeling, Adaptation, and Personalization*, ed. by S. Carberry, S. Weibelzahl, A. Micarelli, G. Semeraro. Lecture Notes in Computer Science, vol. 7899 (Springer, Berlin, 2013), pp. 165–177
50. F.S. da Costa, P. Dolog, Collective embedding for neural context-aware recommender systems, in *Proceedings of the 13th ACM Conference on Recommender Systems, RecSys '19* (Association for Computing Machinery, New York, 2019), pp. 201–209
51. B. De Carolis, I. Mazzotta, N. Novielli, V. Silvestri, Using common sense in providing personalized recommendations in the tourism domain, in *Workshop on Context-Aware Recommender Systems (CARS 2009), New York* (2009)
52. X. Ding, J. Tang, T. Liu, C. Xu, Y. Zhang, F. Shi, Q. Jiang, D. Shen, Infer implicit contexts in real-time online-to-offline recommendation, in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD '19* (Association for Computing Machinery, New York, 2019), pp. 2336–2346
53. P. Dourish, What we talk about when we talk about context. Pers. Ubiquit. Comput. **8**(1), 19–30 (2004)
54. P. Dragone, R. Mehrotra, M. Lalmas, Deriving user- and content-specific rewards for contextual bandits, in *The World Wide Web Conference, WWW '19* (Association for Computing Machinery, New York, 2019), pp. 2680–2686
55. B. Fling, *Mobile Design and Development: Practical Concepts and Techniques for Creating Mobile Sites and Web Apps*, 1st edn. (O'Reilly Media, Sebastopol, 2009)
56. E. Frolov, I. Oseledets, Tensor methods and recommender systems. Wiley Interdiscip. Rev. Data Min. Knowl. Discov. **7**(3), e1201 (2017)
57. M. Gorgoglione, U. Panniello, A. Tuzhilin, Recommendation strategies in personalization applications. Inf. Manag. **56**(6), 103143 (2019)
58. C. Hansen, C. Hansen, L. Maystre, R. Mehrotra, B. Brost, F. Tomasi, M. Lalmas, Contextual and sequential user embeddings for large-scale music recommendation, in *Fourteenth ACM Conference on Recommender Systems* (2020), pp. 53–62
59. N. Hariri, B. Mobasher, R. Burke, Context-aware music recommendation based on latent topic sequential patterns, in *Proceedings of the Sixth ACM Conference on Recommender Systems, RecSys '12* (ACM, New York, 2012), pp. 131–138

60. N. Hariri, B. Mobasher, R. Burke, Query-driven context aware recommendation, in *Proceedings of the 7th ACM Conference on Recommender Systems, RecSys '13* (ACM, New York, 2013), pp. 9–16
61. N. Hariri, B. Mobasher, R. Burke, Y. Zheng, Context-aware recommendation based on review mining, in *Proceedings of the 9th Workshop on Intelligent Techniques for Web Personalization and Recommender Systems (ITWP 2011)*. Citeseer, (2011), p. 30
62. K. Haruna, M.A. Ismail, S. Suhendroyono, D. Damiasih, A. Pierewan, H. Chiroma, T. Herawan, Context-aware recommender system: a review of recent developmental process and future research direction. Appl. Sci. **7**(12), 1211 (2017)
63. R. Hastings, AWS re:Invent 2012, Day 1 Keynote (2012). http://www.youtube.com/watch?v=8FJ5DBLSFe4. YouTube video; see the video at 44:40 min
64. X. He, T.-S. Chua, Neural factorization machines for sparse predictive analytics, in *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval* (2017), pp. 355–364 (2017)
65. X. He, L. Liao, H. Zhang, L. Nie, X. Hu, T.-S. Chua, Neural collaborative filtering, in *Proceedings of the 26th International Conference on World Wide Web* (2017), pp. 173–182
66. B. Hu, C. Shi, W.X. Zhao, P.S. Yu, Leveraging meta-path based context for top- n recommendation with a neural co-attention model, in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD '18* (Association for Computing Machinery, New York, 2018), pp. 1531–1540
67. T. Hussein, T. Linder, W. Gaulke, J. Ziegler, Context-aware recommendations on rails, in *Workshop on Context-Aware Recommender Systems (CARS 2009), New York* (2009)
68. T. Hussein, T. Linder, W. Gaulke, J. Ziegler, Hybreed: a software framework for developing context-aware hybrid recommender systems. User Model. User-Adapt. Interact. **24**(1–2), 121–174 (2014)
69. D. Jannach, K. Hegelich, A case study on the effectiveness of recommendations in the mobile internet, in *Proceedings of the Third ACM Conference on Recommender Systems, RecSys '09* (ACM, New York, 2009), pp. 205–208
70. T. Jiang, A. Tuzhilin, Improving personalization solutions through optimal segmentation of customer bases. IEEE Trans. Knowl. Data Eng. **21**(3), 305–320 (2009)
71. M. Kaminskas, F. Ricci, Contextual music information retrieval and recommendation: State of the art and challenges. Comput. Sci. Rev. **6**(2–3), 89–119 (2012)
72. A. Karatzoglou, X. Amatriain, L. Baltrunas, N. Oliver, Multiverse recommendation: N-dimensional tensor factorization for context-aware collaborative filtering, in *Proceedings of the Fourth ACM Conference on Recommender Systems, RecSys '10* (ACM, New York, 2010), pp. 79–86
73. A. Karatzoglou, L. Baltrunas, K. Church, M. Böhmer, Climbing the app wall: enabling mobile app discovery through context-aware recommendations, in *Proceedings of the 21st ACM International Conference on Information and Knowledge Management, CIKM '12* (ACM, New York, 2012), pp. 2527–2530
74. R. Kimball, M. Ross, *The Data Warehousing Toolkit* (Wiley, New York, 1996)
75. J. Kiseleva, H.T. Lam, M. Pechenizkiy, T. Calders, Discovering temporal hidden contexts in web sessions for user trail prediction, in *Proceedings of the 22Nd International Conference on World Wide Web Companion, WWW '13 Companion, Republic and Canton of Geneva* (2013), pp. 1067–1074. International World Wide Web Conferences Steering Committee
76. D. Koller, M. Sahami, Toward optimal feature selection, in *Proceedings of the 13th International Conference on Machine Learning* (Morgan Kaufmann, Burlington, 1996), pp. 284–292
77. Y. Koren, Factorization meets the neighborhood: a multifaceted collaborative filtering model, in *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (ACM, New York, 2008), pp. 426–434
78. G. Koutrika, B. Bercovitz, H. Garcia-Molina, Flexrecs: expressing and combining flexible recommendations, in *Proceedings of the 35th SIGMOD international conference on Management of data* (ACM, Providence, 2009), pp. 745–758

79. N. Lathia, *The Anatomy of Mobile Location-Based Recommender Systems* (Springer US, Boston, 2015), pp. 493–510
80. H.J. Lee, S.J. Park, Moners: a news recommender for the mobile web. Expert Syst. Appl. **32**(1), 143–150 (2007)
81. J.J. Levandoski, M.D. Ekstrand, M. Ludwig, A. Eldawy, M.F. Mokbel, J. Riedl, Recbench: benchmarks for evaluating performance of recommender system architectures. Proc. VLDB **4**(11), 911–920 (2011)
82. J.J. Levandoski, A. Eldawy, M.F. Mokbel, M.E. Khalefa, Flexible and extensible preference evaluation in database systems. ACM Trans. Database Syst. **38**(3), 17 (2013)
83. P. Li, A. Tuzhilin, Latent multi-criteria ratings for recommendations, in *Proceedings of the 13th ACM Conference on Recommender Systems, RecSys '19* (Association for Computing Machinery, New York, 2019), pp. 428–431
84. E. Liebman, M. Saar-Tsechansky, P. Stone, The right music at the right time: adaptive personalized playlists based on sequence modeling. MIS Q. **43**(3), 765–786 (2019)
85. H. Liu, J. Hu, M. Rauterberg, Music playlist recommendation based on user heartbeat and music preference, in *2009 International Conference on Computer Technology and Development*, vol. 1 (2009), pp. 545–549
86. H. Liu, H. Motoda, *Feature Selection for Knowledge Discovery and Data Mining* (Springer, New York, 1998)
87. Q. Liu, S. Wu, D. Wang, Z. Li, L. Wang, Context-aware sequential recommendation, in *2016 IEEE 16th International Conference on Data Mining (ICDM)* (IEEE, Piscataway, 2016), pp. 1053–1058
88. S. Lombardi, S.S. Anand, M. Gorgoglione, Context and customer behavior in recommendation, in *Workshop on Context-Aware Recommender Systems (CARS 2009), New York* (2009)
89. W. Luan, G. Liu, C. Jiang, L. Qi, Partition-based collaborative tensor factorization for poi recommendation. IEEE/CAA J. Automat. Sin. **4**(3), 437–446 (2017)
90. T. Mahmood, F. Ricci, A. Venturini, Improving recommendation effectiveness: adapting a dialogue strategy in online travel planning. J. IT Tour. **11**(4), 285–302 (2009)
91. J. McInerney, B. Lacker, S. Hansen, K. Higley, H. Bouchard, A. Gruson, R. Mehrotra, Explore, exploit, and explain: personalizing explainable recommendations with bandits, in *Proceedings of the 12th ACM Conference on Recommender Systems, RecSys '18* (Association for Computing Machinery, New York, 2018), pp. 31–39
92. L. Mei, P. Ren, Z. Chen, L. Nie, J. Ma, J.-Y. Nie, An attentive interaction network for context-aware recommendations, in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management, CIKM '18* (Association for Computing Machinery, New York, 2018), pp. 157–166
93. H.F. Nweke, Y.W. Teh, M.A. Al-Garadi, U.R. Alo, Deep learning algorithms for human activity recognition using mobile and wearable sensor networks: State of the art and research challenges. Expert Syst. Appl. **105**, 233–261 (2018)
94. A. Odic, M. Tkalcic, J.F. Tasic, A. Kosir, Predicting and detecting the relevant contextual information in a movie-recommender system. Interact. Comput. **25**(1), 74–90 (2013)
95. S. Ojagh, M.R. Malek, S. Saeedi, S. Liang, An internet of things (IoT) approach for automatic context detection, in *2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)* (IEEE, Piscataway, 2018), pp. 223–226
96. M. Okawa, T. Iwata, T. Kurashima, Y. Tanaka, H. Toda, N. Ueda, Deep mixture point processes: Spatio-temporal event prediction with rich contextual information, in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD '19* (Association for Computing Machinery, New York, 2019), pp. 373–383
97. K. Oku, S. Nakajima, J. Miyazaki, S. Uemura, Context-aware SVM for context-dependent information recommendation, in *Proceedings of the 7th International Conference on Mobile Data Management* (2006), p. 109
98. R.O. Oyeleke, C.-Y. Yu, C.K. Chang, Situ-centric reinforcement learning for recommendation of tasks in activities of daily living in smart homes, in *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*, vol. 2 (IEEE, Piscataway, 2018), pp. 317–322

99. C. Palmisano, A. Tuzhilin, M. Gorgoglione, Using context to improve predictive modeling of customers in personalization applications. IEEE Trans. Knowl. Data Eng. **20**(11), 1535–1549 (2008)
100. U. Panniello, A. Tuzhilin, M. Gorgoglione, C. Palmisano, A. Pedone, Experimental comparison of pre-vs. post-filtering approaches in context-aware recommender systems, in *Proceedings of the 3rd ACM conference on Recommender Systems* (ACM, New York, 2009), pp. 265–268
101. U. Panniello, M. Gorgoglione, A. Tuzhilin, In carss we trust: How context-aware recommendations affect customers' trust and other business performance measures of recommender systems. Inf. Syst. Res. **27**(1), 182–196 (2016)
102. U. Panniello, A. Tuzhilin, M. Gorgoglione, Comparing context-aware recommender systems in terms of accuracy and diversity. User Model. User-Adapt. Interact. **24**(1–2), 35–65 (2014)
103. H.-S. Park, J.-O. Yoo, S.-B. Cho, A context-aware music recommendation system using fuzzy bayesian networks with utility theory, in *Proceedings of the Third International Conference on Fuzzy Systems and Knowledge Discovery, FSKD'06* (Springer, Berlin, 2006), pp. 970–979
104. M.-H. Park, J.-H. Hong, S.-B. Cho, Location-based recommendation system using bayesian user's preference model in mobile devices, in *Proceedings of the 4th International Conference on Ubiquitous Intelligence and Computing, UIC'07* (Springer, Berlin, 2007), pp. 1130–1139
105. D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, A.A. Efros, Context encoders: feature learning by inpainting, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2016), pp. 2536–2544
106. D.M. Pennock, E. Horvitz, Collaborative filtering by personality diagnosis: a hybrid memory- and model-based approach, in *IJCAI'99 Workshop: Machine Learning for Information Filtering* (1999)
107. M. Quadrana, P. Cremonesi, D. Jannach, Sequence-aware recommender systems. ACM Comput. Surv. **51**(4), 1–36 (2018)
108. S. Reddy, J. Mascia, Lifetrak: music in tune with your life, in *Proceedings of the 1st ACM International Workshop on Human-centered Multimedia, HCM '06* (ACM, New York, 2006), pp. 25–34
109. A. Rettinger, H. Wermser, Y. Huang, V. Tresp, Context-aware tensor decomposition for relation prediction in social networks. Soc. Netw. Anal. Min. **2**(4), 373–385 (2012)
110. F. Ricci, Q.N. Nguyen, Mobyrek: a conversational recommender system for on-the-move travelers, in *Destination Recommendation Systems: Behavioural Foundations and Applications* (2006), pp. 281–294
111. S. Sae-Ueng, S. Pinyapong, A. Ogino, T. Kato, Personalized shopping assistance service at ubiquitous shop space, in *Proceedings of the 22nd International Conference on Advanced Information Networking and Applications - Workshops, AINAW '08* (IEEE Computer Society, Washington, DC, 2008), pp. 838–843
112. B. Sarwar, G. Karypis, J. Konstan, J. Reidl, Item-based collaborative filtering recommendation algorithms, in *Proceedings of the 10th International Conference on World Wide Web* (ACM, New York, 2001), pp. 285–295
113. M. Sarwat, J. Avery, M.F. Mokbel, A recdb in action: recommendation made easy in relational databases. Proc. VLDB **6**(12), 1242–1245 (2013)
114. M. Sarwat, J.J. Levandoski, A. Eldawy, M.F. Mokbel, Lars*: an efficient and scalable location-aware recommender system. IEEE Trans. Knowl. Data Eng. **26**(6), 1384–1399 (2014)
115. G. Shani, A. Gunawardana, Evaluating recommendation systems, in *Recommender Systems Handbook* (Springer, Boston, 2011), pp. 257–297
116. G. Shani, D. Heckerman, R.I. Brafman, An MDP-based recommender system. J. Mach. Learn. Res. **6**(Sep), 1265–1295 (2005)
117. P. Sitkrongwong, S. Maneeroj, A. Takasu, Latent probabilistic model for context-aware recommendations, in *2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*, vol. 1 (IEEE, Piscataway, 2013), pp. 95–100

118. E. Smirnova, F. Vasile, Contextual sequence modeling for recommendation with recurrent neural networks, in *Proceedings of the 2nd Workshop on Deep Learning for Recommender Systems, DLRS 2017* (Association for Computing Machinery, New York, 2017), pp. 2–9
119. B. Smyth, P. Cotter, Mp3 - mobile portals, profiles and personalization, in *Web Dynamics* (Springer, Berlin, 2004), pp. 411–433
120. S. Solorio-Fernández, J.A. Carrasco-Ochoa, J.F. Martínez-Trinidad, A review of unsupervised feature selection methods. Artif. Intell. Rev. **53**(2), 907–948 (2020)
121. A. Srivihok, P. Sukonmanee, E-commerce intelligent agent: personalization travel support agent using q learning, in *Proceedings of the 7th International Conference on Electronic Commerce, ICEC '05* (Association for Computing Machinery, New York, 2005), pp. 287–292
122. X. Su, T.M. Khoshgoftaar, A survey of collaborative filtering techniques. Adv. Artif. Intell. **2009** (2009). https://doi.org/10.1155/2009/421425
123. A. Tripathi, T.S. Ashwin, R.M.R. Guddeti, EmoWare: a context-aware framework for personalized video recommendation using affective video sequences. IEEE Access **7**, 51185–51200 (2019)
124. M. Unger, A. Bar, B. Shapira, L. Rokach, Towards latent context-aware recommendation systems. Knowl. Based Syst. **104**(C), 165–178 (2016)
125. M. Unger, A. Tuzhilin, Hierarchical latent context representation for context-aware recommendations. IEEE Trans. Knowl. Data Eng. (2020). https://doi.org/10.1109/TKDE.2020.3022102
126. M. Unger, A. Tuzhilin, A. Livne, Context-aware recommendations based on deep learning frameworks. ACM Trans. Manag. Inf. Syst. **11**(2), 1–15 (2020)
127. M. Van Setten, S. Pokraev, J. Koolwaaij, Context-aware recommendations in the mobile tourist application compass, in *Adaptive Hypermedia*, ed. by W. Nejdl, P. De Bra (Springer, New York, 2004), pp. 235–244
128. G. Vigliensoni, I. Fujinaga, The music listening histories dataset, in *Proceedings of the 18th International Society for Music Information Retrieval Conference, Suzhou* (2017), pp. 96–102
129. N.M. Villegas, C. Sánchez, J. Díaz-Cely, G. Tamura, Characterizing context-aware recommender systems: a systematic literature review. Knowl. Based Syst. **140**, 173–200 (2018)
130. Q. Wang, H. Yin, T. Chen, Z. Huang, H. Wang, Y. Zhao, N.Q.V. Hung, Next point-of-interest recommendation on resource-constrained mobile devices, in *Proceedings of The Web Conference 2020* (2020), pp. 906–916
131. H. Wermser, A. Rettinger, V. Tresp, Modeling and learning context-aware recommendation scenarios using tensor decomposition, in *2011 International Conference on Advances in Social Networks Analysis and Mining* (IEEE, Piscataway, 2011), pp. 137–144
132. W. Woerndl, J. Huebner, R. Bader, D. Gallego-Vico, A model for proactivity in mobile, context-aware recommender systems, in *Proceedings of the Fifth ACM Conference on Recommender Systems, RecSys '11* (ACM, New York, 2011), pp. 273–276
133. W. Wu, J. Zhao, C. Zhang, F. Meng, Z. Zhang, Y. Zhang, Q. Sun, Improving performance of tensor-based context-aware recommenders using bias tensor factorization with context feature auto-encoding. Knowl. Based Syst. **128**, 71–77 (2017)
134. M. Xie, H. Yin, H. Wang, F. Xu, W. Chen, S. Wang, Learning graph-based poi embedding for location-based recommendation, in *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management* (2016), pp. 15–24
135. X. Xin, B. Chen, X. He, D. Wang, Y. Ding, J. Jose, CFM: convolutional factorization machines for context-aware recommendation, in *Proceedings of the 28th International Joint Conference on Artificial Intelligence* (AAAI Press, Palo Alto, 2019), pp. 3926–3932
136. L. Xiong, X. Chen, T.-K. Huang, J. Schneider, J.G. Carbonell, Temporal collaborative filtering with bayesian probabilistic tensor factorization, in *Proceedings of the 2010 SIAM International Conference on Data Mining* (SIAM, Philadelphia, 2010), pp. 211–222
137. H. Yin, Y. Sun, B. Cui, Z. Hu, L. Chen, LCARS: a location-content-aware recommender system, in *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2013), pp. 221–229

138. Z. Yu, X. Zhou, D. Zhang, C.Y. Chin, X. Wang, J. Men, Supporting context-aware media recommendations for smart phones. IEEE Pervasive Comput. **5**(3), 68–75 (2006)
139. S. Zhang, L. Yao, A. Sun, Y. Tay, Deep learning based recommender system: a survey and new perspectives. ACM Comput. Surv. **52**(1), 1–38 (2019)
140. G. Zheng, F. Zhang, Z. Zheng, Y. Xiang, N.J. Yuan, X. Xie, Z. Li, DRN: a deep reinforcement learning framework for news recommendation, in *Proceedings of the 2018 World Wide Web Conference* (2018), pp. 167–176
141. Y. Zheng, R. Burke, B. Mobasher, Differential context relaxation for context-aware travel recommendation, in *E-Commerce and Web Technologies*, ed. by C. Huemer, P. Lops. Lecture Notes in Business Information Processing, vol. 123 (Springer, Berlin 2012), pp. 88–99
142. Y. Zheng, B. Mobasher, R. Burke, CARSKit: a java-based context-aware recommendation engine, in *2015 IEEE International Conference on Data Mining Workshop (ICDMW)* (IEEE, Piscataway, 2015), pp. 1668–1671
143. F. Zhou, R. Yin, K. Zhang, G. Trajcevski, T. Zhong, J. Wu, Adversarial point-of-interest recommendation, in *The World Wide Web Conference* (2019), pp. 3462–34618