Context-Aware Recommender Systems

Gediminas Adomavicius, Alexander Tuzhilin

Abstract The importance of contextual information has been recognized by researchers and practitioners in many disciplines, including e-commerce personalization, information retrieval, ubiquitous and mobile computing, data mining, marketing, and management. While a substantial amount of research has already been performed in the area of recommender systems, most existing approaches focus on recommending the most relevant items to users without taking into account any additional contextual information, such as time, location, or the company of other people (e.g., for watching movies or dining out). In this chapter we argue that relevant contextual information does matter in recommender systems and that it is important to take this information into account when providing recommendations. We discuss the general notion of context and how it can be modeled in recommender systems. Furthermore, we introduce three different algorithmic paradigms – contextual prefiltering, post-filtering, and modeling – for incorporating contextual information into the recommendation process, discuss the possibilities of combining several context-aware recommendation techniques into a single unifying approach, and provide a case study of one such combined approach. Finally, we present additional capabilities for context-aware recommenders and discuss important and promising directions for future research.

Gediminas Adomavicius
Department of Information and Decision Sciences
Carlson School of Management, University of Minnesota
e-mail: gedas@umn.edu

Alexander Tuzhilin
Department of Information, Operations and Management Sciences
Stern School of Business, New York University
e-mail: atuzhili@stern.nyu.edu

1 Introduction and Motivation

The majority of 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 in 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 the case of personalized content delivery on a Web site, it is important to determine what content needs to be delivered (recommended) to a customer and when. More specifically, on weekdays a user might prefer to read world news when she logs on in the morning and the stock market report in the evening, and on weekends to read movie reviews and do shopping.

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.

More recently, companies started incorporating some contextual information into their recommendation engines. For example, when selecting a song for the customer, Sourcetone interactive radio (www.sourcetone.com) takes into the consideration the current mood of the listener (the context) that she specified. In case of music recommenders, some of the contextual information, such as listener's mood, may matter for providing better recommendations. However, it is still not clear if context matters for a broad range of other recommendation applications.

In this chapter we discuss the topic of *context-aware recommender systems* (*CARS*), address this and several other related questions, and demonstrate that, depending on the application domain and the available data, at least certain contextual information can be useful for providing better recommendations. We also propose three major approaches in which the contextual information can be incorporated into recommender systems, individually examine these three approaches, and also discuss how these three separate methods can be combined into one unified approach. Finally, the inclusion of the contextual information into the recommendation process presents opportunities for richer and more diverse interactions between the end-users and recommender systems. Therefore, in this chapter we also discuss novel flexible interaction capabilities in the form of the recommendation query language for context-aware recommender systems.

The rest of the chapter is organized as follows. Section 2 discusses the general notion of context as well as how it can be modeled in recommender systems. Section 3 presents three different algorithmic paradigms for incorporating contextual information into the recommendation process. Section 4 discusses the possibilities of combining several context-aware recommendation techniques and provides a case study of one such combined approach. Additional important capabilities for context-aware recommender systems are described in Section 5, and the conclusions and some opportunities for future work are presented in Section 6.

2 Context in Recommender Systems

Before discussing the role and opportunities of contextual information in recommender systems, in Section 2.1 we start by discussing the general notion of context. Then, in Section 2.2, we focus on recommender systems and explain how context is specified and modeled there.

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 (see, for example, http://context-07.ruc.dk) – is dedicated exclusively to studying this topic and incorporating it into various other branches of science, including medicine, law, and business. In reference to the latter, a well-known business researcher and practitioner C. K. Prahalad has stated that "the ability to reach out and touch customers anywhere at anytime means that companies must deliver not just competitive products but also unique, real-time customer experiences shaped by *customer context*" and that this would be the next main issue ("big thing") for the CRM practitioners [56].

Since context has been studied in multiple disciplines, 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 "conditions or circumstances which affect some thing" [69]. Therefore, there exist many definitions of context across various disciplines and even within specific subfields of these disciplines. Bazire and Brézillon [16] present and examine 150 different definitions of context from different fields. This is not surprising, given the complexity and the multifaceted nature of the concept. As Bazire and Brézillon [16] observe:

> "... it is difficult to find a relevant definition satisfying in any discipline. Is context a frame for a given object? Is it the set of elements that have any influence on the object? Is it possible to define context a priori or just state the effects a posteriori? Is it something static

> or dynamic? Some approaches emerge now in Artificial Intelligence [...]. In Psychology, we generally study a person doing a task in a given situation. Which context is relevant for our study? The context of the person? The context of the task? The context of the interaction? The context of the situation? When does a context begin and where does it stop? What are the real relationships between context and cognition?"

Since we focus on recommender systems in this paper and since the general concept of context is very broad, we try to focus on those fields that are directly related to recommender systems, such as data mining, e-commerce personalization, databases, information retrieval, ubiquitous and mobile context-aware systems, marketing, and management. We follow Palmisano et al. [53] in this section when describing these areas.

Data Mining. In the data mining community, context is sometimes defined as those events which characterize the life stages of a customer and that can determine a change in his/her preferences, status, and value for a company [17]. Examples of context include a new job, the birth of a child, marriage, divorce, and retirement. Knowledge of this contextual information helps (a) mining patterns pertaining to this particular context by focusing only on the *relevant data*; for example, the data pertaining to the daughter's wedding, or (b) selecting only *relevant results*, i.e., those data mining results that are applicable to the particular context, such as the discovered patterns that are related to the retirement of a person.

E-commerce Personalization. Palmisano et al. [53] use the *intent* of a purchase made by a customer in an e-commerce application as contextual information. Different purchasing intents may lead to different types of behavior. For example, the same customer may buy from the same online account different products for different reasons: a book for improving her personal work skills, a book as a gift, or an electronic device for her hobby. To deal with different purchasing intentions, Palmisano et al. [53] build a separate profile of a customer for each purchasing context, and these separate profiles are used for building separate models predicting customer's behavior in specific contexts and for specific segments of customers. Such contextual segmentation of customers is useful, because it results in better predictive models across different e-commerce applications [53].

Recommender systems are also related to e-commerce personalization, since personalized recommendations of various products and services are provided to the customers. The importance of including and using the contextual information in recommendation systems has been demonstrated in [3], where the authors presented a multidimensional approach that can provide recommendations based on contextual information in addition to the typical information on users and items used in many recommendation applications. It was also demonstrated by Adomavicius et al. [3] that the contextual information does matter in recommender systems: it helps to increase the quality of recommendations in certain settings.

Similarly, Oku et al. [52] incorporate additional contextual dimensions (such as time, companion, and weather) into the recommendation process and use machine learning techniques to provide recommendations in a restaurant recommender system. They empirically show that the context-aware approach significantly outper-

forms the corresponding non-contextual approach in terms of recommendation accuracy and user's satisfaction with recommendations.

Since we focus on the use of context in recommender systems in this paper, we will describe these and similar approaches later in the chapter.

Ubiquitous and mobile context-aware systems. In the literature pertaining to the context-aware systems, context was initially defined as the location of the user, the identity of people near the user, the objects around, and the changes in these elements [62]. Other factors have been added to this definition subsequently. For instance, Brown et al. [22] include the date, the season, and the temperature. Ryan et al. [60] add the physical and conceptual statuses of interest for a user. Dey et al. [32] include the user's emotional status and broaden the definition to any information which can characterize and is relevant to the interaction between a user and an application. Some associate the context with the user [32, 34], while others emphasize how context relates to the application [59, 68]. More recently, a number of other techniques for context-aware systems have been discussed in research literature, including hybrid techniques for mobile applications [58, 70] and graphical models for visual recommendation [19].

This contextual information is crucial for providing a broad range of Location-Based Services (LBSes) to the mobile customers [63]. For example, a Broadway theater may want to recommend heavily discounted theater tickets to the Time Square visitors in New York thirty minutes before the show starts (since these tickets will be wasted anyway after the show begins) and send this information to the visitors' smart phones or other communication devices. Note that time, location, and the type of the communication device (e.g., smart phone) constitute contextual information in this application. Brown et al. [21] introduce another interesting application that allows tourists interactively share their sightseeing experiences with remote users, demonstrating the value that context-aware techniques can provide in supporting social activities.

A survey of context-aware mobile computing research can be found in [29], which discusses different models of contextual information, context-sensing technologies, different possible architectures, and a number of context-aware application examples.

Databases. Contextual capabilities have been added to some of the database management systems by incorporating user preferences and returning different answers to database queries depending on the context in which the queries have been expressed and the particular user preferences corresponding to specific contexts. More specifically, in Stephanidis et al. [65] a set of contextual parameters is introduced and preferences are defined for each combination of regular relational attributes and these contextual parameters. Then Stephanidis et al. [65] present a context-aware extension of SQL to accommodate for such preferences and contextual information. Agrawal et al. [7] present another method for incorporating context and user preferences into query languages and develop methods of reconciling and ranking different preferences in order to expeditiously provide ranked answers to contextual queries. Mokbel and Levandoski [51] describe the context-aware and location-aware

database server CoreDB and discuss several issues related to its implementation, including challenges related to context-aware query operators, continuous queries, multi-objective query processing, and query optimization.

Information Retrieval. Contextual information has been proven to be helpful in information retrieval and access [39], although most existing systems base their retrieval decisions solely on queries and document collections, whereas information about search context is often ignored [9]. The effectiveness of a proactive retrieval system depends on the ability to perform context-based retrieval, generating queries which return context-relevant results [45, 64]. In Web searching, context is considered as the set of topics potentially related to the search term. For instance, Lawrence [44] describes how contextual information can be used and proposes several specialized domain-specific context-based search engines. Integration of context into the Web services composition is suggested by Maamar et al. [50]. Most of the current context-aware information access and retrieval techniques focus on the short-term problems and immediate user interests and requests (such as "find all files created during a spring meeting on a sunny day outside an Italian restaurant in New York"), and are not designed to model long-term user tastes and preferences.

Marketing and Management. Marketing researchers have maintained that the purchasing process is contingent upon the context in which the transaction takes place, since the same customer can adopt different decision strategies and prefer different products or brands depending on the context [18, 49]. According to Lilien et al. [46], "consumers vary in their decision-making rules because of the usage situation, the use of the good or service (for family, for gift, for self) and purchase situation (catalog sale, in-store shelf selection, and sales person aided purchase)." Therefore, accurate predictions of consumer preferences should depend on the degree to which we have incorporated the relevant contextual information. In the marketing literature, context has been also studied in the field of behavioral decision theory. In Lussier and Olshavsky [49], context is defined as a task complexity in the brand choice strategy.

The context is defined in Prahalad [56] as "the precise physical location of a customer at any given time, the exact minute he or she needs the service, and the kind of technological mobile device over which that experience will be received." Further, Prahalad [56] focuses on the applications where the contextual information is used for delivering "unique, real-time customer experiences" based on this contextual information, as opposed to the delivery of competitive products. Prahalad [56] provides an example about the case when he left his laptop in a hotel in Boston, and was willing to pay significant premiums for the hotel shipping the laptop to him in New York in that particular context (he was in New York and needed the laptop really urgently in that particular situation).

To generalize his statements, Prahalad [56] really distinguishes among the following three dimensions of the contextual information: temporal (when to deliver customer experiences), spatial (where to deliver), and technological (how to deliver). Although Prahalad focuses on the real-time experiences (implying that it is really the present time, "now"), the temporal dimension can be generalized to the

past and the future (e.g., I want to see a movie tomorrow in the evening).

As this section clearly demonstrates, context is a multifaceted concept used across various disciplines, each discipline taking a certain angle and putting its "stamp" on this concept. To bring some "order" to this diversity of views, Dourish [33] 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 attributes, the structure (or schema, using database terminology) of which does not change significantly over time. In other words, the representational view assumes that the contextual attributes 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 [33] 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 also different activities giving rise to different contexts.

In the next section, we take all these different definitions and approaches to context and adapt them to the idiosyncratic needs of recommender systems. As a result, we will also revise and enhance the prior definitions of context used in recommender systems, including those provided in [3, 52, 71].

2.2 Modeling Contextual Information in Recommender Systems

Recommender systems emerged as an independent research area in the mid-1990s, when researchers and practitioners started focusing on recommendation problems that explicitly rely on the notion of ratings as a way to capture user preferences for different items. For example, in case of a movie recommender system, John Doe may assign a rating of 7 (out of 10) for the movie "Gladiator," i.e., set R_{movie}(John_Doe, Gladiator)=7. The recommendation process typically starts 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 tries to estimate the rating function R

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

for the (user, item) pairs that have not been rated yet by the users. Here *Rating* is a totally ordered set (e.g., non-negative integers or real numbers within a certain range), 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 recommender system can recommend the highest-rated item (or k highest-rated items) for each user. We call such systems *traditional* or *two-dimensional* (2D) since they consider only the *User* and *Item* dimensions in the recommendation process.

In other words, in its most common formulation, the recommendation problem is reduced to the problem of estimating ratings for the items that have not been seen by a user. This estimation is usually based on the ratings given by this user to other items, ratings given to this item by other users, and possibly on some other information as well (e.g., user demographics, item characteristics). Note that, while a substantial amount of research has been performed in the area of recommender systems, the vast majority of the existing approaches focus on recommending items to users or users to items and do not take into the consideration any additional contextual information, such as time, place, the company of other people (e.g., for watching movies). Motivated by this, in this chapter we explore the area of *context-aware recommender systems* (*CARS*), which deal with modeling and predicting user tastes and preferences by incorporating available contextual information into the recommendation process as explicit additional categories of data. These long-term preferences and tastes are usually expressed as *ratings* and are modeled as the function of not only items and users, but also of the context. In other words, ratings are defined with the *rating function* as

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

where *User* and *Item* are the domains of users and items respectively, *Rating* is the domain of ratings, and *Context* specifies the contextual information associated with the application. To illustrate these concepts, consider the following example.

Example 1. Consider the application for recommending movies to users, where users and movies are described as relations having the following attributes:

- Movie: the set of all the movies that can be recommended; it is defined as Movie(MovieID, Title, Length, ReleaseYear, Director, Genre).
- User: the people to whom movies are recommended; it is defined as User(UserID, Name, Address, Age, Gender, Profession).

Further, the contextual information consists of the following three types that are also defined as relations having the following attributes:

- Theater: the movie theaters showing the movies; it is defined as Theater(TheaterID, Name, Address, Capacity, City, State, Country).
- Time: the time when the movie can be or has been seen; it is defined as Time(Date, DayOfWeek, TimeOfWeek, Month, Quarter, Year). Here, attribute DayOfWeek has values Mon, Tue, Wed, Thu, Fri, Sat, Sun, and attribute TimeOfWeek has values "Weekday" and "Weekend".
- Companion: represents a person or a group of persons with whom one can see a movie. It is defined as Companion(companionType), where attribute companionType has values "alone", "friends", "girlfriend/boyfriend", "family", "co-workers", and "others".

Then the rating assigned to a movie by a person also depends on where and how the movie has been seen, with whom, and at what time. For example,

the type of movie to recommend to college student Jane Doe can differ significantly depending on whether she is planning to see it on a Saturday night with her boyfriend vs. on a weekday with her parents.

As we can see from this example and other cases, the contextual information Context can be of different *types*, each type defining a certain aspect of context, such as time, location (e.g., Theater), companion (e.g., for seeing a movie), purpose of a purchase, etc. Further, each contextual type can have a complicated structure reflecting complex nature of the contextual information. Although this complexity of contextual information 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 [3] and [53]. For instance, the three contexts from Example 1 can have the following hierarchies associated with them: *Theater*: TheaterID $\rightarrow$ City $\rightarrow$ State $\rightarrow$ Country; *Time*: Date $\rightarrow$ DayOfWeek $\rightarrow$ TimeOfWeek, Date $\rightarrow$ Month $\rightarrow$ Quarter $\rightarrow$ Year.[1]

Furthermore, we follow the representational view of Dourish [33], as described at the end of Section 2.1, and assume that the context is defined with a predefined set of observable attributes, the structure of which does not change significantly over time. Although there are some papers in the literature that take the interactional approach to modeling contextual recommendations, such as [11] that models context through a short-term memory (STM) interactional approach borrowed from psychology, most of the work on context-aware recommender systems follows the representational view. As stated before, we also adopt this representational view in this chapter and assume that there is a predefined finite set of contextual types in a given application and that each of these types has a well-defined structure.

More specifically, we follow Palmisano et al. [53], and also Adomavicius et al. [3] to some extent, in this paper and define the contextual information with a *set* of *contextual dimensions* $\boldsymbol{K}$, each contextual dimension K in $\boldsymbol{K}$ being defined by a set of q attributes $K = (K^1, \ldots, K^q)$ having a hierarchical structure and capturing a particular type of context, such as *Time* or *CommunicatingDevice*. The values taken by attribute K^q define *finer* (more *granular*) levels, while K^1 values define *coarser* (less granular) levels of contextual knowledge. For example, Figure 1(a) presents a four-level hierarchy for the contextual attribute K specifying the intent of a purchasing transaction in an e-retailer application. While the root (coarsest level) of the hierarchy for K defines purchases in all possible contexts, the next level is defined by attribute $K^1 = \{Personal, Gift\}$, which labels each customer purchase either as a personal purchase or as a gift. At the next, finer level of the hierarchy, "Personal" value of attribute K^1 is further split into a more detailed personal context: personal purchase made for the work-related or other purposes. Similarly, the *Gift* value for K^1 can be split into a gift for a partner or a friend and a gift for parents or others.

[1] For the sake of completeness, we would like to point out that not only the contextual dimensions, but also the traditional User and Item dimensions can have their attributes form hierarchical relationships. For example, the main two dimensions from Example 1 can have the following hierarchies associated with them: *Movie*: MovieID $\rightarrow$ Genre; *User*: UserID $\rightarrow$ Age, UserID $\rightarrow$ Gender, UserID $\rightarrow$ Profession.

Thus, the K^2 level is $K^2 = \{$*PersonalWork*, *PersonalOther*, *GiftPartner/Friend*, *GiftParent/Other*$\}$. Finally, attribute K^2 can be split into further levels of hierarchy, as shown in Figure 1(a).[2]

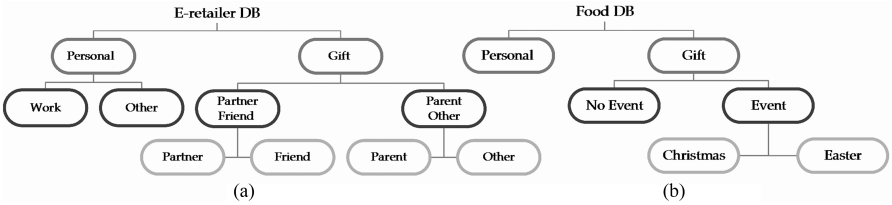


Fig. 1 Contextual information hierarchical structure: (a) e-retailer dataset, (b) food dataset [53].

Contextual information was also defined in [3] as follows. In addition to the classical *User* and *Item* dimensions, additional contextual dimensions, such as *Time*, *Location*, etc., were also introduced using the OLAP-based[3] *multidimensional data* (*MD*) *model* widely used in the data warehousing applications in databases [28, 40]. Formally, let $D_1, D_2, \ldots, D_n$ be dimensions, two of these dimensions being *User* and *Item*, and the rest being contextual. Each dimension D_i is a subset of a Cartesian product of some attributes (or fields) $A_{ij}, (j = 1, \ldots, k_i)$, i.e., $D_i \subseteq A_{i1} \times A_{i2} \times \ldots \times A_{ik_i}$, where each attribute defines a domain (or a set) of values. Moreover, one or several attributes form a *key*, i.e., they uniquely define the rest of the attributes [57]. In some cases, a dimension can be defined by a single attribute, and k_i =1 in such cases. For example, consider the three-dimensional recommendation space $User \times Item \times Time$, where the *User* dimension is defined as $User \subseteq UName \times Address \times Income \times Age$ and consists of a set of users having certain names, addresses, incomes, and being of a certain age. Similarly, the *Item* dimension is defined as $Item \subseteq IName \times Type \times Price$ and consists of a set of items defined by their names, types and the price. Finally, the *Time* dimension can be defined as $Time \subseteq Year \times Month \times Day$ and consists of a list of days from the starting to the ending date (e.g. from January 1, 2003 to December 31, 2003).

Given dimensions $D_1, D_2, \ldots, D_n$, 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 $D_1 \times \ldots \times D_n$ as

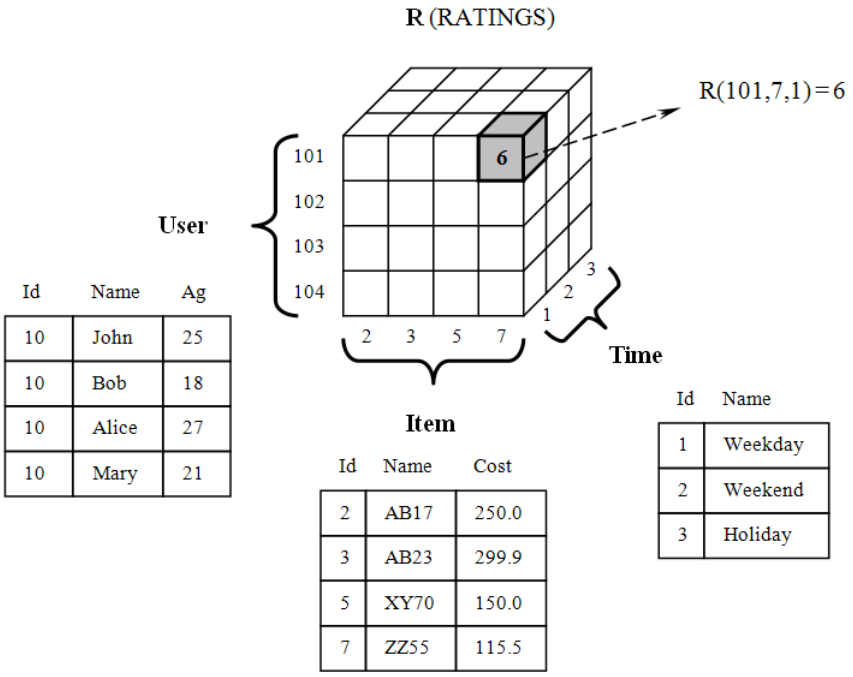
$$R : D_1 \times \ldots \times D_n \rightarrow Rating.$$

[2] For simplicity and illustration purposes, this figure uses only two-way splits. Obviously, three-way, four-way and, more generally, multi-way splits are also allowed.

[3] OLAP stands for OnLine Analytical Processing, which represents a popular approach to manipulation and analysis of data stored in multi-dimensional cube structures and which is widely used for decision support.

For instance, continuing the *User* × *Item* × *Time* example considered above, we can define a rating function R on the recommendation space *User* × *Item* × *Time* specifying how much user $u \in User$ liked item $i \in Item$ at time $t \in Time$, $R(u,i,t)$.

Visually, ratings $R(d_1,\ldots,d_n)$ on the recommendation space $S = D_1 \times D_2 \times \ldots \times D_n$ can be stored in a multidimensional cube, such as the one shown in Figure 2. For example, the cube in Figure 2 stores ratings $R(u,i,t)$ for the recommendation space *User* × *Item* × *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 Figure 2 means that for the user with User ID 101 and the item with Item ID 7, rating 6 was specified during the weekday.



Id	Name	Ag
10	John	25
10	Bob	18
10	Alice	27
10	Mary	21

Id	Name	Cost
2	AB17	250.0
3	AB23	299.9
5	XY70	150.0
7	ZZ55	115.5

Id	Name
1	Weekday
2	Weekend
3	Holiday

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

The rating function R introduced above is usually defined as a partial function, where the initial set of ratings is known. Then, as usual in recommender systems, the goal is to estimate the unknown ratings, i.e., make the rating function R total.

The main difference between the multidimensional (MD) contextual model described above and the previously described contextual model lies in that contextual information in the MD model is defined using classical OLAP hierarchies, whereas the contextual information in the previous case is defined with more general hierarchical taxonomies, that can be represented as trees (both balanced and unbalanced), directed acyclic graphs (DAGs), or various other types of taxonomies. Further, the

ratings in the MD model are stored in the multidimensional cubes, whereas the ratings in the other contextual model are stored in more general hierarchical structures.

We would also like to point out that not all contextual information 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) purpose of buying the book (possible options: for work, for leisure, ...); (b) planned reading time (weekday, weekend, ...); (c) planned reading place (at home, at school, on a plane, ...); (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 example, in the previous example, the value of a stock market can be less relevant as contextual information than the purpose of buying a book. There are several approaches to determining the relevance of a given type of contextual information. In particular, the relevance determination can either be done *manually*, e.g., using domain knowledge of the recommender system's designer or a market expert in a given application domain, or *automatically*, e.g., using numerous existing feature selection procedures from machine learning [41], data mining [47], and statistics [27], based on existing ratings data during the data preprocessing phase. The detailed discussion of the specific feature selection procedures is beyond the scope of this discussion; in the remainder of this chapter we will assume that only the relevant contextual information is stored in the data.

2.3 Obtaining Contextual Information

The contextual information can be obtained in a number of ways, including:

- *Explicitly*, i.e., by directly approaching relevant people and other sources of contextual information and explicitly gathering this information either by asking direct questions or eliciting this information through other means. For example, a website may obtain contextual information by asking a person to fill out a web form or to answer some specific questions before providing access to certain web pages.
- *Implicitly* from the data or the environment, such as a change in location of the user detected by a mobile telephone company. Alternatively, temporal contextual information can be implicitly obtained from the timestamp of a transaction. Nothing needs to be done in these cases in terms of interacting with the user or other sources of contextual information – the source of the implicit contextual information is accessed directly and the data is extracted from it.
- *Inferring* the context using statistical or data mining methods. 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 inferred with reasonable accuracy by observing the TV programs watched and the channels visited using various data mining methods. In order to infer this contextual information, it is necessary to build a predictive model (i.e., a classifier)

and train it on the appropriate data. The success of inferring this contextual information depends very significantly on the quality of such classifier, and it also varies considerably across different applications. For example, it was demonstrated in [53] that various types of contextual information can be inferred with a reasonably high degree of accuracy in certain applications and using certain data mining methods, such as Naïve Bayes classifiers and Bayesian Networks.

Finally, the contextual information can be "hidden" in the data in some latent form, and we can use it *implicitly* to better estimate the unknown ratings *without explicitly knowing* this contextual information. For instance, in the previous example, we may want to estimate how much a person likes a particular TV program by modeling the member of the household (husband, wife, etc.) watching the TV program as a latent variable. It was also shown in [53] that this deployment of latent variables, such as intent of purchasing a product (e.g., for yourself vs. as a gift, work-related vs. pleasure, etc.), whose true values were unknown but that were explicitly modeled as a part of a Bayesian Network (BN), indeed improved the predictive performance of that BN classifier. Therefore, even without any *explicit* knowledge of the contextual information (e.g., which member of the household is watching the program), recommendation accuracy can still be improved by modeling and inferring this contextual information implicitly using carefully chosen learning techniques (e.g., by using latent variables inside well-designed recommendation models). A similar approach of using latent variables is presented in [11].

As explained in Section 2.1, we focus on the representational view of Dourish [33], and assume that the context is defined with a *predefined* set of contextual attributes, the structure of which does not change over time. The implication of this assumption is that we need to identify and acquire contextual information before actual recommendations are made. If the acquisition process of this contextual information is done explicitly or even implicitly, it should be conducted as a part of the overall data collection process. All this implies that the decisions of which contextual information should be relevant and collected for an application should be done at the application design stage and well in advance of the time when actual recommendations are provided.

One methodology of deciding which contextual attributes should be used in a recommendation application (and which should not) is presented in [3]. In particular, Adomavicius et al. [3] propose that a wide range of contextual attributes should be initially selected by the domain experts as possible candidates for the contextual attributes for the application. For example, in a movie recommendation application described in Example 1, we can initially consider such contextual attributes as Time, Theater, Companion, Weather, as well as a broad set of other contextual attributes that can possibly affect the movie watching experiences, as initially identified by the domain experts for the application. Then, after collecting the data, including the rating data and the contextual information, we may apply various types of statistical tests identifying which of the chosen contextual attributes 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 attribute. For example, we may apply pairwise t-tests to see if good weather vs. bad weather or

seeing a movie alone vs. with a companion significantly affect the movie watching experiences (as indicated by statistically significant changes in rating distributions). This procedure provides an example of screening all the initially considered contextual attributes and filtering out those that do not matter for a particular recommendation application. For example, we may conclude that the Time, Theater and Companion contexts matter, while the Weather context does not in the considered movie recommendation application.

3 Paradigms for Incorporating Context in Recommender Systems

The usage of contextual information in recommender systems can be traced to the work by Herlocker and Konstan [35], who hypothesized that the inclusion of knowledge about the user's *task* into the recommendation algorithm in certain applications can lead to better recommendations. For example, if we want to recommend books as gifts for a child, then we might want to specify several books that the child already has (and likes) and provide this information (i.e., a task profile) to the recommender system for calculating new recommendations. Note that this approach operates within the traditional 2D $User \times Item$ space, since the task specification for a specific user consists of a list of sample items; in other words, besides the standard *User* and *Item* dimensions, no additional contextual dimensions are used. However, this approach serves as a successful illustration of how additional relevant information (in the form of user-specified task-relevant item examples) can be incorporated into the standard collaborative filtering paradigm. Further, the use of interest scores assigned to topics has been applied to building contextual user profiles in recommender systems [72].

Different approaches to using contextual information in the recommendation process can be broadly categorized into two groups: (1) recommendation via *context-driven querying and search*, and (2) recommendation via *contextual preference elicitation and estimation*. The context-driven querying and search approach has been used by a wide variety of mobile and tourist recommender systems [2, 26, 67]. Systems using this approach 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 best matching resources (e.g., nearby restaurants that are currently open) to the user. One of the early examples of this approach is the Cyberguide project [2], which developed several *tour guide* prototypes for different hand-held platforms. Abowd et al. [2] discuss different architectures and features necessary to provide realistic tour guide services to mobile users and, more specifically, the role that the contextual knowledge of the user's current and past locations can play in the recommendation and guiding process. Among the many other examples of context-aware tourist guide

systems proposed in research literature we can mention GUIDE [30], INTRIGUE [13], COMPASS [67], and MyMap [31] systems.

The other general approach to using contextual information in the recommendation process, i.e., via contextual preference elicitation and estimation, represents a more recent trend in context-aware recommender systems literature [3, 52, 54, 71]. In contrast to the previously discussed context-driven querying and search approach (where the recommender systems use the current context information and specified current user's interest as queries to search for the most appropriate content), techniques that follow this second approach attempt to model and learn 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. To model users' context-sensitive preferences and generate recommendations, these techniques typically either adopt existing collaborative filtering, content-based, or hybrid recommendation methods to context-aware recommendation settings or apply various intelligent data analysis techniques from data mining or machine learning (such as Bayesian classifiers or support vector machines).

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. We do want to mention that it is possible to design applications that 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. For example, the UbiquiTO system [26], which implements a mobile tourist guide, provides intelligent adaptation not only based on the specific context information, but also uses various rule-based and fuzzy set techniques to adapt the application content based on the user preferences and interests. Similarly, the News@hand system [25] uses semantic technologies to provide personalized news recommendations that are retrieved using user's concept-based queries or calculated according to a specific user's (or a user group's) profile.

To start the discussion of the contextual preference elicitation and estimation techniques, note that, in its general form, a traditional 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, Figure 3 presents a general overview of the traditional 2D recommendation process, which includes three components: data (input), 2D recommender system (function), and recommendation list (output). Note that, as indicated in Figure 3, after the recommendation function is defined (or constructed) based on the available data, recommendation list 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.

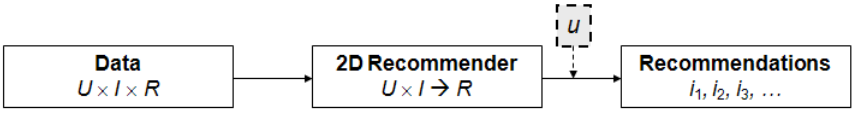


Fig. 3 General components of the traditional recommendation process.

As mentioned in Section 2.2, traditional recommender systems are built based on the knowledge of *partial user preferences*, i.e., user preferences for some (often limited) set of items, and the input data for traditional recommender systems is typically based on the records of 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 $< user, item, context, rating >$, where each specific record includes not only how much a given user liked a specific item, but also the contextual information in which the item was consumed by this user (e.g., *Context* = Saturday). Also, in addition to the descriptive information about users (e.g., demographics), items (e.g., item features), and ratings (e.g., multi-criteria rating information), context-aware recommender systems may also make use of additional context attributes, such as context hierarchies (e.g., Saturday $\rightarrow$ Weekend) mentioned in Section 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 traditional (non-contextual) recommender systems to generate context-aware recommendations? We will explore these questions in this chapter in more detail.

In the presence of available contextual information, following the diagrams in Figure 4, 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 Figure 3, which does not take into account the contextual information, we can apply the information about the current (or desired) 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 Figure 4:

- *Contextual pre-filtering* (or contextualization of recommendation input). In this recommendation paradigm (presented in Figure 4a), 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 traditional 2D recommender system on the selected data.
- *Contextual post-filtering* (or contextualization of recommendation output). In this recommendation paradigm (presented in Figure 4b), contextual information is initially ignored, and the ratings are predicted using any traditional 2D recom-

mender 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 Figure 4c), contextual information is used directly in the modeling technique as part of rating estimation.

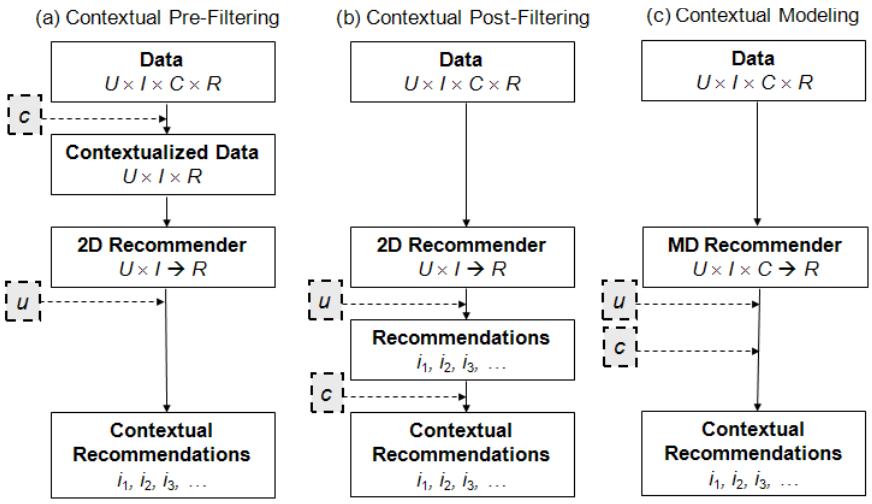


Fig. 4 Paradigms for incorporating context in recommender systems.

In the remainder of this section we will discuss these three approaches in detail.

3.1 Contextual Pre-Filtering

As shown in Figure 4a, the contextual pre-filtering approach uses contextual information to select or construct the most relevant 2D (*User* $\times$ *Item*) data for generating recommendations. One major advantage of this approach is that it allows deployment of any of the numerous traditional recommendation techniques previously proposed in the literature [5]. In particular, in one possible use of this approach, context c essentially serves as a *query* for selecting (filtering) relevant ratings data. An example of a contextual data filter for a movie recommender system would be: if a person wants to see a movie on Saturday, *only* the Saturday rating data is used to recommend movies. Note that this example represents an *exact pre-filter*. In other words, the data filtering query has been constructed using exactly the specified context.

For example, following the contextual pre-filtering paradigm, Adomavicius et al. [3] proposed a *reduction-based approach*, which reduces the problem of multidi-

mensional (MD) contextual recommendations to the standard 2D $User \times Item$ recommendation space. Therefore, as with any contextual pre-filtering approach, one important benefit of the reduction-based approach is that all the previous research on 2D recommender systems is directly applicable in the MD case after the reduction is done. In particular, let $R^D_{User \times Item}$: $U \times I \rightarrow Rating$ be *any* 2D rating estimation function that, given existing ratings D (i.e., D contains records $< user, item, rating >$ for each of the known, user-specified ratings), can calculate a prediction for any rating, e.g., $R^D_{User \times Item}(John, StarWars)$. Then, a 3-dimensional rating prediction function supporting the context of time can be defined similarly as $R^D_{User \times Item \times Time}$: $U \times I \times T \rightarrow Rating$, where D contains records $< user, item, time, rating >$ for the user-specified ratings. Then the 3-dimensional prediction function can be expressed through a 2D prediction function in several ways, including:

$$\forall (u,i,t) \in U \times I \times T, R^D_{User \times Item \times Time}(u,i,t) = R^{D[Time=t](User,Item,Rating)}_{User \times Item}(u,i).$$

Here $[Time = t]$ denotes a simple contextual pre-filter, and $D[Time = t](User, Item, Rating)$ denotes a rating dataset obtained from D by selecting only the records where $Time$ dimension has value t and keeping only the values for $User$ and $Item$ dimensions, as well as the value of the rating itself. I.e., if we treat a dataset of 3-dimensional ratings D as a relation, then $D[Time = t](User, Item, Rating)$ is simply another relation obtained from D by performing two relational operations: selection and, subsequently, projection.

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.

Context generalization. Adomavicius et al. [3] introduce the notion of *generalized pre-filtering*, which allows 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 $c_i \rightarrow c'_i$ for every $i = 1, \ldots, k$ in the corresponding context hierarchy. Then, c' (instead of c) can be used as a data query to obtain contextualized ratings data.

Following the idea of context generalization, Adomavicius et al. [3] proposed to use not a simple pre-filter $[Time = t]$, which represents the exact context t of the rating (u,i,t), but rather a generalized pre-filter $[Time \in S_t]$, where S_t denotes some

superset of context t. Here S_t is called a *contextual segment* [3]. For example, if we would like to predict how much John Doe would like to see the "Gladiator" movie on Monday, i.e., to calculate $R^D_{User \times Item \times Time}(JohnDoe, Gladiator, Monday)$, we could use not only other user-specified *Monday* ratings for prediction, but *Weekday* ratings in general. In other words, for every (u,i,t) where $t \in Weekday$, we can predict the rating as $R^D_{User \times Item \times Time}(u,i,t) = R^{D[Time \in Weekday](User,Item,AGGR(Rating))}_{User \times Item}(u,i)$. More generally, in order to estimate some rating $R(u,i,t)$, we can use some specific contextual segment S_t as: $R^D_{User \times Item \times Time}(u,i,t) = R^{D[Time \in S_t](User,Item,AGGR(Rating))}_{User \times Item}(u,i)$.

Note, that we have used the $AGGR(Rating)$ notation in the above expressions, since there may be several user-specified ratings with the same *User* and *Item* values for different *Time* instances in dataset D belonging to some contextual segment S_t (e.g., different ratings for *Monday* and *Tuesday*, all belonging to segment *Weekday*). Therefore, we have to aggregate these values using some aggregation function, e.g., averaging, when reducing the dimensionality of the recommendation space. The above 3-dimensional reduction-based approach can be extended to a general pre-filtering method reducing an arbitrary n-dimensional recommendation space to an m-dimensional one (where $m < n$). In this chapter we will assume that $m = 2$ because traditional recommendation algorithms are only designed for the two-dimensional $User \times Item$ case. Note, that there typically exist multiple different possibilities for context generalization, based on the context taxonomy and the desired context granularity. For example, let's assume that we have the following contextual taxonomies (*is-a* or *belongs-to* relationships) that can be derived from context hierarchies:

- *Company*: Girlfriend → Friends → NotAlone → AnyCompany;
- *Place*: Theater → AnyPlace;
- *Time*: Saturday → Weekend → AnyTime.

Then, the following are just several examples of possible generalizations c' of the above-mentioned context c = (Girlfriend, Theater, Saturday):

- c' = (Girlfriend, AnyPlace, Saturday);
- c' = (Friends, Theater, AnyTime);
- c' = (NotAlone, Theater, Weekend);

Therefore, choosing the "right" generalized pre-filter becomes an important problem. One option is to use a manual, expert-driven approach; e.g., always generalize specific days of week into more general Weekday or Weekend. Another option is to use a more automated approach, which could empirically evaluate the predictive performance of the recommender system on contextualized input datasets obtained from each generalized pre-filter, and then would automatically choose the pre-filter with best performance. An interesting and important research issue is how to deal with potential computational complexity of this approach due to context granularity; in other words, in cases of applications with highly granular contexts, there may exist a very large number of possible context generalizations, for which exhaustive search techniques would not be practical. For such cases, effective greedy approaches would need to be developed. Among the related work, Jiang

and Tuzhilin [38] 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.

Also note that the reduction-based approach is related to the problems of building local models in machine learning and data mining [10]. Rather than building the global rating estimation model utilizing all the available ratings, the reduction-based 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). It is important to know if a local model generated by the reduction-based approach outperforms the global model of the traditional 2D technique, where all the information associated with the contextual dimensions is simply ignored. For example, it is possible that it is better to use the contextual pre-filtering to recommend movies to see in the movie theaters on weekends, but use the traditional 2D technique for movies to see at home on VCRs. This is the case because the reduction-based approach, on the one hand, focuses recommendations on a *particular segment* and builds a local prediction model for this segment, but, on the other hand, computes these recommendations based on a *smaller* number of points limited to the considered segment. This tradeoff between having *more relevant* data for calculating an unknown rating based only on the ratings with the same or similar context and having *fewer* data points used in this calculation belonging to a particular segment (i.e., the *sparsity* effect) explains why the reduction-based recommendation method can outperform traditional 2D recommendation techniques on some segments and underperform on others. Which of these two trends dominates on a particular segment may depend on the application domain and on the specifics of the available data. Based on this observation, Adomavicius et al. [3] propose to combine a number of contextual pre-filters with the traditional 2D technique (i.e., as a default filter, where no filtering is done); this approach will be described as a case study in Section 4.

Among some recent developments, Ahn et al. [8] 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. [48] evaluate the effect of contextual information using a pre-filtering approach on the data obtained from an online retailer. Also, Baltrunas and Ricci [15] 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 [14] introduce the idea of *micro-profiling* (or user splitting), which splits the user profile into several (possibly overlapping) subprofiles, each representing the given user in a particular context. The predictions are done using these contextual micro-profiles instead of a single user model. Note that these data construction techniques fit well under the contextual pre-filtering paradigm, because they are following the same basic idea (as the data filtering techniques described earlier) – using contextual information to reduce the problem of multidimensional recommendations to the standard 2D $User \times Item$ space, which then allows to use any traditional recommendation techniques for rating prediction.

3.2 Contextual Post-Filtering

As shown in Figure 4b, the contextual post-filtering approach ignores context information in the input data when generating recommendations, i.e., when generating the ranked list of all candidate items from which any number of top-N recommendations can be made, depending on specific values of N. Then, the contextual post-filtering approach adjusts the obtained recommendation list for each user using contextual information. The recommendation list adjustments can be made by:

- *Filtering* out recommendations that are irrelevant (in a given context), or
- *Adjusting* the ranking of recommendations on the list (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 Figure 5.

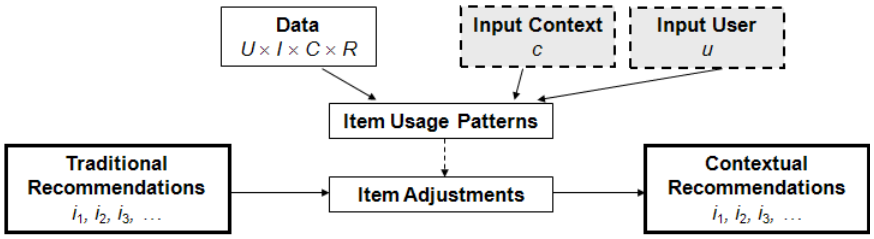


Fig. 5 Final phase of the contextual post-filtering approach: recommendation list adjustment.

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 (e.g., preferred actors to watch in a given context), and then use these attributes to adjust the recommendations, including:

- *Filtering* out recommended items that do not have a significant number of these characteristics (e.g., to be recommended, the movies must have at least two of the preferred actors in a given context), or
- *Ranking* recommended items based on how many of these relevant characteristics they have (e.g., the movies that star more of the user's preferred actors in a given context will be ranked higher).

In contrast, *model-based* post-filtering approaches can build predictive models that calculate the probability with which the user chooses a certain type of item in a given context, i.e., probability of relevance (e.g., likelihood of choosing movies of a certain genre in a given context), and then use this probability to adjust the recommendations, including:

- *Filtering* out recommended items that have the probability of relevance smaller than a pre-defined minimal threshold (e.g., remove movies of genres that have a low likelihood of being picked), or
- *Ranking* recommended items by weighting the predicted rating with the probability of relevance.

Panniello et al. [54] provide an experimental comparison of the exact pre-filtering method (discussed in Section 3.1) versus two different post-filtering methods – *Weight* and *Filter* – using several real-world e-commerce datasets. The *Weight* post-filtering method reorders the recommended items by weighting the predicted rating with the probability of relevance in that specific context, and the *Filter* post-filtering method filters out recommended items that have small probability of relevance in the specific context. Interestingly, the empirical results show that the *Weight* post-filtering method dominates the exact pre-filtering, which in turn dominates the *Filter* post-filtering method, thus, indicating that the best approach to use (pre- or post-filtering) really depends on a given application.

As was the case with the contextual pre-filtering approach, a major advantage of the contextual post-filtering approach is that it allows using any of the numerous traditional recommendation techniques previously proposed in the literature [5]. Also, similarly to the contextual pre-filtering approaches, incorporating context generalization techniques into post-filtering techniques constitutes an interesting issue for future research.

3.3 Contextual Modeling

As shown in Figure 4c, the contextual modeling approach uses contextual information *directly* in 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 traditional 2D recommendation functions, the contextual modeling approach gives rise to truly multidimensional recommendation functions, which essentially represent predictive models (built using decision tree, regression, probabilistic model, or other technique) or heuristic calculations that incorporate contextual information in addition to the user and item data, i.e., $Rating = R(User, Item, Context)$. A significant number of recommendation algorithms – based on a variety of heuristics as well as predictive modeling techniques – have been developed over the last 10-15 years, and some of these techniques can be extended from the 2D to the multidimensional recommendation settings. We present a few examples of multidimensional

heuristic-based and model-based approaches for contextual modeling [4] in the rest of this section.

3.3.1 Heuristic-Based Approaches

The traditional two-dimensional (2D) neighborhood-based approach [20, 61] can be extended to the multidimensional case, which includes the contextual information, in a straightforward manner by using an n-dimensional distance metric instead of the user-user or item-item similarity metrics traditionally used in such techniques. To see how this is done, consider an example of the $User \times Item \times Time$ recommendation space. Following the traditional nearest neighbor heuristic that is based on the weighted sum of relevant ratings, the prediction of a specific rating $r_{u,i,t}$ in this example can be expressed as (see [4] for additional details):

$$r_{u,i,t} = k \sum_{(u',i',t') \neq (u,i,t)} W((u,i,t),(u',i',t')) \times r_{u',i',t'},$$

where $W((u,i,t),(u',i',t'))$ describes the "weight" rating $r_{u',i',t'}$ carries in the prediction of $r_{u,i,t}$, and k is a normalizing factor. Weight $W((u,i,t),(u',i',t'))$ is typically inversely related to the distance between points (u,i,t) and (u',i',t') in multidimensional space, i.e., $dist[(u,i,t),(u',i',t')]$. In other words, the closer the two points are (i.e., the smaller the distance between them), the more weight $r_{u',i',t'}$ carries in the weighted sum. One example of such relationship would be $W((u,i,t),(u',i',t')) = 1/dist[(u,i,t),(u',i',t')]$, but many alternative specifications are also possible. As before, the choice of the distance metric *dist* is likely to depend on a specific application. One of the simplest ways to define a multidimensional *dist* function is by using the reduction-like approach (somewhat similar to the one described in Section 3.1), by taking into account only the points with the same contextual information, i.e.,

$$dist[(u,i,t),(u',i',t')] = \begin{cases} dist[(u,i),(u',i')], & \text{if } t = t' \\ +\infty, & \text{otherwise} \end{cases}$$

This distance function makes $r_{u,i,t}$ depend only on the ratings from the segment of points having the same values of time t. Therefore, this case is reduced to the standard 2-dimensional rating estimation on the segment of ratings having the same context t as point (u,i,t). Furthermore, if we further refine function $dist[(u,i),(u',i')]$ in so that it depends only on the distance between users when $i = i'$, then we would obtain a method that is similar to the pre-filtering approach described earlier. Moreover this approach easily extends to an arbitrary n-dimensional case by setting the distance d between two rating points to $dist[(u,i),(u',i')]$ if and only if the *contexts* of these two points are the same.

Other ways to define the distance function would be to use the weighted Manhattan distance metric, i.e.,

$$dist[(u,i,t),(u',i',t')] = w_1 d_1(u,u') + w_2 d_2(i,i') + w_3 d_3(t,t'),$$

or the weighted Euclidean distance metric, i.e.,

$$dist[(u,i,t),(u',i',t')] = \sqrt{w_1 d_1^2(u,u') + w_2 d_2^2(i,i') + w_3 d_3^2(t,t')}$$

where d_1, d_2, and d_3 are distance functions defined for dimensions User, Item, and Time respectively, and w_1, w_2, and w_3 are the weights assigned for each of these dimensions (e.g., according to their importance). In summary, distance function $dist[(u,i,t),(u',i',t')]$ can be defined in many different ways and, while in many systems it is typically computed between ratings of the same user or of the same item, it constitutes an interesting research problem to identify various more general ways to define this distance and compare these different ways in terms of predictive performance.

3.3.2 Model-Based Approaches

There have been several model-based recommendation techniques proposed in recommender systems literature for the traditional two-dimensional recommendation model [5]. Some of these methods can be directly extended to the multidimensional case, such as the method proposed in [12], who show that their 2D technique outperforms some of the previously known collaborative filtering methods.

The method proposed by Ansari et al. [12] combines the information about users and items into a single hierarchical regression-based Bayesian preference model that uses Markov Chain Monte Carlo (MCMC) techniques to estimate its parameters. Specifically, let's assume that there are N_U users, where each user u is defined by vector z_u of observed attributes of user u, such as gender, age, and income. Also assume that there are N_I items, where each item i is defined by vector w_i of attributes of the item, such as price, weight, and size. Let r_{ui} be the rating assigned to item i by user u, where r_{ui} is a real-valued number. Moreover, ratings r_{ui} are only known for some subset of all possible (user, item) pairs. Then the unknown rating estimation problem is defined as

$$r_{ui} = x'_{ui}\mu + z'_u\gamma_i + w'_i\lambda_u + e_{ui}, \;\; e_{ui} \sim N(0,\sigma^2), \; \lambda_u \sim N(0,\Lambda), \; \gamma_i \sim N(0,\Gamma)$$

where observed (known) values of the model are ratings r_{ui} assigned by user u for item i, user attributes z_u, item attributes w_i, and vector $x_{ui} = z_u \otimes w_i$, where $\otimes$ is the Kronecker product, i.e., a long vector containing all possible cross-product combinations between individual elements of z_u and w_i. Intuitively, this equation presents a regression model specifying unknown ratings r_{ui} in terms of the characteristics z_u of user u, the characteristics w_i of item i, and the interaction effects x_{ui} between them. Interaction effects arise from the hierarchical structure of the model and are intended to capture effects such as how the age of a user changes his or her preferences for certain genres of movies. Vector μ in the above equation represents unobserved (unknown) slope of the regression line, i.e., unknown coefficients in the regression model that need to be estimated, as discussed below. Vector γ_i rep-

resents weight coefficients specific to item i that determine idiosyncrasy of item i, i.e., the unobserved heterogeneity of item i that are not explicitly recorded in the item profiles, such as direction, music and acting for the movies. Similarly, vector λ_u represents weight coefficients specific to user u that determine idiosyncrasy of that user, i.e., the unobserved user effects (heterogeneity) of user u. Finally, the error term e_{ui} is normally distributed with mean zero and standard deviation σ. Further, we consider a hierarchical model and assume that regression parameters γ_i and λ_u are normally distributed as $\gamma_i \sim N(0,\Gamma)$ and $\lambda_u \sim N(0,\Lambda)$, where Γ and Λ are unknown covariance matrices. The parameters of the model are μ, σ^2, Λ, and Γ. They are estimated from the data containing the already known ratings using the MCMC methods described in [12].

While the approach presented in [12] is described in the context of the traditional two-dimensional recommender systems, it can be directly extended to include the contextual information. For example, assume that we have a third dimension *Time* that is defined by the following two attributes (variables): (a) Boolean variable *weekend* specifying whether a movie was seen on a weekend or not, and (b) a positive integer variable *numdays* indicating the number of days after the release when the movie was seen.

In such a case, the Ansari et al. [12] model can be extended to the third (*Time*) dimension as in Adomavicius and Tuzhilin [4]:

$$r_{uit} = x'_{uit}\mu + p'_{ui}\theta_t + q'_{it}\lambda_u + r'_{tu}\gamma_i + z'_u\delta_{it} + w'_i\pi_{tu} + y'_t\sigma_{ui} + e_{uit}$$

where $e_{uit} \sim N(0,\sigma^2)$, $\gamma_i \sim N(0,\Gamma), \lambda_u \sim N(0,\Lambda), \theta_t \sim N(0,\Theta), \delta_{it} \sim N(0,\Delta), \pi_{tu} \sim N(0,\Pi)$, and $\sigma_{ui} \sim N(0,\Sigma)$.

This model encompasses the effects of observed and unobserved user-, item- and temporal-variables and their interactions on rating r_{uit} of user u for movie i seen at time t. The variables z_u, w_i and y_t stand for the observed attributes of users (e.g., demographics), movies (e.g., genre) and time dimension (e.g., weekend, numdays). The vector x_{uit} represents the interaction effects of the user, movies, and time variables, and its coefficient μ represents the unobserved (unknown) slope of the regression line, i.e., unknown coefficients in the above regression model that need to be estimated, as discussed below. The vectors λ_u, γ_i and θ_t are random effects that stand for the unobserved sources of heterogeneity of users (e.g., their ethnic background), movies (e.g., the story, screenplay, etc.) and temporal effects (e.g., was the movie seen on a holiday or not, the season when it was released, etc). The vector p_{ui} represents the interaction of the observed user and item variables, and likewise q_{it} and r_{tu}. The vector σ_{ui} represents the interaction of the unobserved user and item attributes, and vectors π_{tu} and δ_{it} have similar types of interactions. Finally, the parameters $\mu, \sigma^2, \Lambda, \Gamma, \Theta, \Delta, \Pi$, and Σ of the model can be estimated from the data of the already known ratings using Markov Chain Monte Carlo (MCMC) methods, as was done in [12].

Finally, note that the number of parameters in the above model that would need to be estimated rises with the number of dimensions and, therefore, the sparsity of known ratings may become a problem. If this is a serious problem for a particular

application, then some of the terms in the above model can be dropped, leading to a simplified model. For example, we may decide to drop the term $q_{it}\lambda_u$, or perhaps some other terms, which would lead to a simpler model with fewer parameters. Note that this simpler model would still take into account the contextual information, e.g., time in this case. Furthermore, parameter estimation of this model can be very time consuming and not scalable. Therefore, one of the research challenges is to make such models more scalable and more robust in terms of the more accurate estimations of unknown ratings. Some of the initial ideas of how to make these approaches more scalable are presented in [66].

In addition to possible extensions of existing 2D recommendation techniques to multiple dimensions, there have also been some new techniques developed specifically for context-aware recommender systems based on the context modeling paradigm. For example, following the general contextual modeling paradigm, Oku et al. [52] propose to incorporate additional contextual dimensions (such as time, companion, and weather) directly into recommendation space and use machine learning technique to provide recommendations in a restaurant recommender system. In particular, they use 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. [52] empirically show that context-aware SVM significantly outperforms non-contextual SVM-based recommendation algorithm in terms of predictive accuracy and user's satisfaction with recommendations. Similarly, Yu et al. [71] 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.

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. Then Abbar et al. [1] describe how these services can be combined to form Context-Aware Recommender Systems (CARS) and deployed in order to provide superior context-aware recommendations.

In this section we described various ways to incorporate contextual information into recommendation algorithms within the framework of pre-filtering, post-filtering, and contextual modeling methods. Since CARS is a new and an emerging area of recommender systems, the presented methods constitute only the initial approaches to providing recommendations, and better-performing methods need and should be developed across all these three approaches.

In the next section, we discuss how these different individual methods can be combined together into one common approach.

4 Combining Multiple Approaches

As 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 [23, 24, 42, 55]. The three paradigms for context-aware recommender systems offer several different opportunities for employing combined approaches.

One possibility is to develop and combine several models of the same type. For example, Adomavicius et al. [3] followed this approach to develop a technique that combines information from several different contextual pre-filters. The rationale for having a number of different pre-filters is based on the fact that, as mentioned earlier, typically there can be multiple different (and potentially relevant) generalizations of the same specific context. For example, context c = (Girlfriend, Theater, Saturday) can be generalized to c_1 = (Friend, AnyPlace, Saturday), c_2 = (NotAlone, Theater, AnyTime), and a number of other contexts. Following this idea, Adomavicius et al. [3] use pre-filters based on the number of possible contexts for each rating, and then combine recommendations resulting from each contextual pre-filter. The general overview of this approach is shown in Figure 6. Note that the combination of several pre-filters can be done in multiple ways. For example, for a given context, (a) one could choose the best-performing pre-filter, or (b) use an "ensemble" of pre-filters. In the remainder of Section 4, we will discuss the approach developed by Adomavicius et al. [3] in more detail as a case study.

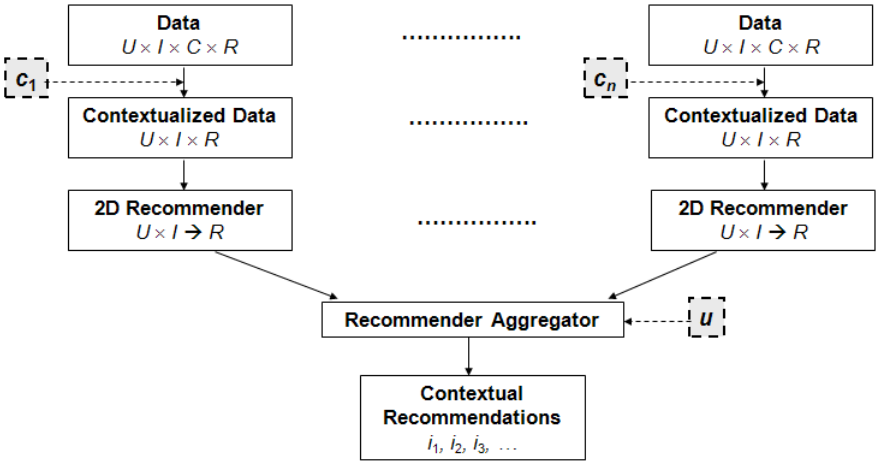


Fig. 6 Combining multiple pre-filters: an overview.

Another interesting possibility stems from an observation that complex contextual information can be split into several components, and the utility of each piece of contextual information may be different depending on whether it is used in the pre-filtering, post-filtering, or modeling stage. For example, time information (weekday

vs. weekend) may be most useful to pre-filter relevant data, but weather information (sunny vs. rainy) may be the most appropriate to use as a post-filter. Determining the utility of different contextual information with respect to different paradigms of context-aware recommender systems constitutes an interesting and promising direction for future research.

4.1 Case Study of Combining Multiple Pre-Filters: Algorithms

The combined approach to rating estimation consists of two phases [3]: (i) using known user-specified ratings (i.e., training data), determine the use of which pre-filters outperforms the traditional CF method; (ii) in order to predict a specific rating in a given context, choose the best pre-filter for that particular context and use the two-dimensional recommendation algorithm on this segment.

The first phase is a pre-processing phase and is usually performed "offline." It can work with any traditional 2D rating estimation method A, and consists of the following three steps [3]:

1. Among all possible generalized pre-filters, find the ones which result in contextual segments having a significantly large amount of data, i.e., segments with more than N ratings, where N is some predetermined threshold (e.g., $N = 250$ was used in that study). If the recommendation space is "small" (in the number of dimensions and the ranges of attributes in each dimension), the large segments can be obtained simply by doing an exhaustive search in the space of all possible segments. Alternatively, the help of a domain expert (e.g., a marketing manager) or some greedy heuristics could be used to determine the important large segments for the application.
2. For each generalized pre-filter c' determined in Step 1, algorithm A is run using this pre-filter and its predictive performance is determined using a chosen performance metric (this study used F-measure, which is defined as a harmonic mean of Precision and Recall – two standard decision support metrics widely used in information retrieval and, more recently, in recommender systems [36]). Only those pre-filters are kept, where the performance of algorithm A on contextually pre-filtered inputs exceeds the performance of the standard, i.e., non-filtered, version of algorithm A on that *same* data segment.
3. Among the remaining pre-filters, if there exist pre-filters c' and c'' such that $c' \rightarrow c''$ (i.e., c'' is strictly more general than c') and algorithm A demonstrates better performance using pre-filter c'' than pre-filter c', then c' is deemed to be redundant (less general *and* less accurate) and is removed from the set of pre-filters. The set of remaining contextual segments, denoted $SEGM^*$, constitutes the result of the "offline" phase of the combined approach.

Once the set of high-performing pre-filters $SEGM^*$ is computed, we can perform the second phase of the combined approach and determine which pre-filter to use in "real-time" when an actual recommendation needs to be produced. Given the

specific context c of recommendation, the best-performing pre-filter $c' \in SEGM^*$ such that $c \rightarrow c'$ or $c = c'$ is used in conjunction with algorithm A. If no such pre-filter c' exists, then the standard 2D non-filtered algorithm A (i.e., trained on the entire dataset) is used for rating prediction.

The main advantage of the combined pre-filtering approach described in this section is that it uses the contextual pre-filters *only* for those contextual situations where this method outperforms the standard 2D recommendation algorithm, and continues to use the latter where there is no improvement. Therefore, the combined approach is expected to perform equally well or better than the pure 2D approach in practice. The extent to which the combined approach can outperform the 2D approach depends on many different factors, such as the problem domain or quality of data.

4.2 Case Study of Combining Multiple Pre-Filters: Experimental Results

To illustrate how the combined approach presented in Section 4.1 performs in practice, it was evaluated on a real-world movie recommendation application and compared its performance with the traditional 2D CF method [3]. In this application, in addition to being asked to rate their movie-going experience, the users were asked to specify: (a) *time* when the movie was seen (choices: weekday, weekend, don't remember); furthermore, if seen on a weekend, was it the opening weekend for the movie (choices: yes, no, don't remember); (b) *place* where the movie was seen (choices: in a movie theater, at home, don't remember); and (c) *companion* with whom the movie was seen (choices: alone, with friends, with boyfriend/girlfriend, with family or others). Overall, 1755 ratings were entered by 117 students over a period of 12 months (May'01-Apr'02). Since some students rated very few movies, those students with fewer than 10 ratings were dropped in our analysis. Therefore, from an initial data, the final dataset had only 62 students, 202 movies and 1457 total ratings.

During the first step of the "offline", pre-filter selection phase, 9 large contextual segments were discovered, as presented in Table 1. However, after comparing the performance of the standard CF technique and the contextual pre-filtering CF technique on each of these segments (i.e., second step of the pre-filter selection phase), only 4 "high-performing" segments were found; one of them was subsequently removed after the redundancy check (i.e., third step of the pre-filter selection phase, as described in Section 4.1). The remaining set of 3 high-performing pre-filters is shown in Table 2, i.e., $SEGM^* = \{Theater\text{-}Weekend, Theater, Weekend\}$.

Finally, the resulting high-performing segments $SEGM^*$ were used in the "online", rating estimation phase. Overall, there were 1373 (of the 1457 collected) ratings that both 2D CF and the combined pre-filtering CF approaches were able to predict (not all ratings were predicted because of the sparsity-related limitations of data). The results show that the combined pre-filtering CF approach substantially outperformed the traditional 2D CF (1^{st} row in Table 3).

Table 1 Large contextual segments generated in Step 1 of the pre-filter selection algorithm.

Name	Size	Description
Home	727	Movies watched at home
Friends	565	Movies watched with friends
NonRelease	551	Movies watched on other than the opening weekend
Weekend	538	Movies watched on weekends
Theater	526	Movies watched in the movie theater
Weekday	340	Movies watched on weekdays
GBFriend	319	Movies watched with girlfriend/boyfriend
Theater-Weekend	301	Movies watched in the movie theater on weekends
Theater-Friends	274	Movies watched in the movie theater with friends

Table 2 High-performing large contextual segments.

Segment	CF: Segment-trained F-measure	CF: Whole-data-trained F-measure
Theater-Weekend	0.641	0.528
Theater	0.608	0.479
Weekend	0.542	0.484

Note that, as discussed earlier, the combined pre-filtering CF approach incorporates the standard CF approach, since it would use the standard 2D CF to predict the value of any rating that does not belong to any of the discovered high-performing pre-filters. Consequently, in this application, the predictions of the two approaches are identical for all ratings that do not belong to any segment in {*Theater-Weekend*, *Theater*, *Weekend*}. Since such ratings do not contribute to the differentiation between the two approaches, it is important to determine how well the two approaches do on the ratings from $SEGM^*$. In this case, there were 743 such ratings in $SEGM^*$ (out of 1373), and the difference in F-measure performance of the two approaches is 0.095 (2^{nd} row in Table 3), which is even more substantial than for the previously described case.

Table 3 Overall comparison based on F-measure.

Comparison	Overall F-measure		Difference between F-measures
	Standard 2D CF	Combined reduction-based CF	
All predictions (1373 ratings)	0.463	0.526	**0.063**
Predictions on ratings from $SEGM^*$ (743 ratings)	0.450	0.545	**0.095**

In this section, we discussed combining multiple pre-filtering, post-filtering, and contextual modeling methods to generate better predictions, focusing primarily on combining multiple pre-filters, based on [3]. We outlined only some of the main ideas, while leaving most of the problems in this area as wide open and subject of future research. We believe that creative combinations of multiple methods using ensemble techniques from machine learning can significantly increase performance of CARS and constitute an important and interesting area of research.

5 Additional Issues in Context-Aware Recommender Systems

In addition to the three paradigms of incorporating context in recommender systems and the methods of combining these three paradigms, there are several other important topics in context-aware recommender systems, such as how to better utilize contextual information, how to develop richer interaction capabilities with CARS that make recommendations more flexible, and how to build high-performing CARS systems. We discuss these issues in the rest of this section.

Studying Tradeoffs Between Pre-, Post-Filtering, and Contextual Modeling Approaches and Developing Better Understanding of How to Combine Them. In Section 3, we only described three types of general CARS paradigms and did not consider tradeoffs between them. In order to achieve better understanding of pre-filtering, post-filtering, and contextual modeling approaches, it is important to conduct studies comparing all the three approaches and identify relative advantages and disadvantages of these methods. One such study is described in [54], where a pre-filtering method is compared to a particular type of post-filtering method in terms of the quality of recommendations. It was shown that neither method dominated the other in terms of providing better recommendations, as measured using the F-measure. Furthermore, Panniello et al. [54] proposed a procedure identifying a set of conditions under which the pre-filtering method should be used vis-à-vis the post-filtering methods.

The work reported in [54] constitutes only the first step towards a systematic program of comparing the pre-filtering, post-filtering, and contextual modeling methods, and much more work is required to develop comprehensive understanding of the relative merits of each of the three methods and to understand which methods perform better than others and under which conditions.

Similarly, more work is required to better understand the combined approach discussed in Section 4. This is a fruitful area of research that is open to numerous improvements and important advances including various types of deployments of ensemble methods combining the pre- and post-filtering as well contextual modeling approaches.

Developing Richer Interaction and More Flexible Recommendation Capabilities of CARS. Context-aware recommendations have the following two important properties:

- *Complexity.* Since CARS involve not only users and items in the recommendation process, but also various types of contextual information, the types of such recommendations can be significantly more complex in comparison to the traditional 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.
- *Interactivity.* The contextual information 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 being "hard-wired" into the recommendation engines provided by most of the current vendors that, primarily, focus on recommending top-*N* items to the user and vice versa. 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 interface (UI).

Such flexible context-aware recommendations can be supported in several ways. First, Adomavicius et al. [6] developed a *recommendation query language* REQUEST[4] that 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. REQUEST is based on the multidimensional contextual recommendation model described in Section 2.2 and also in [3]. REQUEST supports a wide variety of features, and the interested reader can find the detailed account of these features as well as the formal syntax and various properties of the language in [6]. In addition, Adomavicius et al. [6] provide a discussion of the expressive power of REQUEST and present a multidimensional recommendation algebra that provides the theoretical basis for this language.

So far, we have briefly mentioned only the recommendation query language itself. Since a major argument for introducing such a language is its use by the end-users, it is also very important to develop simple, friendly, and expressive user interfaces (UIs) for supporting flexible but sometimes complex contextual recommendations. High-quality UIs should reduce the complexity and simplify interactions between the end-users and the recommender system and make them available to wider audiences. Developing such UIs constitutes a topic of future research.

Another proposal to provide flexible recommendations is presented in [43], where the FlexRecs system and framework are described. FlexRecs approach supports flexible recommendations over structured data by decoupling the definition of

[4] REQUEST is an acronym for REcommendation QUEry STatements.

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.

In addition to developing languages for expressing context-aware recommendations, it is also important to provide appropriate user interfaces so that the users were able to express flexible recommendations in an interactive manner. For FlexRecs, this entails building a UI for defining and managing recommendation workflows, and for REQUEST this entails providing front-end UI allowing users to express REQUEST queries using visual and interactive methods.

Developing High-Performing CARS Systems and Testing Them on Practical Applications. Most of the work on context-aware recommender systems has been conceptual, where a certain method has been developed, tested on some (often limited) data, and shown to perform well in comparison to certain benchmarks. There has been little work done on developing novel data structures, efficient storage methods, and new systems architectures for CARS. One example of such work is the paper by Hussein et al. [37], 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. The implementation is done using global software architecture of CARS developed by the authors and described in [1]. These two papers constitute only the initial steps towards developing better understanding of how to build more user-friendly, scalable, and better performing CARS systems, and much more work is required to achieve this goal.

6 Conclusions

In this chapter we argued that relevant contextual information does matter in recommender systems and that it is important to take this contextual information into account when providing recommendations. We also explained that the contextual information can be utilized at various stages of the recommendation process, including at the pre-filtering and the post-filtering stages and also as an integral part of the contextual modeling. We have also showed that various techniques of using the contextual information, including these three methods, can be combined into a single recommendation approach, and we presented a case study describing one possible way of such combining.

Overall, the field of context-aware recommender systems (CARS) is a relatively new and underexplored area of research, and much more work is needed to explore it comprehensively. We provided suggestions of several possible future research directions that were presented throughout the paper. In conclusion, CARS constitutes a newly developing and promising research area with many interesting and practically important research problems.

Acknowledgements Research of G. Adomavicius was supported in part by the National Science Foundation grant IIS-0546443, USA. The authors thank YoungOk Kwon for editorial assistance.

References

1. S. Abbar, M. Bouzeghoub, and S. Lopez. Context-aware recommender systems: A service-oriented approach. *VLDB PersDB Workshop*, 2009.
2. G. D. Abowd, C. G. Atkeson, J. Hong, S. Long, R. Kooper, and M. Pinkerton. Cyberguide: A mobile context-aware tour guide. *Wireless Networks*, 3(5):421–433, 1997.
3. G. Adomavicius, R. Sankaranarayanan, S. Sen, and A. Tuzhilin. Incorporating contextual information in recommender systems using a multidimensional approach. *ACM Transactions on Information Systems (TOIS)*, 23(1):103–145, 2005.
4. G. Adomavicius and 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.
5. G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734–749, 2005.
6. G. Adomavicius, A. Tuzhilin, and R. Zheng. REQUEST: A query language for customizing recommendations. *Information System Research*, 2010.
7. R. Agrawal, R. Rantzau, and E. Terzi. Context-sensitive ranking. In *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, pages 383–394. ACM, 2006.
8. H. Ahn, K. Kim, and 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,*.
9. G. Akrivas, M. Wallace, G. Andreou, G. Stamou, and S. Kollias. Context-sensitive semantic query expansion. In *Proceedings of the IEEE international conference on artificial intelligence systems (ICAIS)*, pages 109–114. Divnomorskoe, Russia, 2002.
10. E. Alpaydin. *Introduction to machine learning*. The MIT Press, 2004.
11. S. S. Anand and B. Mobasher. Contextual recommendation. *WebMine, LNAI*, 4737:142–160, 2007.
12. A. Ansari, S. Essegaier, and R. Kohli. Internet recommendation systems. *Journal of Marketing Research*, 37(3):363–375, 2000.
13. L. Ardissono, A. Goy, G. Petrone, M. Segnan, and P. Torasso. Intrigue: personalized recommendation of tourist attractions for desktop and hand held devices. *Applied Artificial Intelligence*, 17(8):687–714, 2003.
14. L. Baltrunas and X. Amatriain. Towards time-dependant recommendation based on implicit feedback. In *Workshop on Context-Aware Recommender Systems (CARS 2009)*. New York, 2009.
15. L. Baltrunas and F. Ricci. Context-dependent items generation in collaborative filtering. In *Workshop on Context-Aware Recommender Systems (CARS 2009)*. New York, 2009.

16. M. Bazire and P. Brzillon. Understanding context before using it. In A. Dey and et al., editors, *Proceedings of the 5th International Conference on Modeling and Using Context*. Springer-Verlag, 2005.
17. M. J. Berry and G. Linoff. *Data mining techniques: for marketing, sales, and customer support*. John Wiley & Sons, Inc. New York, NY, USA, 1997.
18. J. R. Bettman, M. F. Luce, and J. W. Payne. Consumer decision making: A constructive perspective. pages 1–42, 1991.
19. S. Boutemedjet and D. Ziou. A graphical model for context-aware visual content recommendation. *IEEE Transactions on Multimedia*, 10(1):52–62, 2008.
20. J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, volume 461, pages 43–52. San Francisco, CA, 1998.
21. B. Brown, M. Chalmers, M. Bell, M. Hall, I. MacColl, and P. Rudman. Sharing the square: collaborative leisure in the city streets. In H. Gellersen, K. Schmidt, M. Beaudouin-Lafon, and W. E. Mackay, editors, *Proceedings of the ninth conference on European Conference on Computer Supported Cooperative Work*, pages 427–447. Springer, 2005.
22. P. J. Brown, J. D. Bovey, and X. Chen. Context-aware applications: from the laboratory to the marketplace. *IEEE Personal Communications*, 4:58–64, 1997.
23. R. Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, 2002.
24. R. Burke. Hybrid web recommender systems. *The Adaptive Web*, pages 377–408, 2007.
25. I. Cantador and P. Castells. Semantic contextualisation in a news recommender system. In *Workshop on Context-Aware Recommender Systems (CARS 2009)*. New York, 2009.
26. F. Cena, L. Console, C. Gena, A. Goy, G. Levi, S. Modeo, and I. Torre. Integrating heterogeneous adaptation techniques to build a flexible and usable mobile tourist guide. *AI Communications*, 19(4):369–384, 2006.
27. S. Chatterjee, A. S. Hadi, and B. Price. *Regression analysis by example*. John Wiley and Sons, 2000.
28. S. Chaudhuri and U. Dayal. An overview of data warehousing and olap technology. *ACM Sigmod record*, 26(1):65–74, 1997.
29. G. Chen and D. Kotz. A survey of context-aware mobile computing research. *Technical Report TR2000-381, Dartmouth Computer Science*, 2000.
30. K. Cheverst, N. Davies, K. Mitchell, A. Friday, and 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*, pages 17–24. ACM, 2000.
31. B. De Carolis, I. Mazzotta, N. Novielli, and 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.
32. A. K. Dey, G. D. Abowd, and D. Salber. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Human-Computer Interaction*, 16(2):97–166, 2001.
33. P. Dourish. What we talk about when we talk about context. *Personal and ubiquitous computing*, 8(1):19–30, 2004.
34. D. Franklin and J. Flachsbart. All gadget and no representation makes jack a dull environment. In *Proceedings of the AAAI 1998 Spring Symposium on Intelligent Environments*, pages 155–160. AAAI Press, 1998.
35. J. L. Herlocker and J. A. Konstan. Content-independent task-focused recommendation. *IEEE Internet Computing*, pages 40–47, 2001.
36. J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)*, 22(1):5–53, 2004.
37. T. Hussein, T. Linder, W. Gaulke, and J. Ziegler. Context-aware recommendations on rails. 2009.
38. T. Jiang and A. Tuzhilin. Improving personalization solutions through optimal segmentation of customer bases. *IEEE Transactions on Knowledge and Data Engineering*, 21(3):305–320, 2009.

39. G. J. F. Jones, D. Glasnevin, and I. Gareth. Challenges and opportunities of context-aware information access. In *International Workshop on Ubiquitous Data Management*, pages 53–62, 2005.
40. R. Kimball and M. Ross. The data warehousing toolkit. *John Wiley & Sons, New York*, 1996.
41. D. Koller and M. Sahami. Toward optimal feature selection. In *Proceedings of the 13th International Conference on Machine Learning*, pages 284–292. Morgan Kaufmann, 1996.
42. 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*, pages 426–434. ACM, New York, NY, 2008.
43. G. Koutrika, B. Bercovitz, and H. Garcia-Molina. Flexrecs: expressing and combining flexible recommendations. In *Proceedings of the 35th SIGMOD international conference on Management of data*, pages 745–758. ACM, Providence, RI, 2009.
44. S. Lawrence. Context in web search. *IEEE Data Engin. Bulletin*, 23(3):25, 2000.
45. D. B. Leake and R. Scherle. Towards context-based search engine selection. In *Proceedings of the 6th international conference on Intelligent user interfaces*, pages 109–112. ACM New York, NY, USA, 2001.
46. G. L. Lilien, P. Kotler, and K. S. Moorthy. *Marketing models*. Prentice Hall, 1992.
47. H. Liu and H. Motoda. *Feature selection for knowledge discovery and data mining*. Springer, 1998.
48. S. Lombardi, S. S. Anand, and M. Gorgoglione. Context and customer behavior in recommendation. In *Workshop on Context-Aware Recommender Systems (CARS 2009)*. New York.
49. D. A. Lussier and R. W. Olshavsky. Task complexity and contingent processing in brand choice. *Journal of Consumer Research*, pages 154–165, 1979.
50. Z. Maamar, D. Benslimane, and N. C. Narendra. What can context do for web services? *Communications of the ACM*, 49(12):98–103, 2006.
51. M. F. Mokbel and J. J. Levandoski. Toward context and preference-aware location-based services. In *Proceedings of the Eighth ACM International Workshop on Data Engineering for Wireless and Mobile Access*, pages 25–32. ACM, 2009.
52. K. Oku, S. Nakajima, J. Miyazaki, and S. Uemura. Context-aware SVM for context-dependent information recommendation. In *Proceedings of the 7th International Conference on Mobile Data Management*, page 109, 2006.
53. C. Palmisano, A. Tuzhilin, and M. Gorgoglione. Using context to improve predictive modeling of customers in personalization applications. *IEEE Transactions on Knowledge and Data Engineering*, 20(11):1535–1549, 2008.
54. U. Panniello, A. Tuzhilin, M. Gorgoglione, C. Palmisano, and 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*, pages 265–268. ACM, 2009.
55. D. M. Pennock and E. Horvitz. Collaborative filtering by personality diagnosis: A hybrid memory-and model-based approach. In *IJCAI'99 Workshop: Machine Learning for Information Filtering*, 1999.
56. C. K. Prahalad. Beyond CRM: CK Prahalad predicts customer context is the next big thing. *American Management Association MwWorld*, 2004.
57. R. Ramakrishnan and J. Gehrke. *Database Management Systems*. USA: McGraw Hill Companies, 2000.
58. F. Ricci and Q. N. Nguyen. Mobyrek: A conversational recommender system for on-the-move travelers. *Destination Recommendation Systems: Behavioural Foundations and Applications*, pages 281–294, 2006.
59. T. Rodden, K. Cheverst, K. Davies, and A. Dix. Exploiting context in hci design for mobile systems. In *Workshop on Human Computer Interaction with Mobile Devices*, pages 21–22, 1998.
60. N. Ryan, J. Pascoe, and D. Morse. *Enhanced Reality Fieldwork: the Context-Aware Archaeological Assistant. Gaffney, V., van Leusen, M., Exxon, S.(eds.) Computer Applications in Archaeology*. British Archaeological Reports, Oxford, 1997.

61. B. Sarwar, G. Karypis, J. Konstan, and J. Reidl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295. ACM, 2001.
62. B. N. Schilit and M. M. Theimer. Disseminating active map information to mobile hosts. *IEEE network*, 8(5):22–32, 1994.
63. J. H. Schiller and A. Voisard. *Location-based services*. Morgan Kaufmann, 2004.
64. A. Sieg, B. Mobasher, and R. Burke. Representing context in web search with ontological user profiles. In *Proceedings of the 6th International Conference on Modeling and Using Context*, 2007.
65. K. Stefanidis, E. Pitoura, and P. Vassiliadis. A context-aware preference database system. *International Journal of Pervasive Computing and Communications*, 3(4):439–600, 2007.
66. A. Umyarov and A. Tuzhilin. Using external aggregate ratings for improving individual recommendations. *ACM Transactions on the Web*, 2010.
67. M. Van Setten, S. Pokraev, and J. Koolwaaij. Context-aware recommendations in the mobile tourist application compass. In W. Nejdl and P. De Bra, editors, *Adaptive Hypermedia*, pages 235–244. Springer Verlag, 2004.
68. A. Ward, A. Jones, and A. Hopper. A new location technique for the active office. *IEEE Personal Communications*, 4(5):42–47, 1997.
69. N. Webster. *Webster's new twentieth century dictionary of the English language*. Springfield, MA: Merriam-Webster, Inc., 1980.
70. W. Woerndl, C. Schueller, and R. Wojtech. A hybrid recommender system for context-aware recommendations of mobile applications. In *Proceedings of the 3rd International Workshop on Web Personalization, Recommender Systems and Intelligent User Interfaces*, pages 871–878, 2007.
71. Z. Yu, X. Zhou, D. Zhang, C. Y. Chin, X. Wang, and J. Men. Supporting context-aware media recommendations for smart phones. *IEEE Pervasive Computing*, 5(3):68–75, 2006.
72. C. N. Ziegler, S. M. McNee, J. A. Konstan, and G. Lausen. Improving recommendation lists through topic diversification. In *Proceedings of the 14th international conference on World Wide Web*, pages 22–32. Chiba, Japan, 2005.